Δυναμικός Προγραμματισμός

Α. Κ. Καπόρης

9 Απριλίου 2011

Περιεχόμενα

1 Δυναμικός Προγραμματισμός **5**

1.1 Υπολογισμός του $\binom{n}{k}$ . . . 5

1.1.1 Κατασκευή της Αναδρομής (1.1) . . . 5

1.1.2 Απομνημόνευση της Αναδρομής (1.1) . . . 6

1.2 Εξυπηρέτηση χρόνο-εξαρτώμενων εργασιών με βάρη (Weighted interval scheduling) . . . 9

1.2.1 Περιγραφή προβλήματος . . . 9

1.2.2 Κατασκευή της αναδρομής . . . 9

1.2.3 Απομνημόνευση της Αναδρομής (1.3) . . . 11

1.3 Εργασίες δοσμένου συνολικού βάρους (Subset sum) . . . 13

1.3.1 Περιγραφή προβλήματος . . . 13

1.3.2 Κατασκευή της Αναδρομής . . . 14

1.3.3 Απομνημόνευση της Αναδρομής (1.12) . . . 16

1.4 Ο σάκος του ληστή (Knapsack) . . . 17

1.4.1 Περιγραφή του Προβλήματος . . . 17

1.4.2 Κατασκευή της Αναδρομής . . . 18

1.4.3 Απομνημόνευση της Αναδρομής . . . 19

1.5 Μέγιστη αύξουσα υπακολουθία (Longest increasing subsequence) . . . 19

1.5.1 Περιγραφή του Προβλήματος . . . 19

1.5.2 Κατασκευή της Αναδρομής . . . 21

1.5.3 Απομνημόνευση της Αναδρομής . . . 22

1.6 Ανταγωνιστική τοποθέτηση εταιριών σε Εθνική Οδό (Competing firms) . . . 22

1.6.1 Περιγραφή του Προβλήματος . . . 22

1.6.2 Κατασκευή της Αναδρομής . . . 23

1.6.3 Απομνημόνευση της Αναδρομής . . . 24

1.7 Μέθοδος Ελαχίστων Τετραγώνων σε τομείς (Segmented Least Squares) . . . 25

1.7.1 Περιγραφή του Προβλήματος . . . 25

1.7.2 Κατασκευή της Αναδρομής . . . 28

1.7.3 Απομνημόνευση της Αναδρομής . . . 29

1.8 Πολλαπλασιασμός αλληλουχίας πινάκων (Matrix chain multiplication) . . . 29

1.8.1 Περιγραφή του Προβλήματος . . . 29

1.8.2 Κατασκευή της Αναδρομής . . . 31

1.8.3 Απομνημόνευση της Αναδρομής . . . 31

1.9 RNA δευτερεύουσα δομή . . . 32

1.9.1 Περιγραφή του Προβλήματος . . . 32

1.9.2 Κατασκευή της Αναδρομής . . . 34

1.9.3 Απομνημόνευση της Αναδρομής . . . 35

1.10 Βέλτιστα Δυαδικά Δέντρα Αναζήτησης (Optimal Binary Search Trees) . . . 36

1.10.1 Περιγραφή του Προβλήματος . . . 36

1.10.2 Κατασκευή της Αναδρομής . . . 38

1.10.3 Απομνημόνευση της Αναδρομής 39
1.10.4 Παράδειγμα 39
1.11 Ταίριασμα λέξεων (Sequence alignment) 41
1.11.1 Περιγραφή του Προβλήματος 41
1.11.2 Κατασκευή της Αναδρομής 42
1.11.3 Απομνημόνευση της Αναδρομής 43
1.11.4 Παράδειγμα 44
1.12 Μονοπάτι ελάχιστου κόστους (Shortest paths.) **under construction!!!** 46
1.12.1 Περιγραφή του Προβλήματος 46
1.12.2 Κατασκευή της Αναδρομής 46
1.12.3 Απομνημόνευση της Αναδρομής 46
1.13 Ένα πρόβλημα Περιοδεύοντος Πωλητή (A Traveling Salesman problem) **under construction!!!** 46
1.13.1 Περιγραφή του Προβλήματος 46
1.13.2 Κατασκευή της Αναδρομής 47
1.13.3 Απομνημόνευση της Αναδρομής 48

Κεφάλαιο 1

Δυναμικός Προγραμματισμός

Διαισθητικά, το κλειδί είναι να λύσουμε το τρέχον πρόβλημα ορισμένου μεγέθους λύνοντας μια σειρά από ανεξάρτητα προβλήματα *γνησίως μικρότερου* μεγέθους. Το πετυχαίνουμε με μια *αναδρομική σχέση* που συνδέει το τρέχον πρόβλημα με τα μικρότερα (και λίγα) προβλήματα. Η εύρεση της αναδρομικής σχέσης είναι το πρώτο (από τα δύο) δύσκολο σημείο στο δυναμικό προγραμματισμό.

Στο πιο κάτω παράδειγμα θεωρούμε γνωστή την αναδρομική σχέση.

Επικεντρώνουμε στο δεύτερο δύσκολο σημείο. Δηλαδή στο τρόπο της *αλγοριθμικής χρήσης* της αναδρομικής σχέσης ώστε να λύσουμε γρήγορα το πρόβλημα. Πρέπει να κατανοήσουμε αυτό τον απλό μα και ισχυρό αλγοριθμικό τρόπο, διότι θα εφαρμοστεί σε όλα τα παραδείγματα που εξετάζουμε σε αυτό το κεφάλαιο.

1.1 Υπολογισμός του $\binom{n}{k}$

Μας ζητάνε να υπολογίσουμε το *Δυωνυμικού Συντελεστή* $C(n,k) = \binom{n}{k}$ με δυναμικό προγραμματισμό. Ο υπολογισμός του $C(n,k)$ εξαρτάται από τον φυσικό n που το καλούμε *μέγεθος* του προβλήματος $C(n,k)$.

1.1.1 Κατασκευή της Αναδρομής (1.1)

- **1ο βήμα:** Η ιδέα είναι να καταφέρουμε να βρούμε μια αναδρομική σχέση που να συνδέει το παρόν πρόβλημα $C(n,k) = \binom{n}{k}$ μεγέθους n σε *πολυωνυμικού* πλήθους *μικρότερα* και *ανεξάρτητα* υποπροβλήματα όπου το κάθε ένα θα έχει μέγεθος $< n$.

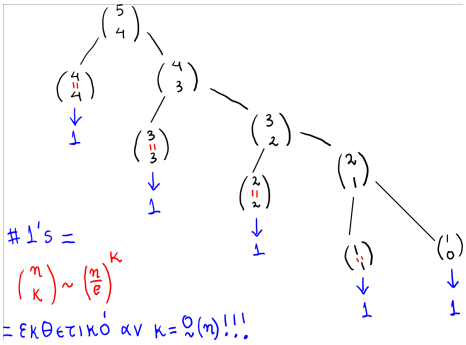
Στην περίπτωση του προβλήματος $C(n,k)$ ξέρουμε από τα Διακριτά Μαθηματικά ότι η αναδρομική σχέση είναι:

$$C(n,k) = \begin{cases} C(n-1,k) + C(n-1,k-1), & \text{για } 1 \leq k < n, \\ 1, & \text{για } n = k, \text{ και } k = 0 \end{cases} \tag{1.1}$$

Παρατηρούμε στη Σχέση (1.1) ότι το αρχικό πρόβλημα $C(n,k)$ μεγέθους n έχει αναχθεί αναδρομικά σε 2 μικρότερα υποπροβλήματα $C(n-1,k)$ και $C(n-1,k-1)$ μεγέθους $n-1$.

- **Αποτέλεσμα 1ου βήματος:** Καταφέραμε κάτι πολύ σημαντικό με την αναδρομική Σχέση (1.1): ανά 1 αναδρομικό βήμα *μειώνουμε* το μέγεθος του προβλήματος κατά 1. Συνεπώς μετά από το πολύ n αναδρομικά βήματα το μέγεθος του προβλήματος θα γίνει 0 και εύκολο να υπολογιστεί.

Η μόνη δυσκολία που απομένει είναι να λύσουμε το πρόβλημα $C(n,k)$ με την βοήθεια των υποπροβλημάτων που προκύπτουν από τη Σχέση (1.1). Ένας βιαστικός τρόπος είναι να λύσουμε αναδρομικά την (1.1). Όμως το Σχήμα (1.1) κάνει φανερό ότι αυτό θα μας κοστίσει εκθετικό χρόνο υπολογισμού.



Σχήμα 1.1: Αναπτύσσοντας αναδρομικά τη Σχέση (1.1) για το $\binom{5}{4}$ λαμβάνουμε το άνω δέντρο. Ο χρόνος υπολογισμού είναι όσα και τα φύλλα του με τη τιμή 1. Διότι κάθε μονοπάτι αναδρομικών επικλήσεων της αναδρομικής σχέσης τερματίζει μόλις λάβουμε τη τιμή 1. Γενικότερα, ο υπολογισμός του $\binom{n}{k}$ θα δημιουργήσει ένα δέντρο με ασυμπτωτικά $\left(\frac{n}{e}\right)^k$ φύλλα με τιμή 1. Το τεράστιο μειονέκτημα της μεθόδου είναι ότι αριθμός των φύλλων (χρόνος υπολογισμού) γίνεται *εκθετικά* μεγάλος αν το $k = \Omega(n)$.

1.1.2 Απομνημόνευση της Αναδρομής (1.1)

Μπορούμε να κάνουμε κάτι πιο έξυπνο. Μπορούμε να αποφύγουμε τον εκθετικό χρόνο υπολογισμού του $C(n,k)$ αν *απομνημονεύουμε* δυναμικά κάθε όρο $C(i,j)$ με $i = 0,\ldots,n$ και $j = 0,\ldots,k$. Ο τρόπος αυτός παρουσιάζεται στο Σχήμα (1.2) όπου υπολογίζουμε το $\binom{10}{4}$ με χρήση ενός Πίνακα $M[i = 0\ldots10, j = 0\ldots4]$ όπου σε κάθε θέση $M[i,j]$ αποθηκεύουμε την αντίστοιχη τιμή $C(i,j)$.

1. Αρχικά απομνημονεύουμε τις εύκολες τιμές (μπλε) στο 1ο πίνακα του Σχήματος (1.2). Από τη Σχέση (1.1), οι εύκολες (άμεσες) τιμές είναι η 1. Προκύπτει από το κάτω κλάδο της Σχέσης (1.1). Λόγω του κάτω κλάδου, αποθηκεύουμε $M[i,0] = \binom{i}{0} = 1, i = 0,\ldots,10$ στη στήλη $k = 0$ του 1ου πίνακα στο Σχήμα (1.2). Λόγω του κάτω κλάδου, πάλι, αποθηκεύουμε $M[i,i] = \binom{i}{i} = 1, i = 0,\ldots,10$ στη διαγώνιο του ίδιου σχήματος.

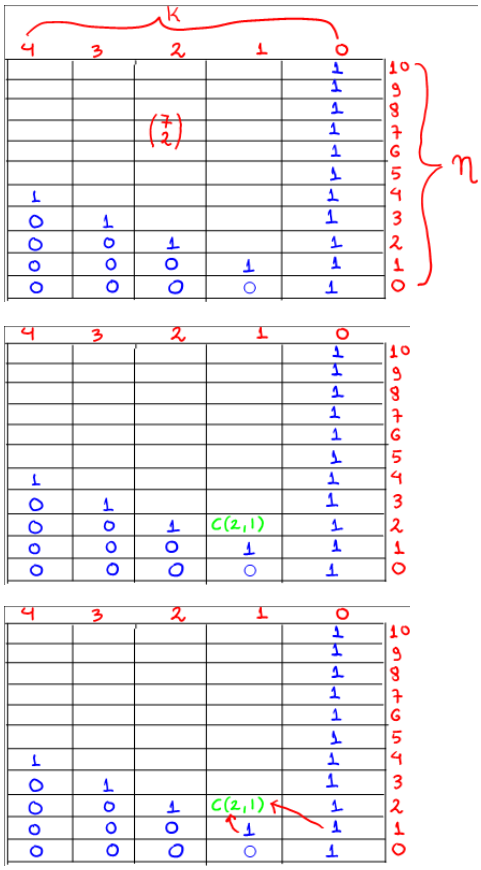
2. **Χρόνος αποθήκευσης σε μπλε κουτάκια.** είναι ο χρόνος $O(\max\{n,k\})$ ολοκλήρωσης του αλγορίθμου

   for$(i = 0; i <= k; i++)$
   $\quad M[i,i] = 1;$
   for$(i = 0; i <= n; i++)$
   $\quad M[i,0] = 1;$

3. **Μαγική παρατήρηση:** Στο 1ο πίνακα του Σχήματος (1.2) τα άδεια κουτάκια που απομένουν (εξαιρώντας τα μπλε) είναι $<$ γραμμές $\times$ στήλες. Αν καταφέρουμε να γεμίσουμε διαδοχικά το ένα μετά το άλλο όλα τα κουτάκια, γεμίζοντας κάθε ένα κουτί σε σταθερό χρόνο, τότε πληρώνουμε συνολικό χρόνο $<$ γραμμές $\times$ στήλες $=$ όλα τα κουτιά. Έτσι, επιτυγχάνουμε το γέμισμα (υπολογισμό) ενός οποιουδήποτε *κουτί-στόχου* $M[n,k]$ (εδώ του $M[10,4]$) σε χρόνο $O(n \times k) << \left(\frac{n}{e}\right)^k$.

   Αυτό είναι τεράστιο όφελος.

4. **Ο Λαβύρινθος:** Η δυσκολία είναι να "οσμιστούμε" το δρόμο που θα διατρέξουμε τα κουτάκια γεμίζοντας τα διαδοχικά. Π.χ. στο 1ο πίνακα του Σχήματος (1.2) δεν είναι δυνατόν να γεμίσουμε πρώτα το κουτί $M[7,2]$.



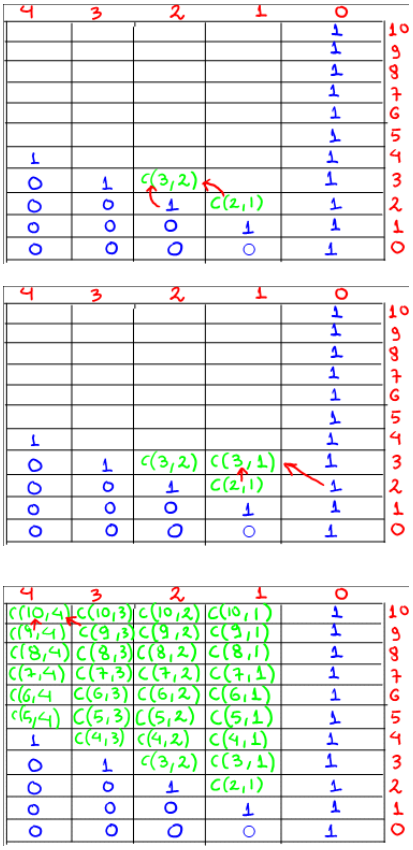
Σχήμα 1.2: Για τον υπολογισμό του $\binom{n}{k}$ απαιτείται η απομνημόνευση όλων των συντελεστών $\binom{i}{j}$ όπου το $i = 0, \ldots, n$ και το $j = 0, \ldots, k$. Απαιτούνται $(n+1) \times (k+1)$ διαφορετικές θέσεις μνήμης. Για παράδειγμα, στις 11×5 θέσεις του άνω πίνακα θα αποθηκεύσουμε τους απαιτούμενους συντελεστές για τον υπολογισμό του $\binom{10}{4}$.

5. **Ο Μίτος:** είναι η Σχέση (1.1)! Μας λέει ότι σε σταθερό χρόνο μπορούμε να γεμίσουμε το κουτί $M[i,j]$ που βρίσκεται στη γραμμή i προσθέτοντας τα 2, ήδη γεμάτα, κουτάκια $M[i-1,j], M[i-1,j-1]$ στην κάτω $(i-1)$-γραμμή τα οποία βρίσκονται στην ίδια j-στήλη και στην δεξιότερη $(j-1)$-στήλη. Εύκολα βρίσκουμε, δείτε 2ο, 3ο πίνακα στο Σχήμα (1.2), ότι η αρχή του κουβαριού είναι το κουτί $M[2,1]$ που σε σταθερό χρόνο γεμίζει με την πρόσθεση των αποθηκευμένων *ήδη* τιμών στα $M[1,1], M[1,0]$.

6. **Το ξετύλιγμα του Μίτου:** γίνεται στο Σχήμα (1.3), όπου στο 1ο πίνακα με το κόλπο της Σχέσης (1.1) γεμίζουμε σε σταθερό χρόνο το επόμενο κουτί $M[3,2]$. Όμοια, στο 2ο πίνακα και γεμίζουμε το κουτί $M[3,1]$, και συνεχίζουμε παρόμοια. Τέλος, στο 3ο πίνακα στο Σχήμα (1.3) γέμισαν όλα τα κουτιά και συνεπώς υπολογίσαμε το πάνω αριστερό κουτί-στόχος $M[10,4]$.

7. **Χρόνος αποθήκευσης σε πράσινα κουτάκια:** είναι ο χρόνος $O(nk)$ που απαιτεί ο απλός διπλός βρόγχος:

```
for(i = 2; i <= n; i++)
    for(j = 1; j <= min{i, k}; j++)
        M[i,j] = M[i-1,j] + M[i-1,j+1];
```



Σχήμα 1.3: Κάθε πράσινο κουτάκι γεμίζει σε $O(1)$ χρόνο με απλή πρόσθεση των αμέσως από κάτω του και του δεξιού σε αυτό πράσινων κουτιών.

Μια ίσως καλύτερη παρομοίωση είναι να φανταστούμε το γέμισμα των κουτιών του πίνακα M που παρουσιάσαμε στα Σχήματα (1.2) και (1.3) σαν τοποθέτηση τούβλων σε ένα τοίχο. Αρχικά πρέπει να μπουν τα

τούβλα των θεμελίων. Είναι αυτά που παίρνουν άμεσα τις βασικές τιμές 0 ή 1 από το μεσαίο και κάτω κλάδο της Σχέσης (1.1). Κάθε νέο τούβλο τοποθετείτε σε σταθερό χρόνο μόνο αν έχει γεμίσει όλη η από κάτω του σειρά από τούβλα (ακριβέστερα, τα 2 δεξιόστροφα τούβλα από κάτω του). Αν θέλω να βάλω, σε σταθερό χρόνο, το τούβλο $M[n,k] = C[n,k]$ πρέπει να χτίσω, σειρά-σειρά, όλα τα προηγούμενα nk τούβλα. Κάθε νέο τούβλο μου κοστίζει σταθερό χρόνο, έτσι, συνολικά πληρώνω χρόνο $O(nk)$.

1.2 Εξυπηρέτηση χρόνο-εξαρτώμενων εργασιών με βάρη (Weighted interval scheduling)

1.2.1 Περιγραφή προβλήματος

Στο πρόβλημα αυτό έχουμε n εργασίες $1,\ldots,n$ (πχ. αρχεία) και κάθε εργασία i μπορεί να εξυπηρετηθεί από ένα κοινό πόρο σε καθορισμένο χρονικό διάστημα $I_i = [s_i, f_i]$ (πχ. εκτυπωτής) αποδίδοντας μας καθορισμένη χρηματική αξία w_i. Οι εργασίες $i \neq j$ που επικαλύπτονται χρονικά $[s_i, f_i] \cap [s_j, f_j] \neq \emptyset$ λέγονται *ασύμβατες* και δεν μπορούν να εξυπηρετηθούν ταυτόχρονα.

Στόχος είναι η εύρεση του συνόλου $\mathcal{O}_n \subseteq \{1,\ldots,n\}$ αμοιβαία συμβατών εργασιών με τη *μέγιστη* συνολική αξία. Δηλαδή το σύνολο $\mathcal{O}_n$ συμβατών εργασιών επιτυγχάνει το μέγιστο άθροισμα $\sum_{i\in\mathcal{O}_n} w_i$ από οποιοδήποτε άλλο υποσύνολο S του $1,\ldots,n$.

Παρόλο που ένα οποιοδήποτε υποσύνολο S απαιτεί $\Omega(|S|)$ (γραμμικό χρόνο) για τον υπολογισμό του κέρδους $\sum_{i\in S} w_i$ που θα μας δώσει, η δυσκολία είναι ότι τα δυνατά υποσύνολα του $\{1,\ldots,n\}$ είναι 2^n. Άρα, ο εξαντλητικός υπολογισμός όλων των αθροισμάτων ανά υποσύνολο εργασιών θα δώσει $\Omega(2^n)$ χρόνο.

1.2.2 Κατασκευή της αναδρομής

Αν όλες οι εργασίες ήταν συμβατές τότε προφανώς η βέλτιστη λύση είναι να επιλέξουμε όλες τις εργασίες διότι έτσι λαμβάνουμε όλες τις αντίστοιχες αξίες και μεγιστοποιούμε το άθροισμα αξιών. Επειδή υπάρχουν ασύμβατες εργασίες, όταν επιλέγω μια εργασία j (επιθυμώντας να μου αποφέρει το κέρδος της w_j) αναγκάζομαι να απορρίψω όλες τις ασύμβατες με την j (χάνοντας όλες τις αντίστοιχες αξίες τους). Αυτό είναι το μεγάλο δίλημμα για κάθε μία εργασία $j = 1,\ldots,n$.

Έστω $\mathcal{O}_n$ η βέλτιστη λύση (υποσύνολο εργασιών) στο σύνολο εργασιών $\{1,\ldots,n\}$ και $\mathcal{O}(n)$ το αντίστοιχο βέλτιστο κέρδος που μας αποφέρει. Μελετούμε με τη βοήθεια ενός γέρο σοφού την επίδραση που έχουν στο βέλτιστο υποσύνολο εργασιών $\mathcal{O}_n$ οι δύο όψεις του άνω διλήμματος.

• Ο *σοφός* μας λέει ότι η εργασία $j \notin \mathcal{O}_n$. Το συμπέρασμα τότε είναι ότι η αξία w_j της εργασίας j δεν προστίθεται στην βέλτιστη λύση και επίσης είναι ότι η βέλτιστη λύση $\mathcal{O}_n$ μπορεί να αποτελείται από οποιεσδήποτε από τις υπόλοιπες εργασίες, εκτός της j, δηλαδή από τις $\{1,\ldots,n\} \setminus \{j\}$. Συνεπώς, η αξία $\mathcal{O}(n)$ του βέλτιστου υποσυνόλου εργασιών $\mathcal{O}_n$ στο σύνολο $\{1,\ldots,n\}$ μεγέθους n ισούται με την αξία $\mathcal{O}(n-1)$ του βέλτιστου υποσυνόλου εργασιών στο εναπομένον σύνολο $\{1,\ldots,n\} \setminus \{j\}$ μεγέθους $n-1$. Συμβολικά:

$$\text{βέλτιστη λύση στο } \{1,\ldots,n\} = \underbrace{\text{βέλτιστη λύση στο } \{1,\ldots,n\} \setminus \{j\}}_{\text{μέγεθος}=\, n-1}$$

Το όφελος της σοφής συμβουλής είναι *τεράστιο*, θυμηθείτε την Ενότητα 1.1.1 την παράγραφο ακριβώς κάτω από την Αναδρομή (1.1). Διότι από πρόβλημα με μέγεθος n, ακούγοντας τη σοφή συμβουλή, καταλήξαμε σε υπό-πρόβλημα με μέγεθος $n-1$.

• Ο σοφός μας λέει ότι $j \in \mathcal{O}_n$. Τότε η αξία $\mathcal{O}(n)$ του βέλτιστου υποσυνόλου εργασιών $\mathcal{O}_n$ στο σύνολο $\{1,\ldots,n\}$ θα αποτελείτε σίγουρα από την αξία w_j της εργασίας j. Επίσης, το βέλτιστο υποσύνολο εργασιών $\mathcal{O}_n$ θα περιέχει σίγουρα την εργασία j. Τέλος, το βέλτιστο υποσύνολο εργασιών $\mathcal{O}_n$ μπορεί να περιέχει εργασίες μόνο από το *εναπομένον* σύνολο από το αρχικό $\{1,\ldots,n\}$ εξαιρώντας τις *ασύμβατες* εργασίες με τη j. Άρα, η βέλτιστη λύση $\mathcal{O}_n$ στο σύνολο $\{1,\ldots,n\}$ έχει αξία ίση με την αξία w_j (διότι

τώρα $j \in \mathcal{O}_n$) συν την αξία της βέλτιστης λύσης στο υποσύνολο $\{1, \ldots, n\} \setminus \{$ασύμβατες εργασίες με τη $j\}$. Συμβολικά:

$$\text{βέλτιστη λύση στο } \{1, \ldots, n\} = \underbrace{w_j + \text{βέλτιστη λύση στο } \{1, \ldots, n\} \setminus \{\text{ασύμβατες εργασίες με τη } j\}}_{\text{μέγεθος} = \sharp\text{συμβατές με } j}$$

Και σε αυτή την εκδοχή το όφελος της σοφής συμβουλής είναι *τεράστιο*, διότι μειώσαμε το μέγεθος n του αρχικού προβλήματος, ακούγοντας την σοφή συμβουλή, σε μέγεθος $< n$ ίσο με το πλήθος των συμβατών εργασιών με την j.

Μπορούμε εύκολα να αποφύγουμε την εξάρτησή μας από τον σοφό γέρο με την παρακάτω λογική παρατήρηση.

• Από τις δυο δυνατότητες (συμβουλές) $j \in \mathcal{O}_n$ ή $j \notin \mathcal{O}_n$ η λογικά *βέλτιστη* για εμάς είναι η δυνατότητα που μας αποφέρει το μεγαλύτερο κέρδος, δηλαδή:

$$\begin{aligned}&(\text{βέλτιστη λύση στο } \{1, \ldots, n\}) = \\ &\max\{\underbrace{\text{βέλτιστη λύση στο } \{1, \ldots, n\} \setminus \{j\}}_{\text{μέγεθος} = n-1}, \underbrace{w_j + \text{βέλτιστη λύση στο } \{1, \ldots, n\} \setminus \{\text{ασύμβατες εργασίες με τη } j\}}_{\text{μέγεθος} = \sharp\text{συμβατές με } j}\} \quad (1.2)\end{aligned}$$

Σύγκριση αναδρομής (1.2) με αναδρομή (1.1). Στην αναδρομή (1.1) το πρόβλημα $C(n, k)$ μεγέθους n προέκυπτε από τη *πρόσθεση* των υποπροβλημάτων $C(n-1, k)$ και $C(n-1, k-1)$ μεγέθους $n-1$. Στην αναδρομή (1.2) το πρόβλημα

$$\text{βέλτιστη λύση στο } \{1, \ldots, n\}$$

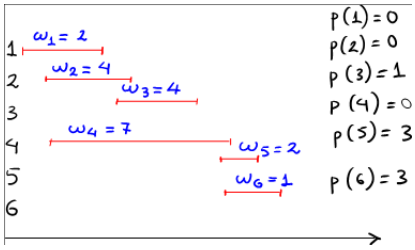
μεγέθους n προκύπτει από τη *μέγιστη* αξία που μπορούμε να λάβουμε από τα δύο υποπροβλήματα:

$$\text{βέλτιστη λύση στο } \{1, \ldots, n\} \setminus \{j\}$$

και

$$w_j + \text{βέλτιστη λύση στο } \{1, \ldots, n\} \setminus \{\text{ασύμβατες εργασίες με τη } \{j\}\}$$

Οι παραπάνω ομοιότητες μας φανερώνουν ότι είμαστε σε καλό δρόμο, υπό την έννοια ότι θα χρειαστεί να επιλύσουμε λίγα υποπροβλήματα με το δρόμο που θα μας δείξει η αναδρομική σχέση.



Σχήμα 1.4: Οι εργασίες έχουν ταξινομηθεί ως προς αύξουσα σειρά χρόνου ολοκλήρωσης. Παρατηρήστε τα αντίστοιχα $p(j)$.

Τώρα θα πρέπει να εξετάζουμε τις εργασίες με μια σειρά ώστε γρήγορα να βρίσκουμε τις εργασίες που είναι *συμβατές* με την τρέχουσα εργασία j που θα εξετάζουμε τις 2 δυνατότητες της (δηλαδή αν $j \in \mathcal{O}$ ή αν $j \notin \mathcal{O}$) στο τρέχον υποσύνολο εργασιών.

Ένας τρόπος (μπορούμε να θεωρήσουμε όποιον άλλο τρόπο θέλουμε) είναι να βάλουμε τις εργασίες σε αύξουσα σειρά ως προς τους χρόνους ολοκλήρωσης τους, δείτε τις εργασίες 1 ως 6 στο Σχήμα (1.4).

Δηλαδή θεωρούμε σειρά εργασιών $1, 2, \ldots, n$ ώστε να ισχύει $f_1 \leq f_2 \leq \ldots \leq f_n$. Θα εξετάζουμε τις 2 δυνατότητες (αν $\in \mathcal{O}$ ή αν $\notin \mathcal{O}$) διαδοχικά για κάθε εργασία αρχίζοντας από τη δεξιότερη εργασία n και θα προχωρούμε προς την αριστερότερη 1. Με αυτή την διάταξη είναι πολύ εύκολο να βρίσκουμε το σύνολο των συμβατών εργασιών μιας εργασίας j, διότι είναι όλες οι αριστερότερες εργασίες που τελειώνουν πριν αρχίσει η j. Έτσι, στο σύνολο εργασιών $\{1, \ldots, j\}$ για την εργασία j ορίζουμε ως $p(j)$ την αμέσως προηγούμενη συμβατή της εργασία (η τελευταία που θα μπορούσε να είναι συμβατή με τη j). Έτσι το $\{1, \ldots, p(j)\} \subseteq \{1, \ldots, j\}$ είναι το σύνολο των συμβατών εργασιών της j, δείτε Σχήμα (1.4).

Ορίζουμε ως $\mathcal{O}_j$ τη βέλτιστη λύση στις εργασίες $\{1, \ldots, j\}$ και ως $\mathcal{O}(j)$ το κέρδος που αποφέρει η βέλτιστη λύση. Ζητούμε τη βέλτιστη λύση $\mathcal{O}_n$ στις εργασίες $\{1, \ldots, n\}$ και ως $\mathcal{O}(n)$ το κέρδος που αποφέρει η βέλτιστη λύση. Σύμφωνα με τα προηγούμενα που οδήγησαν στην Σχέση (1.2), υπάρχουν οι εξής 2 δυνατότητες για το σύνολο εργασιών $\{1, \ldots, n\}$:

- Αν $n \notin \mathcal{O}_n$ τότε το βέλτιστο κέρδος $\mathcal{O}(n)$ στο $\{1, \ldots, n\}$ ισούται με το βέλτιστο κέρδος $\mathcal{O}(n-1)$ του συνόλου εκτός της εργασίας n: $\{1, \ldots, n\} \setminus \{n\} = \{1, \ldots, n-1\}$. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(n-1)$$

- Αν $n \in \mathcal{O}_n$ τότε το βέλτιστο κέρδος $\mathcal{O}(n)$ στο $\{1, \ldots, n\}$ ισούται με την αξία w_n που λάβαμε τοποθετώντας τη εργασία n στη λύση $\mathcal{O}_n$ συν το βέλτιστο κέρδος $\mathcal{O}(p(n))$ του συνόλου συμβατών εργασιών με τη n, δηλαδή το $\{1, \ldots, n\} \setminus \{p(n+1), \ldots, n\} = \{1, \ldots, p(n)\}$. Συμβολικά:

$$\mathcal{O}(n) = w_n + \mathcal{O}(p(n))$$

- Από τις δύο άνω περιπτώσεις πρέπει να επιλέξουμε αυτή που μεγιστοποιεί το κέρδος. Συμβολικά:

$$\begin{aligned} \mathcal{O}(n) &= \max\{\mathcal{O}(n-1), w_n + \mathcal{O}(p(n))\} \\ \mathcal{O}(0) &= 0 \end{aligned} \tag{1.3}$$

Στο Σχήμα (1.5) λύνουμε το παράδειγμα του Σχήματος (1.4) αναπτύσσοντας αναδρομικά την (1.3). Ανάλογα είχαμε εργαστεί στο Σχήμα (1.1) για να δείξουμε το μεγάλο χρονικό κόστος της αναδρομικής επίλυσης.

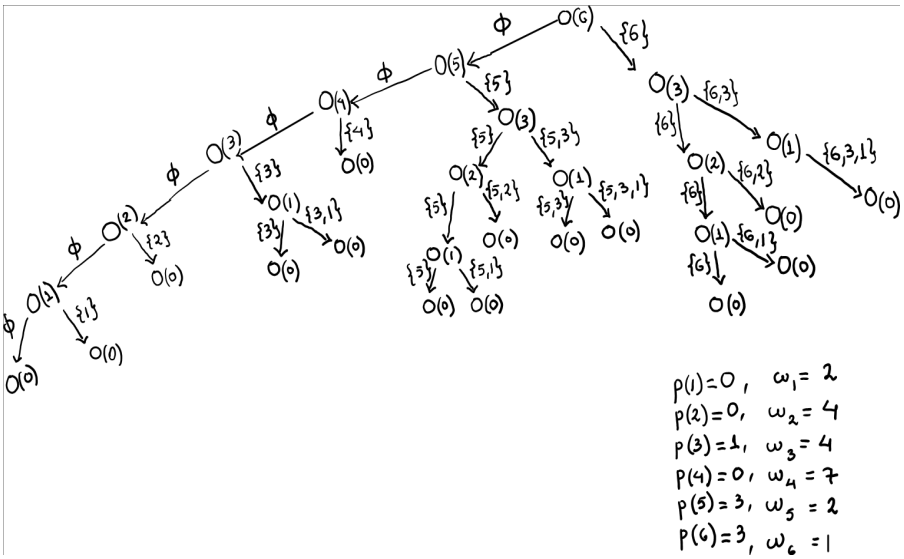
1.2.3 Απομνημόνευση της Αναδρομής (1.3)

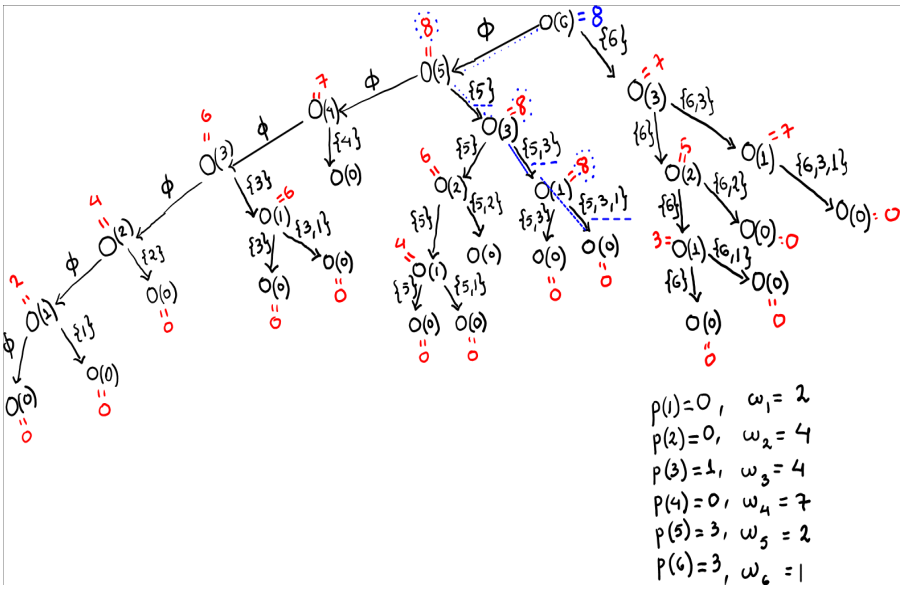
Θα εργαστούμε όπως ακριβώς στην Ενότητα 1.1.2. Θυμηθείτε πως στα Σχήματα (1.2) και (1.3) γεμίσαμε τα κουτάκια (τουβλάκια) με χρήση της Αναδρομής (1.1). Θα εργαστούμε με τον ίδιο τρόπο, όπως και σε *όλα* τα παραδείγματα που θα εξετάσουμε σε αυτές τις σημειώσεις. Τα πράγματα τώρα είναι πιο εύκολα, διότι τώρα η αποθήκευση των τιμών που προκύπτουν από την Αναδρομή (1.3) κτίζονται σε σειρά από τουβλάκια $M[0], \ldots, M[n]$ (ενώ πριν κτίζαμε "τοίχο" $M[i, j]$. Το εύκολο κουτάκι είναι για $i = 0$ το

$$M[0] = \mathcal{O}(0) = 0 \tag{1.4}$$

Αμέσως μπορώ να υπολογίσω το επόμενο κουτάκι με $i = 1$ με χρήση του απομνημονευμένου $M[0]$:

$$\begin{aligned} M[1] &= \max\{M[1-1], w_1 + M[\mathcal{O}(p(1))]\}, \text{ όπου } p(1) = 0 \\ &= \max\{M[0], w_1 + M[0]\}, \text{ όπου } w_1 = 2 \\ &= \max\{0, 2 + M[0]\}, \text{ αφού } 0 < 2 = w_1 \text{ θα επιλέξω την εργασία 1} \\ &= 2 \end{aligned} \tag{1.5}$$





Σχήμα 1.5: Στο άνω δέντρο, κάθε κόμβος $\mathcal{O}(j)$ είναι η βέλτιστη αξία για το υποσύνολο εργασιών $\{1, \ldots, j\}$ αναπτύσσοντας το υποδέντρο του σύμφωνα με την αναδρομή (1.3). Το (αριστερό) δεξιό παιδί του πατέρα $\mathcal{O}(j)$ προκύπτει αν (δεν) επιλέξουμε την εργασία j και τα 2 κλαδιά του αναγράφουν το υποσύνολο εργασιών που προκύπτει από την αντίστοιχη επιλογή κάθε φορά. Στο κάτω δέντρο, κάθε πατέρας αναγράφει με κόκκινο αριθμό την βέλτιστη αξία που λαμβάνει από το υποδέντρου με την (1.3). Οι υπολογισμοί αρχίζουν από τα φύλλα προς τον πατέρα όλου του δέντρου. Με μπλε αριθμό αναγράφεται η βέλτιστη αξία, το βέλτιστο υποσύνολό εργασιών και το βέλτιστο κλαδί από τον πατέρα-κορυφή προς ένα φύλλο. Το ασύμφορο με την αναδρομική επίλυση της (1.3) είναι ότι τα φύλλα μπορούν να γίνουν εκθετικά πολλά.

Πανεύκολα μπορώ να βρω και το επόμενο κουτάκι για $i = 2$ από τα απομνημονευμένα $M[1], M[0]$:

$$\begin{aligned} M[2] &= \max\{M[2-1], w_2 + M[\mathcal{O}(p(2))]\}, \text{ όπου } p(2) = 0 \\ &= \max\{M[1], w_2 + M[0]\}, \text{ όπου } w_2 = 4 \\ &= \max\{2, 4\}, \text{ αφού } 2 < 4 = w_2 \text{ θα επιλέξω την εργασία 2} \\ &= 4 \end{aligned} \tag{1.6}$$

Γενικεύοντας για $i = 3, \ldots, n$, καταλήγουμε στον παρακάτω γραμμικό αλγόριθμο, για γενικά n εργασίες, όπου με είσοδο την λίστα αριθμών $p(1), p(2), \ldots, p(n)$ σε χρόνο $O(n)$ μας επιστρέφει την βέλτιστη αξία $M[n]$:

$M[0] = 0;$
`for`$(i = 1; i <= n; i++)$
  $M[i] = \max\{M[i-1], w_i + M[p(i)]\};$

Ο παραπάνω αλγόριθμος για το παράδειγμα του Σχήματος (1.4) θα μας δώσει στην έξοδο τον πίνακα της βέλτιστης αξίας: $M = [0, 2, 4, 6, 7, 8, 8]$. Τέλος, εύκολα μπορούμε να χρησιμοποιήσουμε τον πίνακα βέλτιστης αξίας M που υπολογίζει ο άνω αλγόριθμος δυναμικού προγραμματισμού, ώστε ο παρακάτω βρόγχος να επιστρέφει το βέλτιστο σύνολο εργασιών $\mathcal{O}$:

$\mathcal{O} = \{\};$
`for`$(i = n; i >= 0; i--)\{$
  `if` $(M[i-1] \leq w_i + M[p(i)])\{\mathcal{O} = \mathcal{O} \cup \{i\}; i = p(i) + 1;\}\}$

Ο οποίος[1] για το παράδειγμα του Σχήματος (1.4) θα μας δώσει το πίνακα με το βέλτιστο σύνολο εργασιών $\mathcal{O} = [1, 3, 5]$.

1.3 Εργασίες δοσμένου συνολικού βάρους (Subset sum)

1.3.1 Περιγραφή προβλήματος

Φανταστείτε έναν κοινόχρηστο υπολογιστικό πόρο (εκτυπωτή) και ένα σύνολο εργασιών (αρχεία) $1, 2, \ldots, n$ που πρέπει να δρομολογήσουμε σε αυτόν. Κάθε μια εργασία j απαιτεί χρόνο εκτέλεσης w_j. Επειδή ο πόρος είναι κοινόχρηστος, μας επιτρέπουν να τον χρησιμοποιήσουμε μόνο στο καθορισμένο χρονικό διάστημα $[0, W]$. Στόχος μας είναι να λειτουργήσει ο πόρος, εκτελώντας τις εργασίες μας, κατά το δυνατόν περισσότερο στο επιτρεπόμενο διάστημα $[0, W]$.

Αυστηρά, έχουμε το σύνολο αντικειμένων $\{1, \ldots, n\}$ όπου κάθε ένα έχει βάρος w_i. Μας δίνουν ένα αριθμό W. Μας ζητάνε να βρούμε ένα υποσύνολο $S \subseteq \{1, \ldots, n\}$ με το μέγιστο βάρος $\sum_{i \in S} w_i$ υπό το περιορισμό το συνολικό βάρος του S να μην υπερβεί το W, δηλαδή $\sum_{i \in S} w_i \leq W$.

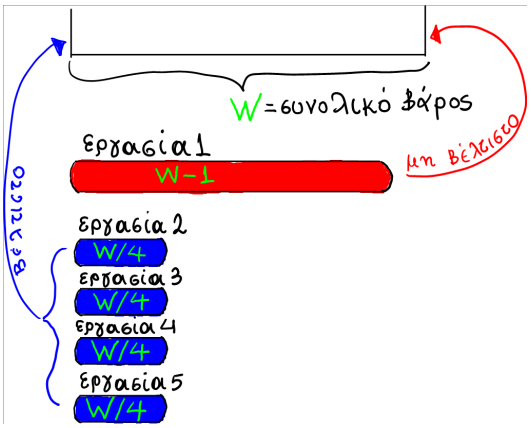
Το πρόβλημα αυτό μοιάζει με το αντίστοιχο της Ενότητας 1.2. Και τα δυο προβλήματα αφορούν δρομολόγηση εργασιών με βάρη (αξίες) με σκοπό την μεγιστοποίηση του συνολικού βάρος (αξίας). Το παρόν δεν απαιτεί συγκεκριμένη έναρξη και τέλος ανά εργασία, συνεπώς τώρα δεν υπάρχουν ασύμβατες εργασίες. Όμως το παρόν έχει τον περιορισμό ότι το συνολικό βάρος δεν πρέπει να υπερβεί το W.

[1] Κάθε φορά που ικανοποιείται η συνθήκη του `if`, η μεταβλητή i παίρνει τιμή $p(i) + 1$ και στην επόμενη επανάληψη του βρόγχου (επειδή έχουμε "$i--$") η i γίνεται $p(i)$. Πιο απλά, μόλις η εργασία i μπει στο βέλτιστο σύνολο $\mathcal{O}$, ο βρόγχος εξετάζει την πλησιέστερη συμβατή της από αριστερά εργασία $p(i)$.

1.3.2 Κατασκευή της Αναδρομής

Θα εργαστούμε όμοια με την Ενότητα 1.2.2. Βάζουμε σε μια σειρά της εργασίες, έστω $1, \ldots, n$. Καλούμε ως $\mathcal{O}_j$ το βέλτιστο υποσύνολο εργασιών που μπορούμε να βρούμε στο υποπρόβλημα με τη σειρά εργασιών $\{1, \ldots, j\}$. Καλούμε $\mathcal{O}(j)$ την αντίστοιχη βέλτιστη αξία του $\mathcal{O}_j$.

Αν όλες οι εργασίες $1, \ldots, n$ είχαν ίδιο βάρος τότε προφανώς το βέλτιστο σύνολο $\mathcal{O}_n$ θα σχηματιζόταν εύκολα αν επιλέγαμε όποια εργασία θέλαμε μέχρι να φτάσουμε κοντύτερα στο W χωρίς να το υπερβούμε. Όταν έχουμε διαφορετικά βάρη, το δίλημμα ανά εργασία j είναι ότι αν την επιλέξουμε το βάρος της w_j θα προστεθεί μέχρι να συμπληρώσουμε το βέλτιστο συνολικό βάρος W, μα επίσης, το εναπομένων επιτρεπτό βάρος μειώνεται από W σε $W - w_j$. Αυτό μπορεί να επηρεάσει θετικά ή αρνητικά το επιτρεπόμενο σύνολο μελλοντικών εργασιών που θα μπορέσουμε να επιλέξουμε ώστε να φτάσουμε κοντύτερα στο W χωρίς να το υπερβούμε, δείτε το Σχήμα 1.6. Αναλύουμε κάθε περίπτωση του διλήμματος $n \in \mathcal{O}_n$ ή $n \notin \mathcal{O}_n$, με τον



Σχήμα 1.6: Η αρχική επιλογή της εργασίας 1, αν και η πιο βαριά, αποβαίνει μοιραία για το βέλτιστο φόρτωμα του πόρου. Διότι αφήνει $W - w[1] = 1$ μονάδα υπολογιστικού φόρτου ανεκμετάλλευτη. Αντίθετα, η επιλογή των εργασιών 2, 3, 4, 5 επειδή $w[2] + \ldots + w[5] = W$ καταναλώνει όλο το αρχικό διαθέσιμο βάρος W. Παρατηρούμε ότι η σειρά που θα επιλέξουμε μια εργασία j, επειδή το βάρος της $w[j]$ επιδρά μειώνοντας κατά $w[j]$ το βάρος που απομένει στο εξής διαθέσιμο, είναι μεγάλης σημασίας για την κατασκευή της βέλτιστης λύσης.

τρόπο που στην Ενότητα 1.2.2 εξετάσαμε κάθε σοφή συμβουλή.

• Έστω ότι η εργασία $n \notin \mathcal{O}_n$. Τότε προφανώς το βάρος της w_n δεν θα συμπεριληφθεί στο άθροισμα που μας δίνει την βέλτιστη αξία $\mathcal{O}(n)$. Επίσης, το βέλτιστο υποσύνολο εργασιών $\mathcal{O}_n$ που θέλουμε να υπολογίσουμε στο σύνολο $\{1, \ldots, n\}$, αφού εξαιρέσαμε την εργασία n από τη βέλτιστη λύση, αρκεί να το υπολογίσουμε στο σύνολο $\{1, \ldots, n\} \setminus \{j\} = \{1, \ldots, n-1\}$. Δηλαδή η βέλτιστη λύση $\mathcal{O}_n$ με αξία $\mathcal{O}(n)$ ταυτίζεται με την βέλτιστη λύση $\mathcal{O}_{n-1}$ με αξία $\mathcal{O}(n-1)$. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(n-1) \tag{1.7}$$

• Έστω ότι η εργασία $n \in \mathcal{O}_n$. Τότε προφανώς το βάρος της w_n θα συμπεριληφθεί στο άθροισμα που μας δίνει την βέλτιστη αξία $\mathcal{O}(n)$. Επίσης, το βέλτιστο υποσύνολο εργασιών $\mathcal{O}_n$ που θέλουμε να υπολογίσουμε στο σύνολο $\{1, \ldots, n\}$ αρκεί να το υπολογίσουμε στο σύνολο $\{1, \ldots, n\} \setminus \{j\} = \{1, \ldots, n-1\}$ και να προσθέσουμε το βάρος w_n, αφού συμπεριλάβαμε την εργασία n στη βέλτιστη λύση. Δηλαδή η βέλτιστη λύση $\mathcal{O}_n$ με αξία $\mathcal{O}(n)$ ταυτίζεται με την βέλτιστη λύση $\mathcal{O}_{n-1}$ με αξία $\mathcal{O}(n-1)$ συν την αξία w_n. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(n-1) + w_n \tag{1.8}$$

```
> restart:
> W:= 18: n:= 5:
Initialization
-----------------------------
for w from 0 by 1 to W do
M[0,w]:=0:
end do:
for i from 0 by 1 to n do
M[i,0]:=0:
end do:
-----------------------------
w[1]:= 10: w[2]:= 5: w[3]:= 1: w[4]:= 15: w[5]:= 17:
>
>
> for i from 1 by 1 to n do
for w1 from 0 by 1 to W do

if(w[i] <= w1) then M[i,w1]:= max( M[i-1,w1], w[i]+M[ i-1, w1-w[i] ]  ):
else
M[i,w1]:= M[i-1,w1]:
end if:

end do:
end do:
The optimal value
M[n,W];
                                                    18
```

Σχήμα 1.7: Το πρόγραμμα maple της Ενότητας 1.3.3

Δυστυχώς, η σχέση (1.8) έχει ένα λεπτό λάθος. Τοποθετώντας την εργασία n στη βέλτιστη λύση $\mathcal{O}_n$ ξεχάσαμε να πληρώσουμε, ως τίμημα της προσθήκης του βάρους w_n, τη μείωση του συνολικού διαθέσιμου βάρους W κατά τη ποσότητα w_n. Δηλαδή το αρχικό διαθέσιμο βάρος W μειώθηκε στο $W - w_n$ (θυμηθείτε και το άνω Σχήμα 1.6). Αντίστοιχο τίμημα πληρώναμε και στη Ενότητα 1.2.2, όπου επιλέγοντας την εργασία j οφείλαμε να περιορίσουμε το αρχικό σύνολο εργασιών $\{1, \ldots, j\}$ στις απομένουσες συμβατές $\{1, \ldots, p(j)\}$ χάνοντας τις αξίες $w_{p(j)+1}, \ldots, w_j$ των ασύμβατων εργασιών με την εργασία j.

Καταλήγουμε ότι για να υπολογίσουμε την αξία $\mathcal{O}(j)$ της βέλτιστη λύσης $\mathcal{O}_j$ στο υποπρόβλημα εργασιών $\{1, \ldots, j\}$ είναι απαραίτητο να κρατάμε σε μια *ξεχωριστή* μεταβλητή w την τρέχουσα επιτρεπτή τιμή από το αρχικό W ώστε αν στο επόμενο βήμα επιλέξουμε μια εργασία j να μειώσουμε την τιμή της w κατά w_j. Ακριβέστερα, ορίζουμε ως:

$$\mathcal{O}(j, w) \tag{1.9}$$

την αξία του βέλτιστου υποσυνόλου εργασιών $\mathcal{O}_{j,w}$ στο $\{1, \ldots, j\}$ όπου το άθροισμα των βαρών του $\mathcal{O}_{j,w}$ δεν ξεπερνά τη τρέχουσα επιτρεπτή τιμή w.

Οπότε η Σχέση (1.7) γράφεται (διότι δεν επιλέξαμε την n άρα δεν λαμβάνουμε την αξία της w_n και δεν μειώνουμε κατά w_n το επιτρεπτό βάρος που παραμένει W):

$$\mathcal{O}(n, W) = \mathcal{O}(n-1, W) \tag{1.10}$$

και η Σχέση (1.8) γράφεται (διότι επιλέξαμε την n και λαμβάνουμε την αξία της w_n ενώ το διαθέσιμο επιτρεπτό βάρος μειώνεται κατά w_n και γίνεται $W - w_n$):

$$\mathcal{O}(n, W) = w_n + \mathcal{O}(n-1, W - w_n) \tag{1.11}$$

Επειδή θέλουμε να μεγιστοποιήσουμε το βάρος των επιλεγμένων εργασιών, οφείλουμε να επιλέξουμε την μέγιστη από τις δύο Σχέσεις (1.10) και (1.11), συνεπώς

$$\begin{aligned}
&\text{αν } w_n \leq W \\
&\quad \mathcal{O}(n, W) = \max\{\mathcal{O}(n-1, W), w_n + \mathcal{O}(n-1, W - w_n)\} \\
&\text{αν } w_n > W \\
&\quad \mathcal{O}(n, W) = \mathcal{O}(n-1, W)
\end{aligned} \tag{1.12}$$

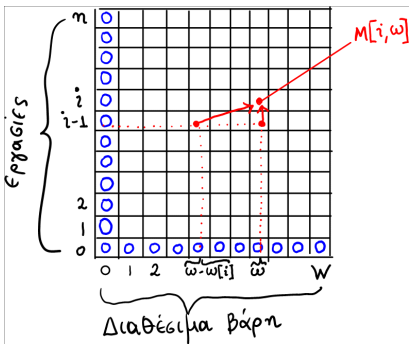
διότι αν το βάρος w_n της εργασίας n είναι μεγαλύτερο από το επιτρεπτό W δεν μπορεί να γίνει δεκτή η n, έτσι την απορρίπτουμε και επικεντρώνουμε στις υπόλοιπες $\{1, \ldots, n-1\}$ αφήνοντας το διαθέσιμο βάρος W αναλλοίωτο.

1.3.3 Απομνημόνευση της Αναδρομής (1.12)

Μας δίνουν τις εργασίες $1, \ldots, n$ και τα αντίστοιχα βάρη τους περιέχονται στη λίστα $w[1], \ldots, w[n]$. Θα απομνημονεύσουμε κάθε τιμή $\mathcal{O}(i, w)$ για $i = 0, \ldots, n$ και $w = 0, \ldots, W$ σε δυσδιάστατο πίνακα $M[i, w]$ με χρήση της αναδρομής (1.12). Αν $n = 0$ τότε δεν υπάρχουν εργασίες για επιλογή (π.χ. ενώ έχουμε σάκο χωρητικότητας W δυστυχώς είμαστε σε κατάστημα με άδεια ράφια) και άρα η αξία της βέλτιστης λύσης $\mathcal{O}(0, w) = 0$, για όλες τις επιτρεπτές τιμές του βάρους $w = 0, 1, 2, \ldots, W$, αρχικοποιείται από το παρακάτω βρόγχο στην τιμή 0.

```
for(w = 0; w <= W; w++)
    M[0, w] = 0;
```

Αν το διαθέσιμο βάρος είναι $w = 0$ τότε (π.χ. ενώ είμαστε σε κατάστημα με γεμάτα ράφια δυστυχώς έχουμε σάκο χωρητικότητας $w = 0$) δεν έχουμε δυνατότητα να επιλέξουμε κάποια εργασία από τις $\{1, \ldots, n\}$ διότι όλες έχουν βάρος > 0. Άρα η αξία της βέλτιστης λύσης $\mathcal{O}(i, 0) = 0$ για όλες τις εργασίες $i = 0, 1, 2, \ldots, n$ αρχικοποιείται από το παρακάτω βρόγχο στην τιμή 0.



Σχήμα 1.8: Παρατηρήστε πως το τουβλάκι $M[i, w]$ προκύπτει από τα $M[i-1, w]$ και $M[i-1, w-w[i]]$.

```
for(i = 0; i <= n; w++)
    M[i, 0] = 0;
```

Το τουβλάκι $M[i, w]$ θα χτιστεί πάνω από το αμέσως από κάτω του τουβλάκι $M[i-1, w]$ και επίσης θα ακουμπήσει λίγο στο αριστερό του τουβλάκι $M[i-1, w-w_i]$, δείτε Σχήμα (1.11). Αφού μπήκαν τα αρχικά τουβλάκια, ο παρακάτω αλγόριθμος χτίζει τον υπόλοιπο τοίχο, όπως ο αντίστοιχος στην Ενότητα (1.1.2).

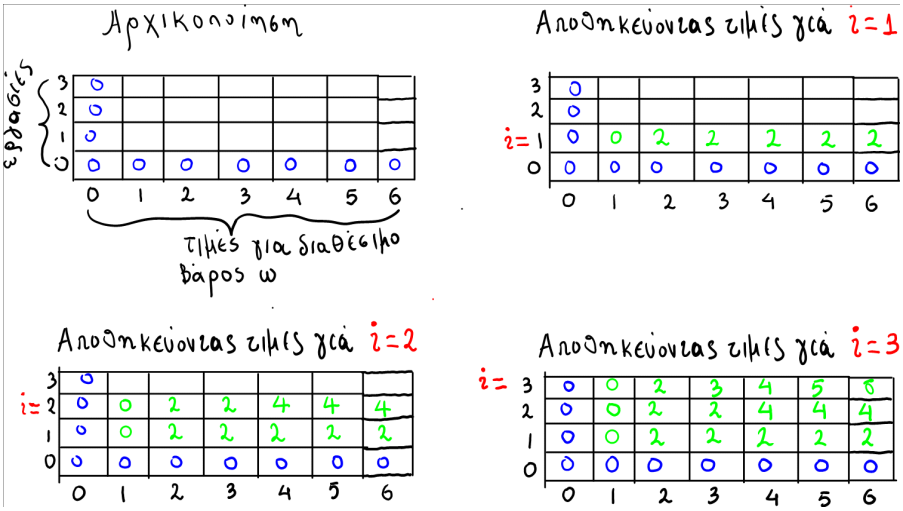
```
for(i = 1; i <= n; i++)
    {
    for(w = 0; w <= W; w++)
        {
        if(w[i] ≤ w) then
          {
          M[i, w] = max{M[i-1, w], w[i] + M[i-1, w-w[i]]};
        else
          M[i, w] = M[i-1, w];
          }
        }
    }
```

Το Σχήμα εμφανίζει την λειτουργία του άνω αλγορίθμου για ένα παράδειγμα δρομολόγησης εργασιών $1, 2, 3$ με βάρη $w[1] = 2, w[2] = 2, w[3] = 3$ όπου δεν πρέπει το συνολικό βάρος να υπερβεί το $W = 6$.

1.4 Ο σάκος του ληστή (Knapsack)

1.4.1 Περιγραφή του Προβλήματος

Το πρόβλημα αυτό έχει πολλές και, πέρα από τις προφανείς, σημαντικές εφαρμογές και εμφανίσεις σε συνθετότερα προβλήματα στην Επιστήμη της Πληροφορικής. Αποτελεί γενίκευση του προηγούμενου προβλήματος που περιγράψαμε στην Ενότητα 1.3.1, όπου στη παρούσα μορφή κάθε εργασία $j = 1, \ldots, n$ διαθέτει (πέρα από το βάρος της w_j και το μέγιστο επιτρεπτό βάρος W) και μια αξία v_j.



Σχήμα 1.9: Δρομολόγηση εργασιών $1, 2, 3$ με βάρη $w[1] = 2w[2] = 2, w[3] = 3$ όπου δεν πρέπει το συνολικό βάρος να υπερβεί το $W = 6$.

Η ονομασία του προβλήματος πηγάζει από την παρακάτω διαισθητική περιγραφή του. Ένας ληστής έχει ένα σάκο που μπορεί να χωρέσει αντικείμενα συνολικού βάρους W. Ο ληστής μπαίνει νύχτα σε ένα σπίτι με στόχο να βάλει στον σάκο αντικείμενα συνολικού βάρους $\leq W$ με την μέγιστη συνολική αξία.

1.4.2 Κατασκευή της Αναδρομής

Εργαζόμαστε όπως στην Ενότητα 1.3.2

• Αν το αντικείμενο $n \notin \mathcal{O}$ τότε επειδή απορρίψαμε την εργασία n από την μια δεν εισπράττουμε την αξία[2] της v_n και από την άλλη δεν μειώνουμε την χωρητικότητα του σάκου κατά w_n που παραμένει W:

$$\mathcal{O}(n, W) = \mathcal{O}(n-1, W) \tag{1.13}$$

• Αν το αντικείμενο $n \in \mathcal{O}$ επειδή επιλέξαμε την εργασία n από την μια εισπράττουμε την αξία[3] της v_n και από την άλλη το επιτρεπτό διαθέσιμο βάρος που μπορεί να χωρέσει πλέον στο σάκο μειώνεται σε $W - w_n$:

$$\mathcal{O}(n, W) = v_n + \mathcal{O}(n-1, W - w_n) \tag{1.14}$$

Επειδή θέλουμε να μεγιστοποιήσουμε το βάρος των επιλεγμένων εργασιών, οφείλουμε να επιλέξουμε την

[2]Εδώ η αξία της εργασίας j γενικά είναι $v_j \neq w_j$ και στόχος είναι η μεγιστοποίηση της συνολικής αξίας των επιλεγμένων εργασιών με συνολικό βάρος $\leq W$. Αυτή είναι η *μόνη* και σημαντικότατη διαφοροποίηση από την Ενότητα 1.4.2 όπου αξία της εργασίας j ήταν το βάρος της w_j και στόχος ήταν η μεγιστοποίηση του συνολικού βάρος επιλεγμένων εργασιών.

[3]Προσέξτε πως αυτό μας οδηγεί στη μοναδική διαφορά της Σχέσης (1.14) από την αντίστοιχη Σχέση (1.11).

μέγιστη από τις δύο Σχέσεις (1.13) και (1.14), συνεπώς

$$\begin{aligned} &\text{αν } w_n \leq W \\ &\quad \mathcal{O}(n, W) = \max\{\mathcal{O}(n-1, W), v_n + \mathcal{O}(n-1, W - w_n)\} \\ &\text{αν } w_n > W \\ &\quad \mathcal{O}(n, W) = \mathcal{O}(n-1, W) \end{aligned} \tag{1.15}$$

διότι αν το βάρος w_n της εργασίας n είναι μεγαλύτερο από το επιτρεπτό W δεν μπορεί να γίνει δεκτή η n, έτσι την απορρίπτουμε και επικεντρώνουμε στις υπόλοιπες $\{1, \ldots, n-1\}$ αφήνοντας το διαθέσιμο βάρος W αναλλοίωτο.

1.4.3 Απομνημόνευση της Αναδρομής

Η βέλτιστη αξία υπολογίζεται παρόμοια με τον αλγόριθμο της Ενότητας 1.3.3.
Αν $n = 0$ τότε δεν υπάρχουν αντικείμενα αξίας για να τα βάλει ο κλέφτης στο σάκο. Άρα η αξία της βέλτιστης λύσης $\mathcal{O}(0, w) = 0$, για όλες τις επιτρεπτές τιμές του βάρους $w = 0, 1, 2, \ldots, W$, αρχικοποιείται από το παρακάτω βρόγχο στην τιμή 0.

```
for(w = 0; w <= W; w++)
    M[0, w] = 0;
```

Αν η διαθέσιμη χωρητικότητα του σάκου είναι $w = 0$, τότε ο κλέφτης δεν μπορεί να πάρει κάποιο αντικείμενο αξίας από τα $\{1, \ldots, n\}$, διότι κάθε ένα αντικείμενο απαιτεί χώρο > 0. Άρα η αξία της βέλτιστης λύσης $\mathcal{O}(i, 0) = 0$ για όλες τις εργασίες $i = 0, 1, 2, \ldots, n$ αρχικοποιείται από το παρακάτω βρόγχο στην τιμή 0.

```
for(i = 0; i <= n; w++)
    M[i, 0] = 0;
```

Υπολογίζουμε τις υπόλοιπες τιμές με την Αναδρομή (1.15):

```
for(i = 1; i <= n; i++)
    {
    for(w = 0; w <= W; w++)
        {
        if(w[i] ≤ w) then
            {
            M[i, w] = max{M[i − 1, w], v_i + M[i − 1, w − w[i]]};
        else
            M[i, w] = M[i − 1, w];
            }
        }
    }
```

1.5 Μέγιστη αύξουσα υπακολουθία (Longest increasing subsequence)

1.5.1 Περιγραφή του Προβλήματος

Έστω η ακολουθία αριθμών

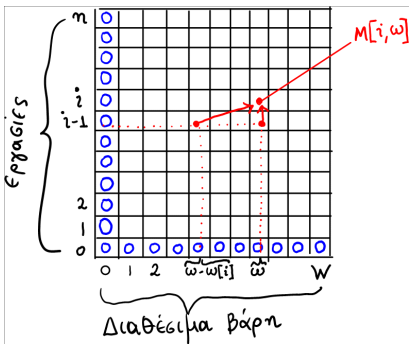
$$A = [a_1, a_2, \ldots, a_n] \tag{1.16}$$

```
> restart:
> W:= 18: n:= 5:
Initialization
----------------------------------------
for w from 0 by 1 to W do
M[0,w]:=0:
end do:
for i from 0 by 1 to n do
M[i,0]:=0:
end do:
-------------------------------------------
The values are:
v[1]:= 4: v[2]:= 2: v[3]:= 10: v[4]:= 3: v[5]:= 100:
The weights are
w[1]:= 10: w[2]:= 5: w[3]:= 1: w[4]:= 15: w[5]:= 17:
>
>
> for i from 1 by 1 to n do
for w1 from 0 by 1 to W do

if(w[i] <= w1) then M[i,w1]:= max( M[i-1,w1], v[i]+M[ i-1, w1-w[i] ]  ):
else
M[i,w1]:= M[i-1,w1]:
end if:

end do:
end do:
The optimal value is:
M[n,W];
                                        110
```

Σχήμα 1.10: Το πρόγραμμα maple της Ενότητας 1.4.3



Σχήμα 1.11: Παρατηρήστε πως το τουβλάκι $M[i, w]$ προκύπτει από τα $M[i-1, w]$ και $M[i-1, w-w[i]]$.

Μια υπακολουθία B της A είναι ένα υποσύνολο όρων της που εμφανίζονται με την ίδια σειρά όπως στην A. Όπου μήκος είναι το πλήθος των όρων της υπακολουθίας. Ζητούμε να υπολογίσουμε το μήκος της μεγαλύτερης αύξουσας υπακολουθίας της A. Έστω $\mathcal{O}(n)$ το μήκος της βέλτιστης αύξουσας υπακολουθίας $\mathcal{O}_n$ με όρους της A που εμφανίζονται στις θέσεις $\{1, \ldots, n\}$.

Για παράδειγμα, αν $A = [5, 2, 8, 6, 3, 6, 7, 9]$ τότε μια υπακολουθία της A είναι η $[5, 2, 6, 3, 7]$, ενώ η $[2, 5, 6, 7, 9]$ δεν είναι υπακολουθία της A. Τέλος, $\mathcal{O}(8) = 4$ είναι το μήκος μιας βέλτιστης αύξουσας υπακολουθίας $\mathcal{O}_8 = \{2, 3, 6, 9\}$ με όρους της A.

1.5.2 Κατασκευή της Αναδρομής

Κάθε στοιχείο της A είτε θα ανήκει είτε δεν θα ανήκει στην βέλτιστη αύξουσα υπακολουθία $\mathcal{O}_n$. Εξετάζουμε τις 2 δυνατότητες με την βοήθεια ενός σοφού.

• Αν ο αριθμός a_n στην θέση n δεν ανήκει στην βέλτιστη υπακολουθία μεταξύ των θέσεων $\{1, \ldots, n\}$, δηλαδή $a_n \notin \mathcal{O}_n$, τότε η 1 θέση του δεν θα προστεθεί στο βέλτιστο μήκος $\mathcal{O}(n)$. Άρα, η βέλτιστη υπακολουθία στις θέσεις $\{1, \ldots, n\}$ έχει μήκος όσο η βέλτιστη υπακολουθία στις θέσεις $\{1, \ldots, n-1\}$. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(n-1) \tag{1.17}$$

• (Λάθος τρόπος.) Αν ο αριθμός a_n στην θέση n ανήκει στην βέλτιστη υπακολουθία μεταξύ των θέσεων $\{1, \ldots, n\}$, δηλαδή $a_n \in \mathcal{O}_n$, τότε η 1 θέση του θα προστεθεί στο βέλτιστο μήκος $\mathcal{O}(n)$. Άρα, η βέλτιστη υπακολουθία στις θέσεις $\{1, \ldots, n\}$ έχει μήκος όσο η βέλτιστη υπακολουθία στις θέσεις $\{1, \ldots, n-1\}$ συν 1. Συμβολικά:

$$\mathcal{O}(n) = 1 + \mathcal{O}(n-1) \tag{1.18}$$

• (Σωστός τρόπος.) Αν ο αριθμός a_n ανήκει στην βέλτιστη αύξουσα υπακολουθία $\mathcal{O}(n)$ τότε αριστερά του στην $\mathcal{O}(n)$ πρέπει να είναι αριθμοί $\leq a_n$, αλλιώς η υπακολουθία $\mathcal{O}(n)$ δεν θα ήταν αύξουσα. Άρα από τις δυνατές αριστερές θέσεις $\{1, \ldots, n-1\}$ πρέπει να εξαιρέσουμε τις θέσεις από την $n-1$ μέχρι την 1η θέση $p(n)$ *αριστερά* της n που συναντούμε τον 1ο αριθμό $\leq a_n$. Με άλλα λόγια, επιλέγοντας τον a_n πρέπει να εξαιρέσουμε όλους τους *ασύμβατους*[4] όρους $\{a_{p(i)+1}, \ldots, a_{n-1}\}$ αμέσως αριστερά του a_n διαδοχικούς

[4] Ο τρόπος εργασίας είναι όμοιος με την Ενότητα 1.2.2 όπου εκεί ορίζαμε $p(i)$ την τελευταία αριστερή εργασία που ήταν συμβατή με την εργασία i, εξαιρώντας όλες τις ασύμβατες $\{p(i)+1, \ldots, i-1\}$ αριστερά από την i.

αριθμούς που είναι $> a_n$. Συμβολικά:

$$\mathcal{O}(n) = 1 + \mathcal{O}(p(n)) \tag{1.19}$$

Επειδή θέλουμε να μεγιστοποιήσουμε το βάρος των επιλεγμένων εργασιών, οφείλουμε να επιλέξουμε την μέγιστη από τις δύο Σχέσεις (1.17) και (1.19), συνεπώς

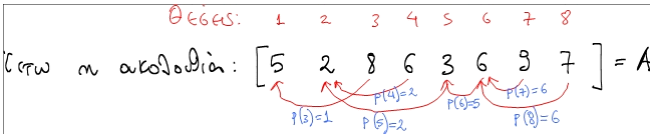
$$\mathcal{O}(n) \quad = \quad \max\{\mathcal{O}(n-1), 1 + \mathcal{O}(p(n))\} \tag{1.20}$$

1.5.3 Απομνημόνευση της Αναδρομής

Η βέλτιστη αξία υπολογίζεται με τον αλγόριθμο:

$M[0] = 0;$
for$(i = 1; i <= n; i++)$
  $M[i] = \max\{M[i-1], 1 + M[p[i]]\};$
$M[n];$

Παράδειγμα



Σχήμα 1.12: Έχουμε $p(8) = 6, p(7) = 6, p(6) = 5, p(5) = 2, p(4) = 2, p(3) = 1, p(2) = 0, p(1) = 0$.

$$\begin{aligned}
\mathcal{O}(1) &= \max\{\mathcal{O}(0), 1 + \mathcal{O}(p(1))\} = \max\{0, 1 + \mathcal{O}(0)\} = \max\{0, 1 + 0\} = 1 \\
\mathcal{O}(2) &= \max\{\mathcal{O}(2-1), 1 + \mathcal{O}(p(2))\} = \max\{1, 1 + \mathcal{O}(0)\} = \max\{1, 1 + 0\} = 1 \\
\mathcal{O}(3) &= \max\{\mathcal{O}(3-1), 1 + \mathcal{O}(p(3))\} = \max\{1, 1 + \mathcal{O}(2)\} = \max\{1, 1 + 2\} = 3 \\
\mathcal{O}(4) &= \max\{\mathcal{O}(4-1), 1 + \mathcal{O}(p(4))\} = \max\{3, 1 + \mathcal{O}(2)\} = \max\{3, 1 + 1\} = 3 \\
\mathcal{O}(5) &= \max\{\mathcal{O}(5-1), 1 + \mathcal{O}(p(5))\} = \max\{3, 1 + \mathcal{O}(2)\} = \max\{3, 1 + 1\} = 3 \\
\mathcal{O}(6) &= \max\{\mathcal{O}(6-1), 1 + \mathcal{O}(p(6))\} = \max\{3, 1 + \mathcal{O}(5)\} = \max\{3, 1 + 2\} = 3 \\
\mathcal{O}(7) &= \max\{\mathcal{O}(7-1), 1 + \mathcal{O}(p(7))\} = \max\{3, 1 + \mathcal{O}(6)\} = \max\{3, 1 + 3\} = 4 \\
\mathcal{O}(8) &= \max\{\mathcal{O}(8-1), 1 + \mathcal{O}(p(8))\} = \max\{4, 1 + \mathcal{O}(6)\} = \max\{3, 1 + 3\} = 4
\end{aligned} \tag{1.21}$$

1.6 Ανταγωνιστική τοποθέτηση εταιριών σε Εθνική Οδό (Competing firms)

1.6.1 Περιγραφή του Προβλήματος

Κάθε εταιρία $i \in \{1, \ldots, n\}$ ανταγωνίζεται για αγορά τοποθεσίας στο m_i-οστό km (είναι δυνατόν > 1 εταιρίες να επιθυμούν την ίδια τοποθεσία) της Εθνικής οδού Αθήνα-Πάτρα, πληρώνοντας ποσό x_i για την αντίστοιχη τοποθεσία. Το Υπουργείο που εκμεταλλεύεται την Οδό επιθυμεί να επιλέξει το βέλτιστο υποσύνολο εταιριών $\mathcal{O}_n \subseteq \{1, \ldots, n\}$ ώστε να μεγιστοποιήσει το κέρδος $\mathcal{O}(n)$ με τους εξής περιορισμούς

1. Μόνο 1 εταιρία μπορεί να εγκατασταθεί σε δοσμένη τοποθεσία.
2. Η ελάχιστη απόσταση μεταξύ διαδοχικών εταιριών να είναι > 20 km.

Για την λύση, μας βολεύει να διατάξουμε τις επιθυμητές τοποθεσίες ως προς αύξουσα σειρά, δηλαδή $m_1 \leq m_2 \leq \ldots \leq m_n$. Έστω $\mathcal{O}(n)$ η αξία της βέλτιστης επιλογής εταιριών $\mathcal{O}_n$ που εμφανίζονται στις θέσεις $\{m_1, \ldots, m_n\}$.

1.6.2 Κατασκευή της Αναδρομής

Κάθε εταιρία i είτε θα ανήκει είτε δεν θα ανήκει στην βέλτιστη συλλογή εταιριών $\mathcal{O}_n$. Εξετάζουμε τις 2 δυνατότητες με την βοήθεια ενός σοφού.
• Αν η εταιρία n στην θέση m_n δεν ανήκει στην βέλτιστη συλλογή εταιριών μεταξύ των θέσεων $\{m_1, \ldots, m_n\}$, δηλαδή $n \notin \mathcal{O}_n$, τότε το ποσό x_n δεν θα προστεθεί στο βέλτιστο ποσό $\mathcal{O}(n)$. Άρα, η βέλτιστη συλλογή εταιριών στις θέσεις $\{m_1, \ldots, m_n\}$ αποφέρει κέρδος όσο η βέλτιστη συλλογή εταιριών στις θέσεις $\{m_1, \ldots, m_{n-1}\}$. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(n-1) \tag{1.22}$$

• (Σωστός τρόπος.) Αν η εταιρία n στην θέση m_n ανήκει στην βέλτιστη συλλογή εταιριών μεταξύ των θέσεων $\{m_1, \ldots, m_n\}$, δηλαδή $n \in \mathcal{O}_n$, τότε το ποσό x_n θα προστεθεί στο βέλτιστο ποσό $\mathcal{O}(n)$. Επίσης, αριστερά της εταιρίας n στην τοποθεσία m_n θα πρέπει να είναι συμβατές[5] εταιρίες, δηλαδή με απόσταση ≥ 20 km. Άρα από τις δυνατές αριστερές θέσεις $\{m_1, \ldots, m_{n-1}\}$ πρέπει να εξαιρέσουμε τις θέσεις από την m_{n-1} μέχρι πριν την 1η θέση $m_{p(n)}$ *αριστερά* της m_n που συναντούμε το 1ο εστιατόριο $p(n)$ ώστε $m_{p(n)} + 20 \leq m_n$. Με άλλα λόγια, επιλέγοντας την εταιρία i πρέπει να εξαιρέσουμε όλες τις $\{m_{p(i)+1}, \ldots, a_i\}$ αμέσως αριστερά διαδοχικές τοποθεσίες που απέχουν < 20 km από την τοποθεσία m_i. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(p(n)) + x_n \tag{1.23}$$

• (Λάθος τρόπος.) Σε αυτό το σημείο, ίσως είναι πιο συμφέρον αντί να επιλέξουμε την εταιρία n, να επιλέξουμε την πιο συμφέρουσα μεταξύ της n και όλες τις αριστερά ασύμβατες της $\{p(n)+1, \ldots, n\}$ που απέχουν < 20 km από την τοποθεσία m_n. Είναι φανερό ότι αυτό συμφέρει πιο πολύ, διότι: $x_n \leq x_{k_0} = \max_{p(n)+1 \leq k \leq n}\{x_k\}$. Επιπλέον, αφού από *όλες* τις ασύμβατες εταιρίες $\{p(n)+1, \ldots, n\}$ για την θέση m_n θα επιλέξουμε *μόνο μία* (την πιο συμφέρουσα εταιρία k_0 μεταξύ αυτών), γίνεται φανερό ότι *δεν* θα παραβιάσουμε το κανονισμό για την θέση m_n. Συμβολικά:

$$\mathcal{O}(n) = \mathcal{O}(p(n)) + \max_{p(n)+1 \leq k \leq n}\{x_k\} \tag{1.24}$$

Παρατηρούμε ότι το κέρδος της αναδρομής (1.24) είναι μεγαλύτερο από το αντίστοιχο της (1.23). Δυστυχώς, ο συλλογισμός είναι λάθος. Το λάθος είναι ότι η εταιρία k_0 παρόλο που δεν δημιουργεί ασυμβατότητα μεταξύ των $\{p(n)+1, \ldots, n\}$ μπορεί να δημιουργήσει ασυμβατότητα με τις αριστερότερες εργασίες στο υποπρόβλημα $\{1, \ldots, p(n)\}$ που απομένει μετά την επιλογή της k_0. Διότι, αν και από ορισμό του $p(n)$ έχουμε $\forall j \in \{1, \ldots, p(n)\}$ ότι η εταιρία j είναι συμβατή με n μπορεί να $\exists j_0 \in \{1, \ldots, p(n)\}$ ώστε εταιρία j_0 ασύμβατη με την επιλεγμένη k_0. Αυτό το λεπτό σημείο γίνεται πιο κατανοητό στο Παράδειγμα της πιο κάτω Ενότητας 1.6.3.

Καταλήγουμε ότι επειδή θέλουμε να μεγιστοποιήσουμε το βάρος των επιλεγμένων εργασιών, οφείλουμε να επιλέξουμε την μέγιστη από τις δύο Σχέσεις (1.22) και (1.23), συνεπώς

$$\mathcal{O}(n) \quad = \quad \max\{\mathcal{O}(n-1), x_n + \mathcal{O}(p(n))\} \tag{1.25}$$

[5] Ο τρόπος εργασίας είναι όμοιος με την Ενότητα 1.2.2 όπου εκεί ορίζαμε $p(i)$ την τελευταία αριστερή εργασία που ήταν συμβατή με την εργασία i, εξαιρώντας όλες τις ασύμβατες $\{p(i)+1, \ldots, i-1\}$ αριστερά από την i. Επίσης όμοια εργαστήκαμε στο αμέσως προηγούμενο πρόβλημα.

1.6.3 Απομνημόνευση της Αναδρομής

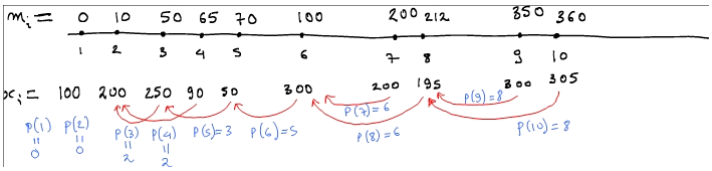
Η προεργασία υπολογισμού των $p[i]$'ς γίνεται με τον αλγόριθμο γραμμικού (γιατί;) χρόνου:

```
p[1] = 0;
for(i = 1; i <= n; i++)
    {
      j = p[i] + 1;
      while (m[j] + 20 < m[i + 1])j++;
      p[i + 1] = j - 1;
    }
```

και βέλτιστη αξία υπολογίζεται με τον αλγόριθμο:

```
M[0] = 0;
for(i = 1; i <= n; i++)
    M[i] = max{M[i - 1], x[i] + M[p[i]]};
M[n];
```

Παράδειγμα



Σχήμα 1.13: Έχουμε $p(10) = 8, p(9) = 8, p(8) = 6, p(7) = 6, p(6) = 5, p(5) = 3, p(4) = 2, p(3) = 2, p(2) = 0, p(1) = 0$.

$$
\begin{aligned}
\mathcal{O}(1) &= \max\{\mathcal{O}(0), x_1 + \mathcal{O}(p(1))\} = \max\{0, 100 + \mathcal{O}(0)\} = 1\max\{0, 100 + 0\} = 100 \\
\mathcal{O}(2) &= \max\{\mathcal{O}(2-1), x_2 + \mathcal{O}(p(2))\} = \max\{100, 200 + \mathcal{O}(0)\} = 200 \\
\mathcal{O}(3) &= \max\{\mathcal{O}(3-1), x_3 + \mathcal{O}(p(3))\} = \max\{200, 250 + \mathcal{O}(2)\} = \max\{200, 450\} = 450 \\
\mathcal{O}(4) &= \max\{\mathcal{O}(4-1), x_4 + \mathcal{O}(p(4))\} = \max\{450, 90 + \mathcal{O}(2)\} = \max\{450, 290\} = 450 \\
\mathcal{O}(5) &= \max\{\mathcal{O}(5-1), x_5 + \mathcal{O}(p(5))\} = \max\{450, 50 + \mathcal{O}(3)\} = \max\{450, 500\} = 500 \\
\mathcal{O}(6) &= \max\{\mathcal{O}(6-1), x_6 + \mathcal{O}(p(6))\} = \max\{450, 300 + 500\} = 800 \\
\mathcal{O}(7) &= \max\{\mathcal{O}(7-1), x_7 + \mathcal{O}(p(7))\} = \max\{750, 200 + \mathcal{O}(6)\} = \max\{750, 200 + 800\} = 1000 \\
\mathcal{O}(8) &= \max\{\mathcal{O}(8-1), x_8 + \mathcal{O}(p(8))\} = \max\{1000, 195 + \mathcal{O}(6)\} = \max\{1000, 195 + 800\} = 1000 \\
\mathcal{O}(9) &= \max\{\mathcal{O}(9-1), x_9 + \mathcal{O}(p(9))\} = \max\{1000, 300 + \mathcal{O}(8)\} = \max\{1000, 300 + 1000\} = 1300 \\
\mathcal{O}(10) &= \max\{\mathcal{O}(10-1), x_{10} + \mathcal{O}(p(10))\} = \max\{1300, 305 + \mathcal{O}(8)\} = \max\{1300, 305 + 1000\} = 1305
\end{aligned}
\tag{1.26}
$$

άρα η βέλτιστη αξία με την αναδρομική σχέση που προκύπτει από τη μέγιστη από τις *σωστές* σχέσεις (1.22) και (1.23) είναι 1305. Επίσης, μια βέλτιστη επιλογή εστιατορίων είναι η $\mathcal{O}_{10} = \{10, 7, 6, 5, 3, 2\}$ και μπορεί να υπολογιστεί "περπατώντας προς τα πίσω" τα αναδρομικά βήματα $\mathcal{O}(10), \mathcal{O}(9), \ldots, \mathcal{O}(1)$ στην (1.26). Πιο αναλυτικά,

1. Στο $\mathcal{O}_{10}$ επέλεξε τη εταιρία 10 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(10)\} = \{1,\ldots,8\}$.
2. Στο $\mathcal{O}_8$ ΔΕΝ επέλεξε τη εταιρία 8 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,8-1\} = \{1,\ldots,7\}$.
3. Στο $\mathcal{O}_7$ επέλεξε τη εταιρία 7 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(7)\} = \{1,\ldots,6\}$.
4. Στο $\mathcal{O}_6$ επέλεξε τη εταιρία 6 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(6)\} = \{1,\ldots,5\}$.
5. Στο $\mathcal{O}_5$ επέλεξε τη εταιρία 5 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(5)\} = \{1,\ldots,3\}$.
6. Στο $\mathcal{O}_3$ επέλεξε τη εταιρία 3 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(3)\} = \{1,\ldots,2\}$.
7. Στο $\mathcal{O}_2$ επέλεξε τη εταιρία 2 και στην συνέχεια είναι επιτρεπτό να επιλέξει από τις $\{1,\ldots,p(2)\} = \{1,\ldots,0\} = \{\emptyset\}$.

Για να δούμε όμως τι συμβαίνει αν χρησιμοποιήσουμε την (1.24) αντί της (1.23).

$$\begin{aligned}
\mathcal{O}(1) &= x_1 = 100 \\
\mathcal{O}(2) &= \max\{\mathcal{O}(2-1), x_2\} = \max\{100, 200 + \mathcal{O}(0)\} = 200 \\
\mathcal{O}(3) &= \mathcal{O}(p(3)) + \max_{p(3)+1\le k\le 3}\{x_k\} = \mathcal{O}(2) + \max_{3\le k\le 3}\{x_k\} = 200 + 250 = 450 \\
\mathcal{O}(4) &= \mathcal{O}(p(4)) + \max_{p(4)+1\le k\le 4}\{x_k\} = \mathcal{O}(2) + \max_{3\le k\le 4}\{x_k\} = 200 + \max\{250, 90\} = 450 \\
\mathcal{O}(5) &= \mathcal{O}(p(5)) + \max_{p(5)+1\le k\le 5}\{x_k\} = \mathcal{O}(3) + \max_{4\le k\le 5}\{x_k\} = 450 + \max\{90, 50\} = 540 \qquad (1.27)
\end{aligned}$$

Παρατηρούμε τώρα ότι $\mathcal{O}(5) = 540$ είναι > 450 για το αντίστοιχο $\mathcal{O}(5)$ στην (1.26). Αυτό συμβαίνει διότι τώρα ο αλγόριθμος επέλεξε ασύμβατες τοποθεσίες $\mathcal{O}_4 = \{4, 3, 2\}$. Συνεχίζοντας με όμοιο τρόπο καταλήγουμε ότι η βέλτιστη αξία είναι η 1345, μεγαλύτερη από την λύση 1305 στην (1.26), αλλά δυστυχώς, επιλέχθηκαν οι μη συμβατές τοποθεσίες 3 και 4.

1.7 Μέθοδος Ελαχίστων Τετραγώνων σε τομείς (Segmented Least Squares)

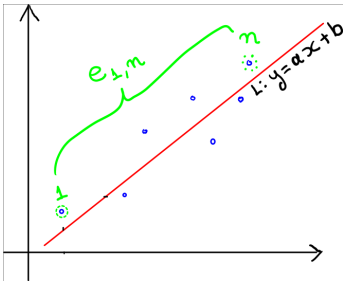
Σε όλα τα προβλήματα που εξετάσαμε μέχρι τώρα η αναδρομή κατασκευαζόταν αναλύοντας τις *δύο* εκδοχές ενός διλήμματος ($n \in \mathcal{O}$ ή $n \notin \mathcal{O}$) και επιλέγοντας την βέλτιστη από αυτές. Στο παρόν πρόβλημα η αναδρομή θα βασίζεται στην ανάλυση *πολυωνυμικού* πλήθους εκδοχών και την επιλογή της βέλτιστης από αυτές.

Το πρόβλημα που θα εξετάσουμε έχει τεράστια σημασία στην Στατιστική Ανάλυση τυχαίων δεδομένων, στη συμπίεση δεδομένων, στην Αριθμητική Ανάλυση κ.τ.λ.

1.7.1 Περιγραφή του Προβλήματος

Προσέγγιση καμπύλης με 1 ευθεία. Συχνά τα δεδομένα που παράγει ένα φαινόμενο που μελετάμε έχουν μορφή καμπύλης $\mathcal{C}$ με n σημεία στο επίπεδο. Οι καμπύλες, παρά την ομορφιά τους, είναι –ιδίως υπολογιστικά– δύσχρηστα αντικείμενα. Αν προσεγγίζαμε την καμπύλη $\mathcal{C}$ με μια ευθεία γραμμή

$$L : y = \alpha x + \beta \qquad (1.28)$$



Σχήμα 1.14: Τα σημεία $1, \ldots, n$ προσεγγίζονται με λάθος $e_{1,n}$ από την ευθεία $L : y = \alpha x + \beta$.

τότε με γνώση *μόνο* των σταθερών συντελεστών α, β της L μπορούμε να περιγράψουμε *οποιοδήποτε* σημείο (στιγμιότυπο) της καμπύλης που διέπει το φαινόμενο που μελετάμε. Αντιλαμβανόμαστε την σπουδαιότητα μιας τέτοιας προσέγγισης που καταφέρνει να *συμπιέσει* στους αριθμούς α, β ένα αρκετά πιο σύνθετο φαινόμενο.

Αυτό το πρόβλημα έχει λυθεί πριν πολλά χρόνια και παρουσιάζουμε συνοπτικά τη λύση του. Έστω ότι η καμπύλη $\mathcal{C}$ αποτελείται από n σημεία $1, \ldots, n$ του επιπέδου με αντίστοιχες καρτεσιανές συντεταγμένες:

$$\mathcal{P} : (x_1, y_1), \ldots, (x_n, y_n) \tag{1.29}$$

όπου δεν βλάπτει να υποθέσουμε ότι $x_1 < x_2 < \ldots < x_n$. Είναι γνωστό ότι η ευθεία (1.28) ελαχιστοποιεί[6] το λάθος $e_{1,n}$:

$$e_{1,n} = \sum_{i=1}^{n} (y_i - \alpha x_i - \beta)^2 \tag{1.30}$$

προσεγγίσεως στα n σημεία $1, \ldots, n$ που περιγράφουν την καμπύλη $\mathcal{C}$ όταν οι συντελεστές α, β της L προκύπτουν από τις απλές εξισώσεις:

$$\alpha = \frac{n \sum_{i=1}^{n} y_i x_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \qquad \beta = \frac{\sum_{i=1}^{n} y_i - \alpha \sum_{i=1}^{n} x_i}{n} \tag{1.31}$$

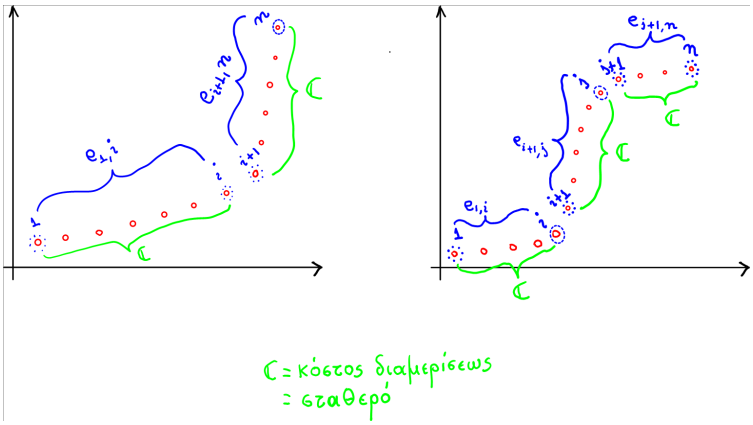
Συνοψίζοντας τα παραπάνω, στο Σχήμα (1.14) αν μας δώσουν τα σημεία (1.29) τότε υπολογίζουμε σε γραμμικό χρόνο το λάθος (1.30) της προσέγγισης της καμπύλης $\mathcal{C}$ από την ευθεία L με χρήση των εξισώσεων (1.31).

Όταν μόνο 1 ευθεία δεν φτάνει. Αντιλαμβανόμαστε οπτικά ότι η μορφή της καμπύλης των σημείων $1, \ldots, n$ του Σχήματος (1.14) είναι τέτοια ώστε η προσέγγιση της με 1 ευθεία L να δίνει μικρό σφάλμα $e_{1,n}$.

Όμως, η μορφή της καμπύλης $\mathcal{C}$ μπορεί να είναι τόσο περίπλοκη ώστε ακόμα και η πιο έξυπνη χρήση 1 ευθείας μόνο να μας δίνει μεγάλο σφάλμα $e_{1,n}$. Π.χ. δείτε το αριστερό γράφημα του Σχήματος (1.15) όπου οι περισσότεροι άνθρωποι εκτιμούν οπτικά, εκμεταλλευόμενοι ταχύτατους μηχανισμούς της βιολογικής τους όρασης, ότι 2 ευθείες δίνουν πολύ μικρότερο σφάλμα από ότι η 1 ευθεία μόνο. Όμοια στο δεξιό γράφημα του Σχήματος (1.15) η βιολογική μας όραση μας οδηγεί γοργά στην εκτίμηση ότι 3 ευθείες δίνουν μικρό σφάλμα σε σχέση με ≤ 2 ευθείες.

Οι σημερινοί υπολογιστές με είσοδο τα n σημεία της (1.29) δεν διαθέτουν τους γρήγορους μηχανισμούς βέλτιστης προσέγγισης του ανθρώπινου ματιού.

[6] Παρατηρήστε ότι το λάθος γίνεται $e_{1,n} = 0$ αν όλα τα σημεία $1, \ldots, n$ ανήκουν στην ευθεία L.



Σχήμα 1.15: Στο αριστερό γράφημα μπορούμε να διαμερίσουμε τα σημεία $\{1, \ldots, n\}$ στο $\{1, \ldots, i\}$ και στο $\{i+1, \ldots, n\}$. Το κόστος κάθε διαμερίσεως δίνεται από την σταθερά C. Με χρήση της (1.30) υπολογίζουμε τα αντίστοιχα σφάλματα $e_{1,i}$ και $e_{i+1,n}$ ανά διαμέριση. Το συνολικό κόστος του αριστερού γραφήματος είναι $2C + e_{1,i} + e_{i+1,n}$. Όμοια, το συνολικό κόστος του δεξιού γραφήματος είναι $3C + e_{1,i} + e_{i+1,j} + e_{j+1,n}$.

Τα παραπάνω εύκολα οδηγούν στο συμπέρασμα ότι όσες ποιο πολλές ευθείες έχουμε την δυνατότητα να χρησιμοποιήσουμε τόσο ποιο μικρό σφάλμα θα έχουμε. Ειδικότερα, το σφάλμα θα γίνει 0 αν μπορούσαμε να χρησιμοποιήσουμε ακριβώς 1 ευθεία για κάθε 1 ζεύγος διαδοχικών σημείων της καμπύλης $\mathcal{C}$. Όμως, θα έχουμε σπατάλη ευθειών, διότι για κάθε μια τέτοια ευθεία πληρώνουμε το κόστος C αποθήκευσης των α, β συντελεστών της. (Βοηθάει εδώ το Σχήμα (1.15) όπου στην αριστερή καμπύλη κάνουμε χρήση 2 και στην δεξιά 3 ευθειών, αντίστοιχα, πληρώνοντας κόστος C ανά ευθεία.) Συνεπώς, το συνολικό κόστος ευθειών, για όλα τα διαδοχικά ζεύγη σημείων της $\mathcal{C}$, θα ανέλθει στο κόστος αποθηκεύσεως όλων των σημείων στην (1.29). Καταστρέφουμε έτσι την ωραία ιδέα συμπιέσεως στην μορφή 2 σταθερών συντελεστών α, β μιας καμπύλης $\mathcal{C}$.

Ορισμός του προβλήματος. Αντιλαμβανόμαστε την εξαιρετική σπουδαιότητα του προβλήματος της προσεγγίσεως μιας οποιαδήποτε καμπύλης με χρήση το δυνατόν λιγότερων ευθειών. Αυστηρά:

Μας δίνουν τα σημεία $p_i = (x_i, y_i), i = 1, \ldots, n$ της καμπύλης $\mathcal{C}$ στην μορφή της Σχέσεως (1.29).

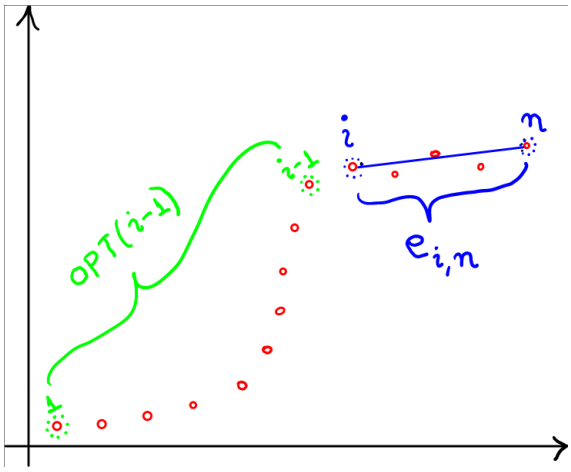
1. Πρέπει να διαμερίσουμε τα n σημεία $\{p_1, \ldots, p_n\} = \{p_1, \ldots, p_{i_1}\} \cup \{p_{i_1+1}, \ldots, p_{i_2}\} \cup \ldots \cup \{p_{i_k+1}, \ldots, p_n\}$ σε διαδοχικά μέρη.
2. Θα πληρώνουμε το σφάλμα $e_{i,j}$ της αντίστοιχης ευθείας που προσεγγίζει το μέρος της καμπύλης αποτελούμενο από τα σημεία $\{p_i, \ldots, p_j\}$, υπολογισμένο σύμφωνα με την (1.30).
3. Επίσης, θα πληρώνουμε ένα προκαθορισμένο κόστος C ανά μέρος $\{p_i, \ldots, p_j\}$ που θα χρησιμοποιήσουμε.

Στόχος είναι να βρούμε την βέλτιστη διαμέριση των σημείων

$$\{p_1, \ldots, p_n\} = \{p_1, \ldots, p_{i_1}\} \cup \{p_{i_1+1}, \ldots, p_{i_2}\} \cup \ldots \cup \{p_{i_k+1}, \ldots, p_n\}$$

της καμπύλης $\mathcal{C}$ που ελαχιστοποιεί το συνολικό κόστος. Αντιλαμβανόμαστε ότι το δίλημμα είναι το εξής: ο διαμερισμός της καμπύλης $\mathcal{C}$ σε πολλά μέρη, από την μία, θα μικρύνει το αντίστοιχο σφάλμα $e_{i,j}$ ανά μέρος $\{p_i, \ldots, p_j\}$ όπως περιγράφουμε στο 2 άνω, από την άλλη, θα πληρώνουμε τίμημα C ανά μέρος $\{p_i, \ldots, p_j\}$, όπως περιγράφουμε στο 3 άνω.

1.7.2 Κατασκευή της Αναδρομής



Σχήμα 1.16:

Για να υπολογίσουμε την βέλτιστη διαμέριση $\mathcal{O}$ των σημείων $p_1, \ldots, p_n$ με αντίστοιχο κόστος $\mathcal{O}(n)$ παρατηρούμε ότι το δεξιότερο σημείο p_n θα ανήκει στο δεξιότερο μέρος της βέλτιστης διαμέρισης $\mathcal{O}$. Το δεξιότερο μέρος, άρα, θα είναι της μορφής $\{p_i, \ldots, p_n\}$ αρχίζοντας σε ένα *άγνωστο* σημείο p_i με $1 \leq i \leq n-1$.

Έστω ότι ένας σοφός μας δείχνει αυτό το άγνωστο σημείο p_i, δείτε το στο Σχήμα 1.16. Τότε βγάζουμε το συμπέρασμα ότι τα σημεία $p_i, \ldots, p_n$ ανήκουν στο δεξιότερο μέρος (στο Σχήμα 1.16 είναι αυτά που τα ενώνει 1 μπλε γραμμή) της βέλτιστης διαμέρισης $\mathcal{O}$ και ξέρουμε ότι θα μας κοστίσουν σφάλμα $e_{i,n}$, συν, το κόστος C που πληρώνουμε για την ευθεία L σε κάθε 1 μέρος. Αφού ο σοφός μας είπε ότι όλα τα $p_i, \ldots, p_n$ ανήκουν στο δεξιότερο μέρος της βέλτιστης διαμερίσεως μπορούμε να τα αγνοήσουμε και να επικεντρώσουμε στα υπόλοιπα $\{p_1, \ldots, p_n\} \setminus \{p_i, \ldots, p_n\} = \{p_1, \ldots, p_{i-1}\}$ σημεία (στο Σχήμα 1.16 είναι αυτά που τα περικλείει η πράσινη αγκύλη) που απομένουν να χωριστούν σε μέρη με βέλτιστο τρόπο.

Συνεπώς, με τη συμβολή του σοφού, η αξία $\mathcal{O}(n)$ της βέλτιστης διαμέρισης του $\{p_1, \ldots, p_n\}$ ισούται με το σφάλμα $e_{i,n}$ (διότι το μέρος $\{p_i, \ldots, p_n\}$ ανήκει στη βέλτιστη διαμέριση) συν το κόστος C που πρέπει να πληρώσουμε για κάθε 1 μέρος συν την αξία $\mathcal{O}(i-1)$ της βέλτιστης διαμέρισης στα σημεία $\{p_1, \ldots, p_{i-1}\}$ που απομένουν να χωριστούν σε μέρη. Συμβολικά:

$$\mathcal{O}(n) = e_{i,n} + C + \mathcal{O}(i-1) \tag{1.32}$$

Το μειονέκτημα της παραπάνω αναδρομής (1.32) είναι ότι προήλθε με την ουτοπική βοήθεια του σοφού. Όπως και στα προηγούμενα παραδείγματα, εύκολα και λογικά, μπορούμε να αποφύγουμε στην αναδρομή (1.32) την εξάρτηση μας από τη σοφή υπόδειξη του σημείου p_i.

Αρκεί να παρατηρήσουμε ότι οι δυνατές σοφές συμβουλές που υπάρχουν είναι όσες και οι δυνατές τιμές του $i = 1, \ldots, n$, δηλαδή $O(n)$. Λαμβάνουμε έτσι $O(n)$ διαφορετικά υποπροβλήματα, ένα για κάθε τιμή του i, με αντίστοιχα κόστη:

$$\{e_{1,n} + C + \mathcal{O}(1-1)\}, \{e_{2,n} + C + \mathcal{O}(2-1)\}, \ldots, \{e_{n,n} + C + \mathcal{O}(n-1)\} \tag{1.33}$$

Επειδή ο σκοπός μας είναι να ελαχιστοποιήσουμε το συνολικό κόστος πρέπει να επιλέξουμε εκείνο το $1 \leq i \leq n$ που θα μας δώσει εκείνο το υποπρόβλημα από τα (1.33) με το *μικρότερο* κόστος. Συμβολικά[7]:

$$\mathcal{O}(n) = \min_{1 \leq i \leq n} \{e_{i,n} + C + \mathcal{O}(i-1)\} \tag{1.34}$$

[7] Αξίζει σε αυτό το σημείο να μελετήσουμε την κατασκευή της Αναδρομής (1.38) στην Ενότητα 1.8.

Η ζητούμενη αναδρομή είναι η (1.34).

1.7.3 Απομνημόνευση της Αναδρομής

Παρατηρούμε ότι για να απομνημονεύσουμε κάθε τιμή $\mathcal{O}(j)$ της αναδρομής (1.34) σε ένα πίνακα $M[j]$ πρέπει να έχουμε προϋπολογίσει για κάθε $j = 1, \ldots, n$ τα αντίστοιχα σφάλματα $e_{i,j}$ για όλες τις τιμές του $i = 1, \ldots, j$. Όπως είπαμε στην αρχή της ενότητας, αυτό γίνεται με τη Σχέση (1.30) που την καλεί ο παρακάτω αλγόριθμος και την αποθηκεύει στην αντίστοιχη λίστα $e[i, j]$:

for$(i = 1; i <= n; i++)$
  for$(j = 1; j <= i; j++)$
    $e[i, j] = e_{i,j}$;

Τα δυνατά ζευγάρια είναι $O(n^2)$. Στην συνέχεια, ο παρακάτω αλγόριθμος υπολογίζει την τιμή $\mathcal{O}(j)$ με την αναδρομή (1.34) και την αποθηκεύει στη λίστα $M[j]$.

$M[0] = 0$;
for$(j = 1; j <= n; j++)$
  $M[j] = \min_{1 \le i \le j} \{e_{i,j} + C + M[i-1]\}$;

Η κρίσιμη διαφορά του αλγορίθμου από όλους τους προγενέστερους είναι η εξής: για να υπολογίσει την τρέχουσα τιμή (τουβλάκι) $M[j]$ κάνει χρήση (για να βρει το ελάχιστο $\min_{1 \le i \le j}$) *όλων των ήδη απομνημονευμένων τιμών* (ήδη χτισμένα τουβλάκια) $M[1], \ldots, M[j-1]$. Αν θυμηθείτε τα προηγούμενα παραδείγματα, οι αλγόριθμοι που εξετάσαμε έκαναν χρήση το πολύ 2 απομνημονευμένων τιμών (2 ήδη χτισμένα τουβλάκια). Ο πολυωνυμικός χρόνος λειτουργίας εξασφαλίζεται διότι οι απομνημονευμένες τιμές είναι $O(j)$ και τα αντίστοιχα σφάλματα $O(j)$ για κάθε $j = 1, \ldots, n$ δίνοντας συνολικά χρόνο $O(n^2)$.

1.8 Πολλαπλασιασμός αλληλουχίας πινάκων (Matrix chain multiplication)

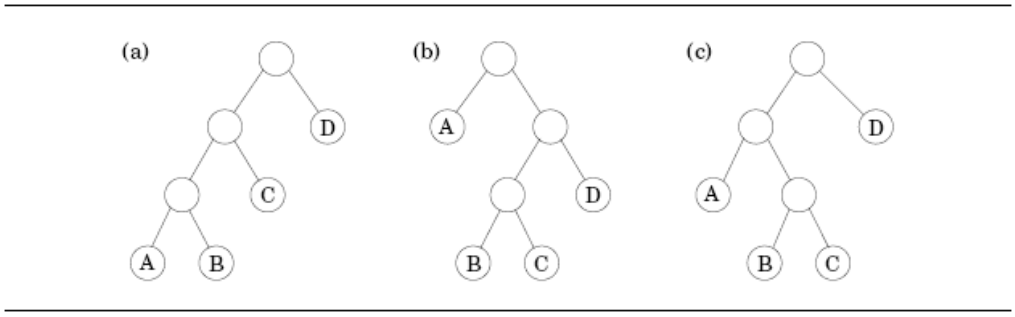
Ο τρόπος εργασίας είναι γενίκευση της προηγούμενης Ενότητας 1.7. Εδώ η επιπλέον δυσκολία είναι ότι η βέλτιστη λύση πρέπει να υπολογιστεί και στο αριστερό και στο δεξιό υποπρόβλημα, εν αντιθέσει με την προηγούμενη ενότητα όπου είχαμε μόνο το αριστερό υποπρόβλημα.

1.8.1 Περιγραφή του Προβλήματος

Μας ζητάνε να πολλαπλασιάσουμε n πίνακες $A_1 \times \ldots \times A_n$ το ταχύτερο δυνατόν. Είναι γνωστό ότι το γινόμενο ενός πίνακα $A[m, n]$ επί $B[n, p]$ απαιτεί *κόστος* $m \cdot n \cdot p$ ίσο με τους επιμέρους πολ/μους που απαιτούνται ώστε να λάβουμε το πίνακα-αποτέλεσμα $C[m, p]$. Η σειρά που θα κάνουμε τους πολ/μους, δηλαδή που θα βάλουμε παρενθέσεις που δηλώνουν την προτεραιότητα, έχει τεράστια σημασία. Για παράδειγμα, υπολογίζουμε το γινόμενο $A[50, 20] \times B[20, 1] \times A[1, 10] \times A[10, 100]$ με 3 διαφορετικούς τρόπους τοποθετήσεως παρενθέσεων στο Σχήμα 1.17. Το Σχήμα 1.17 δείχνει ότι το κόστος μεταβάλλεται εκπληκτικά, όπου ο ταχύτερος (βραδύτερος) τρόπος είναι στη 3η (1η) γραμμή. Το Σχήμα 1.18 δείχνει καλύτερα την σημασία των παρενθέσεων στο γινόμενο $A \times B \times C \times D = A_1 \times A_2 \times A_3 \times A_4$. Από το Σχήμα 1.18 γίνεται φανερό ότι αν σε κάθε κορυφή (πρόβλημα) του δέντρου στα 2 παιδιά (υποπροβλήματα) του τα επιμέρους γινόμενα έχουν γίνει με το βέλτιστο τρόπο τότε και σε ολόκληρο το δέντρο (αρχικό πρόβλημα) θα έχουν γίνει όλα τα γινόμενα με το βέλτιστο τρόπο. Παρατηρούμε ότι κάθε παιδί (υποπρόβλημα) του δέντρου αποτελείται από μια συγκεκριμένη αλληλουχία γινομένων $A_i \times A_{i+1} \times \cdots \times A_j$. Συνεπώς, είναι σημαντικό να υπολογίσουμε το βέλτιστο πλήθος γινομένων για ένα δοσμένο υποπρόβλημα $A_i \times A_{i+1} \times \cdots \times A_j$. Ορίζουμε ως $\mathcal{O}(i, j)$ το βέλτιστο πλήθος γινομένων για το υποπρόβλημα $A_i \times A_{i+1} \times \cdots \times A_j$, όπου οι αντίστοιχες διαστάσεις των πινάκων είναι $A_i[m_{i-1}, m_i], A_{i+1}[m_i, m_{i+1}], \ldots, A_j[m_{j-1}, m_j]$.

$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	$120,200$
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	$60,200$
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	$7,000$

Σχήμα 1.17: 1η στήλη: σειρά των παρενθέσεων. 2η στήλη: άθροισμα κόστους ανά ζεύγους γινομένων με την προτεραιότητα των παρενθέσεων. 3η στήλη: τελικό κόστος.



Σχήμα 1.18: (a) $((A \times B) \times C) \times D$: εδώ το πρόβλημα $A_1, \ldots, A_4$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_1, \ldots, A_3) \times (A_4)$. Μετά το πρόβλημα $(A_1, \ldots, A_3)$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_1, A_2) \times (A_3)$. Τέλος, το πρόβλημα (A_1, A_2) με παρενθέσεις σπάει στα υποπροβλήματα $(A_1) \times (A_2)$.

(b) $A \times ((B \times C) \times D)$: εδώ το πρόβλημα $A_1, \ldots, A_4$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_1) \times (A_2, \ldots, A_4)$. Μετά το πρόβλημα $(A_2, \ldots, A_4)$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_2, A_3) \times (A_4)$. Τέλος, το πρόβλημα (A_2, A_3) με παρενθέσεις σπάει στα υποπροβλήματα $(A_2) \times (A_3)$.

(c) $(A \times (B \times C)) \times D$: εδώ το πρόβλημα $A_1, \ldots, A_4$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_1, \ldots, A_3) \times (A_4)$. Μετά το πρόβλημα $(A_1, \ldots, A_3)$ με παρενθέσεις σπάει στα υποπροβλήματα $(A_1) \times (A_2, A_3)$. Τέλος, το πρόβλημα (A_2, A_3) με παρενθέσεις σπάει στα υποπροβλήματα $(A_2) \times (A_3)$.

1.8.2 Κατασκευή της Αναδρομής

• (Λάθος τρόπος.) Ξέρουμε ότι ο δεξιότερος πίνακας θα είναι η δεξιότερη παρένθεση (από τις 4 συνολικά) που θα χωρίσουν το αρχικό πρόβλημα. Έστω ότι ο σοφός μας λέει ότι *στη θέση* i θα κλείσει αυτή η παρένθεση, δηλαδή $(A_{i+1} \times A_{i+2} \times \cdots \times A_n)$. Συνεπώς, αρκεί να βρούμε το βέλτιστο πλήθος γινομένων $\mathcal{O}(1,i)$ για το αριστερό υποπρόβλημα $(A_1 \times A_2 \times \cdots \times A_i)$. Σχηματικά:

$$\mathcal{O}(1,n) = \mathcal{O}(1,i) \tag{1.35}$$

Το λάθος στον άνω συλλογισμό είναι ότι μελετούμε μόνο τη περαιτέρω βέλτιστη τοποθέτηση παρενθέσεων στο αριστερό υποπρόβλημα $(A_1 \times A_2 \times \cdots \times A_i)$ και αγνοούμε παντελώς το δεξιό υποπρόβλημα $(A_{i+1} \times A_{i+2} \times \cdots \times A_n)$ και την τοποθέτηση παρενθέσεων σε αυτό.

• (Λάθος τρόπος.) Έστω ότι ο σοφός μας λέει ότι *στη θέση* i το αρχικό πρόβλημα $A_1 \times A_2 \times \cdots \times A_n$ (πατέρας του δυαδικού δέντρου στο Σχήμα 1.18) σπάει με παρένθεση στα 2 υποπροβλήματα (2 παιδιά του πατέρα του δυαδικού δέντρου στο Σχήμα 1.18) $(A_1 \times A_2 \times \cdots \times A_i)$ και $(A_{i+1} \times A_{i+2} \times \cdots \times A_n)$. Τότε το βέλτιστο πλήθος γινομένων $\mathcal{O}(1,n)$ του αρχικού προβλήματος $A_1 \times A_2 \times \cdots \times A_n$ θα είναι ίσο με το βέλτιστο πλήθος γινομένων $\mathcal{O}(1,i)$ του αριστερού υποπροβλήματος $(A_1 \times A_2 \times \cdots \times A_i)$ συν το βέλτιστο πλήθος γινομένων $\mathcal{O}(i+1,n)$ του δεξιού υποπροβλήματος $(A_{i+1} \times A_{i+2} \times \cdots \times A_n)$ που προέκυψαν με την τοποθέτηση των παρενθέσεων. Σχηματικά:

$$\mathcal{O}(1,n) = \mathcal{O}(1,i) + \mathcal{O}(i+1,n) \tag{1.36}$$

• (Σωστός τρόπος.) Στην άνω αναδρομή (1.36) δεν έχουμε προσθέσει και το *κόστος* που απαιτεί ο πολ/μος του *αριστερού πίνακα*, διαστάσεως $[m_0, m_i]$, που προκύπτει από την βέλτιστη εκτέλεση των $\mathcal{O}(1,i)$ γινομένων του αριστερού υποπροβλήματος $(A_1 \times A_2 \times \cdots \times A_i)$ επί τον *δεξιό πίνακα*, διαστάσεως $[m_i, m_n]$, που προκύπτει από την βέλτιστη εκτέλεση των $\mathcal{O}(i+1,n)$ γινομένων του δεξιού υποπροβλήματος $(A_{i+1} \times A_{i+2} \times \cdots \times A_n)$. Όπως έχουμε πει, το κόστος πολ/μου πίνακα $[m_0, m_i]$ επί πίνακα $[m_i, m_n]$ είναι ίσο με $m_0 \cdot m_i \cdot m_n$. Καταλήγουμε ότι αν ο σοφός πει ότι το δοσμένο πρόβλημα σπάει *στη θέση* i η σωστή αναδρομή είναι:

$$\mathcal{O}(1,n) = \mathcal{O}(1,i) + \mathcal{O}(i+1,n) + m_{1-1} \cdot m_i \cdot m_n \tag{1.37}$$

• (Τελική αναδρομή.) Η Αναδρομή (1.37) βασίζεται στην συμβουλή του σοφού ότι το πρόβλημα σπάει στην θέση i από τις παρενθέσεις. Επειδή για το δοσμένο πρόβλημα $A_1 \times A_2 \times \cdots \times A_n$ οι δυνατότητες για την θέση σπασίματος i είναι $i = 1, \ldots, n$ θα επιλέξουμε την θέση που δίνει το ελάχιστο κόστος μεταξύ όλων των δυνατών υποπροβλημάτων. Σχηματικά[8]:

$$\mathcal{O}(1,n) = \min_{1 \le i \le n} \{\mathcal{O}(1,i) + \mathcal{O}(i+1,n) + m_{1-1} \cdot m_i \cdot m_n\} \tag{1.38}$$

1.8.3 Απομνημόνευση της Αναδρομής

Η βέλτιστη αξία υπολογίζεται με χρήση της αναδρομής (1.38) με τον αλγόριθμο:

`for`$(i = 1; i <= n; i++)$
$M[i,i] = 0;$

[8] Αξίζει σε αυτό το σημείο να δούμε ξανά το τρόπο κατασκευής της αντίστοιχης αναδρομής στην Ενότητα 1.7.2. Η σημαντική διαφορά της (1.38) από την (1.34) είναι η εξής: Η (1.38) σπάει το πρόβλημα $\{A_s, \ldots, A_t\}$ στα 2 υποπροβλήματα $\{A_s, \ldots, A_i\}$ και $\{A_{i+1}, \ldots, A_t\}$ και απαιτεί 2 δείκτες για τον καθορισμό του αριστερού&δεξιού άκρου κάθε υποπροβλήματος και τα αντίστοιχα βέλτιστα κόστη $\mathcal{O}(s,i), \mathcal{O}(i+1,t)$. Η (1.34) έσπαγε το πρόβλημα $\{A_1, \ldots, A_n\}$ στο 1 υποπρόβλημα $\{A_1, \ldots, A_i\}$ και απαιτούσε μόνο ένα δείκτη i για το αντίστοιχο $\mathcal{O}(i-1)$.

```
for(s = 1; s <= n − 1; i++)
    for(i = 1; i <= n − s; i++)
        j = i + s;
        M[i, j] = min_{i≤k<j}{M[i, k] + M[k + 1, j] + m[i − 1] ∗ m[k] ∗ m[j]};
return(M[1, n]);
```

```
> restart:
  n:= 4:

  for i from 1 by 1 to n do
  M[i,i]:=0:
  end do:

  m[0]:= 50:m[1]:= 20: m[2]:= 1: m[3]:= 10: m[4]:= 100:

> for s from 1 by 1 to n-1 do
  for i from 1 by 1 to n-s do
  j:= i+s:

  M2:= infinity:
  for k from i by 1 to j-1 do
  M1:= M[i,k]+M[k+1,j]+ m[i-1]*m[k]*m[j]:
  if (M1< M2) then M2:= M1 end if:
  end do:
  M[i,j]:= M2:

  end do:
  end do:
> M[1,4];
                                                    7000
```

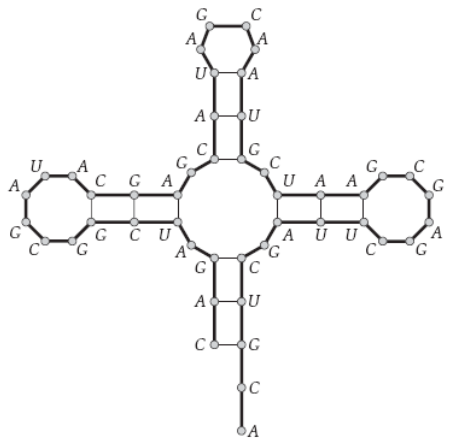
Σχήμα 1.19: Τρέχουμε το maple της αναδρομής (1.38) με είσοδο το παράδειγμα στο Σχήμα 1.17.

1.9 RNA δευτερεύουσα δομή

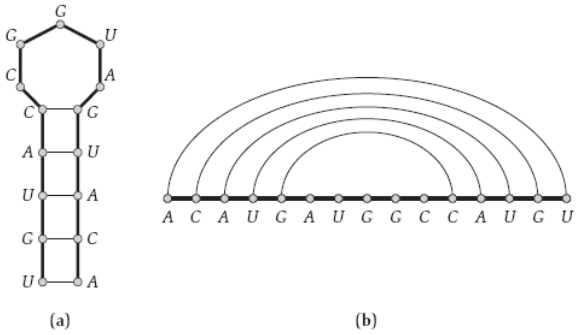
1.9.1 Περιγραφή του Προβλήματος

Το Σχήμα 1.20 εμφανίζει ένα μόριο RNA μήκους (πλήθος βάσεων) 46. Ένα μόριο RNA μήκους n είναι μια αλληλουχία βάσεων (γραμμάτων) $b_1 b_2 \ldots b_n$, όπου σε κάθε θέση $i \in \{1, \ldots, n\}$ του μορίου μπορεί να εμφανιστεί η βάση b_i ως ένα από τα γράμματα $\{A, C, U, G\}$. Η *δευτερεύουσα δομή* ενός μορίου RNA είναι το σύνολο S από ζεύγη θέσεων (i, j) του μορίου, όπου οι αντίστοιχες βάσεις b_i και b_j σε αυτές τις θέσεις είναι ενωμένες. Στο Σχήμα 1.20, η δευτερεύουσα δομή αποτελείται από τα ζεύγη των θέσεων που ενώνονται ανά δύο οι αντίστοιχες βάσεις με λεπτές γραμμές. Οι κανόνες για να ενωθούν με λεπτή γραμμή δύο βάσεις b_i, b_j που βρίσκονται στις θέσεις (i, j) είναι:

1. Μεταξύ των ενωμένων βάσεων b_i, b_j πρέπει να υπάρχουν ≥ 4 άλλες βάσεις, δηλαδή για τις θέσεις (i, j) ισχύει: $i < j - 4$.
2. Οι ενωμένες βάσεις b_i, b_j μπορεί να είναι $\{A, U\}$ ή $\{C, G\}$.
3. Κάθε βάση μπορεί να ενωθεί με ≤ 1 άλλες βάσεις.
4. Οι λεπτές γραμμές που ενώνουν ζεύγη βάσεων δεν τέμνονται. Δηλαδή, αν $(i, j), (k, l) \in S$ τότε δεν είναι δυνατόν να ισχύει $i < k < j < l$.



Σχήμα 1.20: Το μόριο RNA αποτελείται από την αλληλουχία βάσεων που ενώνει η έντονη μαύρη γραμμή. Η δευτερεύουσα δομή του μορίου σχηματίζεται από τις συνδέσεις, με λεπτές γραμμές, μεταξύ των βάσεων του.



Σχήμα 1.21: Τα Σχήματα (a) και (b) εμφανίζουν το ίδιο μόριο RNA. Τα άκρα κάθε ζεύγους ενωμένων βάσεων με λεπτή γραμμή έχουν μεταξύ τους ≥ 4 βάσεις. Καμιά λεπτή γραμμή δεν τέμνει άλλη. Από κάθε βάση ξεκινάει ≤ 1 λεπτή γραμμή. Τα άκρα κάθε λεπτής γραμμής είναι $\{A, U\}$ ή $\{C, G\}$.

Το Σχήμα 1.21 δείχνει πιο καλά τους παραπάνω κανόνες σύνδεσης με λεπτές γραμμές.

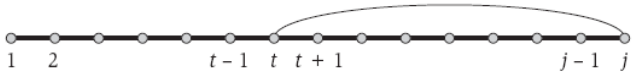
Το ζητούμενο είναι, όταν μας δώσουν ένα μόριο RNA, να υπολογίσουμε την μέγιστη δευτερεύουσα δομή για αυτό το μόριο. Διαισθητικά, να τοποθετήσουμε το μέγιστο πλήθος λεπτών γραμμών, ενώνοντας ζεύγη βάσεων ικανοποιώντας τους παραπάνω τέσσερις περιορισμούς.

1.9.2 Κατασκευή της Αναδρομής

(Λάθος τρόπος.) Ορίζουμε ως $\mathcal{O}(n)$ το μέγιστο πλήθος λεπτών γραμμών (μέγεθος δευτερεύουσας δομής) που μπορούν να εμφανιστούν στις θέσεις $\{1, \ldots, n\}$ του RNA-μορίου $b_1 \ldots b_n$, ικανοποιώντας τους περιορισμούς 1-4 της Ενότητας 1.9.1. Πρέπει να προσέξουμε ότι ορίζοντας έτσι το $\mathcal{O}(n)$ αυτό αναφέρεται πάντα σε υποπρόβλημα της μορφής $\{1, \ldots, n\}$. Αυτό σημαίνει ότι η *αριστερότερη* θέση του υποπροβλήματος $\{1, \ldots, n\}$ είναι *πάντα* το 1, ενώ η *δεξιότερη* θέση του υποπροβλήματος $\{1, \ldots, n\}$ καθορίζεται από τον φυσικό αριθμό n του $\mathcal{O}(n)$. Γενικότερα:

$$\text{Αντιστοιχούμε τη βέλτιστη λύση } \mathcal{O}(j) \text{ στο υποπρόβλημα } \{1, \ldots, j\} \tag{1.39}$$

Στη συνέχεια, με την βοήθεια του Σχήματος 1.22, θα δούμε ότι η κωδικοποίηση (1.39) δεν είναι δυνατόν να μας οδηγήσει σε σωστή κατασκευή της αναδρομικής σχέσης. Σύμφωνα με το Σχήμα 1.22, για την βάση



Σχήμα 1.22: Το υποπρόβλημα $\{1, \ldots, j\}$ που αντιστοιχεί στην βέλτιστη λύση $\mathcal{O}(j)$, βάση της λανθασμένης κωδικοποίησης (1.39).

b_j στην θέση j υπάρχουν οι εξής 2 δυνατότητες:

- Η βάση b_j στη θέση j *δεν* ενώνει με λεπτή γραμμή με άλλη βάση. Προφανώς τότε ισχύει:

$$\mathcal{O}(j) = \mathcal{O}(j-1) \tag{1.40}$$

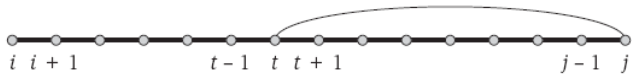
- Η βάση b_j στη θέση j *ενώνει* με λεπτή γραμμή με άλλη βάση b_t σε κάποια θέση $1 \leq t < j-4$. Αφού η βάση b_j ενώνεται με λεπτή γραμμή με τη βάση b_t, ο περιορισμός 4 της Ενότητας 1.9.1 σημαίνει ότι δεν υπάρχουν λεπτές γραμμές με το ένα άκρο στο υποπρόβλημα $\{1, \ldots, t-1\}$ και το άλλο άκρο στο υποπρόβλημα $\{t+1, \ldots, j\}$. Αυτό σημαίνει ότι η ένωση των βάσεων b_j, b_t δημιουργεί τα δύο *ανεξάρτητα* υποπροβλήματα: $\{1, \ldots, t-1\}$ (αριστερά από τη θέση t στο Σχήμα 1.22) και $\{t+1, \ldots, j\}$ (δεξιά από τη θέση t στο Σχήμα 1.22). Συνεπώς, η βέλτιστη λύση $\mathcal{O}(j)$ για το πρόβλημα $\{1, \ldots, j\}$ μπορεί να προκύψει από το άθροισμα των δυο επιμέρους βέλτιστων λύσεων. Η πρώτη επιμέρους λύση (σύμφωνα με την κωδικοποίηση 1.39) είναι η βέλτιστη λύση $\mathcal{O}(t-1)$ του αριστερού υποπροβλήματος $\{1, \ldots, t-1\}$, διότι η αριστερότερη θέση του υποπροβλήματος $\{1, \ldots, t-1\}$ είναι 1. Δυστυχώς, η δεύτερη επιμέρους λύση, είναι η βέλτιστη λύση του δεξιού υποπροβλήματος $\{t+1, \ldots, n\}$, η οποία δεν μπορεί να γραφεί σύμφωνα με την κωδικοποίηση (1.39), διότι η αριστερότερη θέση του υποπροβλήματος $\{t+1, \ldots, n\}$ είναι $t+1 \neq 1$.

Φθάσαμε στο παραπάνω αδιέξοδο, διότι στην κωδικοποίηση (1.39) η αριστερότερη θέση κάθε υποπροβλήματος υποχρεωτικά ήταν το 1.

(Σωστός τρόπος.) Η διέξοδος είναι να γενικεύσουμε την κωδικοποίηση (1.39), ώστε να είναι μεταβλητή η αριστερότερη θέση κάθε υποπροβλήματος, ως εξής:

$$\text{Αντιστοιχούμε τη βέλτιστη λύση } \mathcal{O}(i, j) \text{ στο υποπρόβλημα } \{i, \ldots, j\} \tag{1.41}$$

Με βάση την σχέση αυτή, λαμβάνουμε το Σχήμα 1.23, όπου για την βάση b_j στην θέση j υπάρχουν οι εξής 2 δυνατότητες:



Σχήμα 1.23: Το υποπρόβλημα $\{i, \ldots, j\}$ που αντιστοιχεί στην βέλτιστη λύση $\mathcal{O}(i, j)$, βάση της σωστής κωδικοποίησης (1.41).

- Ο σοφός μας λέει ότι η βάση b_j στη θέση j *δεν* ενώνετε με λεπτή γραμμή με άλλη βάση. Προφανώς τότε ισχύει:

$$\mathcal{O}(i, j) = \mathcal{O}(i, j - 1) \tag{1.42}$$

- Ο σοφός μας δείχνει μια θέση t όπου η βάση b_j στη θέση j *ενώνετε* με λεπτή γραμμή με τη συμβατή βάση b_t στη θέση $1 \leq t < j - 4$.

$$\mathcal{O}(i, j) = \mathcal{O}(i, t - 1) + \mathcal{O}(t + 1, j) \tag{1.43}$$

  Προφανώς, εδώ πρέπει να επιλέξουμε το μέγιστο της Σχέσης (1.43), για όλες τις δυνατές & συμβατές θέσεις $1 \leq t < j - 4$, δηλαδή

$$\mathcal{O}(i, j) = \max_{1 \leq t < j-4} \{\mathcal{O}(i, t - 1) + \mathcal{O}(t + 1, j)\} \tag{1.44}$$

Τότε, είναι λογικό η βέλτιστη λύση να προκύπτει ως η μέγιστη αξία από τις δύο άνω περιπτώσεις (1.42) και (1.44) που προέκυψαν από τις συμβουλές του σοφού. Με άλλα λόγια:

$$\mathcal{O}(i, j) = \max\{\mathcal{O}(i, j - 1), \max_{1 \leq t < j-4} \{\mathcal{O}(i, t - 1) + \mathcal{O}(t + 1, j)\}\} \tag{1.45}$$

1.9.3 Απομνημόνευση της Αναδρομής

Αρχικοποίηση:

for$(i = 1; i <= n; i++)$
  for$(j = 1; j <= i; j++)$
    if $(i \leq j - 4)$ $M[i, j] = 0$;

Στην συνέχεια:

for$(k = 5; i <= n - 1; i++)$
  for$(i = 1; j <= n - k; j++)$
    Set $j = i + k$;
    Return $M(i, j)$ using (1.45)

4	0	0	0	
3	0	0		
2	0			
i = 1				
	j = 6	7	8	9

Initial values

4	0	0	0	1
3	0	0	1	
2	0	0		
i = 1	1			
	j = 6	7	8	9

k= 5

4	0	0	0	1
3	0	0	1	1
2	0	0	1	
i = 1	1	1		
	j = 6	7	8	9

k= 6

4	0	0	0	1
3	0	0	1	1
2	0	0	1	1
i = 1	1	1	1	
	j = 6	7	8	9

k= 7

4	0	0	0	1
3	0	0	1	1
2	0	0	1	1
i = 1	1	1	1	2
	j = 6	7	8	9

k= 8

Σχήμα 1.24: Η λειτουργία του Αλγορίθμου με είσοδο το RNA μόριο ACCGGUAGU για διάφορες τιμές της μεταβλητής $k = j - i$ που μετρά το μήκος του εκάστοτε υποπροβλήματος $\{i, \ldots, j\}$.

1.10 Βέλτιστα Δυαδικά Δέντρα Αναζήτησης (Optimal Binary Search Trees)

1.10.1 Περιγραφή του Προβλήματος

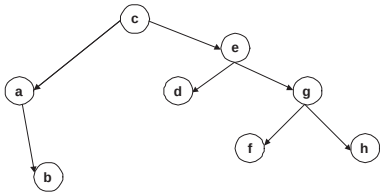
Ένα *δυαδικό δέντρο αναζήτησης* (ΔΔΑ) είναι ένα δέντρο όπου κάθε κόμβος i έχει το πολύ δύο παιδιά, με αξία v_i που είναι:

- $\geq$ όλων των αντίστοιχων αποθηκευμένων αξιών στο αριστερό υποδέντρο του κόμβου i,
- $\leq$ όλων των αντίστοιχων αποθηκευμένων αξιών στο δεξί υποδέντρο του κόμβου i.

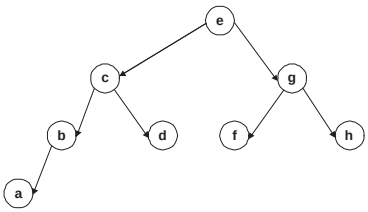
Ρίζα ενός δέντρου (υποδέντρου) είναι ο κόμβος που δεν είναι παιδί άλλου κόμβου, ενώ *φύλλο* είναι ένας κόμβος που δεν έχει παιδιά. Για παράδειγμα, αν αξίες $V = \{a, b, c, d, e, f, g, h\}$ είναι σε αύξουσα σειρά τότε δείτε δύο ΔΔΑ στα Σχήματα 1.25 και 1.26.
Όταν αναζητούμε ένα οποιοδήποτε στοιχείο σε ένα ΔΔΑ (ή υποδέντρο του), η ρίζα του ΔΔΑ (ή του υποδέντρου) έχει μεγάλη σημασία στο χρόνο αναζήτησης, διότι από την ρίζα αρχίζει κάθε αναζήτηση σε ένα δέντρο (ή υποδέντρο). *Βάθος* ($depth(i)$) του κόμβου i είναι το μήκος του μονοπατιού με αρχή την ρίζα και τέλος τον κόμβο i. *Πλήθος συγκρίσεων* $comp(i)$ για να βρούμε την αξία v_i του κόμβου i είναι το πλήθος των κόμβων του μονοπατιού με αρχή την ρίζα και τέλος τον κόμβο i. Ισχύει

$$comp(i) = depth(i) + 1 \tag{1.46}$$

Έχει μεγάλη σημασία η τιμή $comp(i), \forall i \in V$, διότι ο χρόνος αναζήτησης ενός στοιχείου εκφράζεται ως πλήθος συγκρίσεων στο ΔΔΑ μέχρι να βρούμε το στοιχείο που αναζητούμε. Με απλά λόγια, έχει σημασία το ΔΔΑ να έχει την πλειονότητα των κόμβων του σε μικρό βάθος από την ρίζα του. Για παράδειγμα, δείτε τα αντίστοιχα $depth(i), comp(i), \forall i \in V$, για τα δύο ΔΔΑ στα Σχήματα 1.25 και 1.26.



Σχήμα 1.25: Ρίζα είναι ο κόμβος c, φύλλα οι κόμβοι b, d, f, h. Το βάθος κάθε κόμβου είναι $depth(c) = 0, depth(a) = 1, depth(e) = 1, depth(b) = 2, depth(d) = 2, depth(g) = 2, depth(f) = 3, depth(h) = 3$. Τα πλήθη των συγκρίσεων, ξεκινώντας από την ρίζα c, για να φθάσουμε σε κάθε κόμβο είναι $comp(c) = 1, comp(a) = 2, comp(e) = 2, comp(b) = 3, comp(d) = 3, comp(g) = 3, comp(f) = 4, comp(h) = 4$. Αν κάθε κόμβος $\in V$ έχει ίση πιθανότητα να αναζητηθεί στο ΔΔΑ, δηλ. $\forall i \in V$ ισχύει $\Pr[i] = \frac{1}{8}$ τότε ο αναμενόμενος χρόνος αναζήτησης είναι $\sum_{i \in V} \Pr[i] comp(i) = \frac{22}{8}$.



Σχήμα 1.26: Ρίζα είναι ο κόμβος e, φύλλα οι κόμβοι a, d, f, h. Το βάθος κάθε κόμβου είναι $depth(e) = 0, depth(c) = 1, depth(g) = 1, depth(b) = 2, depth(d) = 2, depth(f) = 2, depth(h) = 2, depth(a) = 3$. Τα πλήθη των συγκρίσεων, ξεκινώντας από την ρίζα e, για να φθάσουμε σε ένα κόμβο είναι $depth(e) = 1, depth(c) = 2, depth(g) = 2, depth(b) = 3, depth(d) = 3, depth(f) = 3, depth(h) = 3, depth(a) = 4$. Αν κάθε κόμβος $\in V$ έχει ίση πιθανότητα να αναζητηθεί στο ΔΔΑ, δηλ. $\forall i \in V$ ισχύει $\Pr[i] = \frac{1}{8}$ τότε ο αναμενόμενος χρόνος αναζήτησης είναι $\sum_{i \in V} \Pr[i] comp(i) = \frac{21}{8}$.

Ο *αναμενόμενος χρόνος αναζήτησης* (ΑΧΑ) για ένα δοσμένο ΔΔΑ με σύνολο κόμβων V είναι ένα πολύ κρίσιμο χαρακτηριστικό της απόδοσης του ΔΔΑ. Ο ΑΧΑ ορίζεται ως

$$\sum_{i \in V} \Pr[i] comp(i) = \sum_{i \in V} \Pr[i](depth(i) + 1) \tag{1.47}$$

όπου $\Pr[i]$ δηλώνει πόσες φορές αναζητούμε την αξία v_i του κόμβου i σε ένα δοσμένο ΔΔΑ, όπου η ισότητα απορρέει από την Σχέση (1.46). Ο τρόπος που τοποθετούμε τις αξίες του συνόλου κόμβων V σε ένα ΔΔΑ έχει μεγάλη σημασία για το χρόνο εύρεσης ενός κόμβου με ζητούμενη αξία. Για παράδειγμα το ΔΔΑ στο Σχήμα 1.25 έχει ΑΧΑ $\frac{22}{8} >$ από τον ΑΧΑ $\frac{21}{8}$ του Σχήματος 1.26. Διαισθητικά, ο λόγος είναι επειδή το ΔΔΑ του Σχήματος 1.26 τοποθετεί την πλειονότητα των κόμβων του σε μικρό βάθος από την ρίζα του e.

Ορισμός Προβλήματος. Έστω σε αύξουσα σειρά οι αξίες $V = \{c_1, \ldots, c_n\}$ όπου οι αντίστοιχες φορές που αναζητούμε κάθε κόμβο είναι $\Pr[1], \ldots, \Pr[n]$. Να κατασκευαστεί ΔΔΑ που να ελαχιστοποιεί τον ΑΧΑ: $\sum_{i \in V} \Pr[i] comp(i)$.

Διαισθητικά, λόγω της Σχέσης (1.47), ένα βέλτιστο ΔΔΑ τοποθετεί τις αξίες που αναζητούμε πολλές φορές (έχουν μεγάλο $\Pr[\cdot]$) σε κόμβους κοντά στη ρίζα (σε μικρό βάθος), ενώ οι αξίες που λίγες φορές αναζητούνται (έχουν μικρή $\Pr[\cdot]$) τοποθετούνται σε κόμβους μακριά από την ρίζα (σε μεγαλύτερο βάθος), δείτε Σχήμα 1.27.

1.10.2 Κατασκευή της Αναδρομής

Έστω οι κόμβοι $V = \{1, \ldots, n\}$ με αντίστοιχες αξίες $c_1 \leq \ldots \leq c_n$ σε αύξουσα σειρά, και αντίστοιχα πλήθη αναζητήσεων $\Pr[1], \ldots, \Pr[n]$.

ΟΡΙΣΜΟΣ **1** *Ορίζουμε ως $\mathcal{O}(n)$ τον ΑΧΑ του βέλτιστου υποδέντρου με κόμβους $\{1, \ldots, n\}$ με αύξουσα σειρά αξιών $\{c_1, \ldots, c_n\}$.*

- (Λάθος τρόπος.) Ο σοφός μας λέει ότι ο κόμβος k, όπου $1 \leq k \leq n$, είναι η ρίζα του βέλτιστου ΔΔΑ. Τότε, το βέλτιστο κόστος ΑΧΑ για το δέντρο με κόμβους $\{1, \ldots, n\}$ που το έχουμε συμβολίσει ως $\mathcal{O}(n)$ (δείτε Ορισμό 1) γράφεται ως άθροισμα από τα παρακάτω τρία επιμέρους κόστη:

  1. Το κόστος 1 που πληρώνουμε για κάθε μία σύγκριση από τις $s(1, n) = \sum_{i=1}^{n} \Pr[i]$ συνολικά συγκρίσεις με την αξία v_k της ρίζας k του ΔΔΑ. Αυτό μας δίνει συνολικό κόστος συγκρίσεων με την ρίζα ίσο με $1 \times s(1, n)$.
  2. Το κόστος ΑΧΑ $\mathcal{O}(k-1)$ του *βέλτιστου αριστερού* υποδέντρου της ρίζας k αποτελούμενο από τους κόμβους $\{1, \ldots, k-1\}$.
  3. Το κόστος ΑΧΑ $\mathcal{O}(n)$ (!!!) του *βέλτιστου δεξιού* υποδέντρου της ρίζας k αποτελούμενο από τους κόμβους $\{k+1, \ldots, n\}$.

Το λάθος εδώ είναι ότι ο Ορισμός 1 δεν μας δίνει την δυνατότητα να εκφράσουμε το αντίστοιχο κόστος για την άνω περίπτωση 3. Διότι ο Ορισμός 1 θεωρεί ΑΧΑ $\mathcal{O}(j)$ για βέλτιστα υποδέντρα με κόμβους $\{1, \ldots, j\}$. Δηλαδή, για υποδέντρα που έχουν *σταθερό* τον αριστερότερο κόμβο 1 και *μεταβλητό* μόνο τον δεξιότερο κόμβο j. Συνεπώς, πρέπει να θεωρήσουμε βέλτιστα υποδέντρα με κόμβους όπου είναι - επίσης - *μεταβλητός* ο αριστερότερος κόμβος τους. Καταλήγουμε έτσι στην τροποποίηση του Ορισμού 1 ως εξής:

ΟΡΙΣΜΟΣ **2** *Ορίζουμε ως $\mathcal{O}(i, j)$ τον ΑΧΑ του βέλτιστου δέντρου με κόμβους $\{i, \ldots, j\}$ με αύξουσα σειρά αξιών $\{c_i, \ldots, c_j\}$.*

Με την χρήση του Ορισμού 2 μπορούμε να κατασκευάσουμε την αναδρομή ως εξής:

- (Σωστός τρόπος.) Ο σοφός μας λέει ότι ο κόμβος k, με $1 \leq k \leq n$, είναι η ρίζα του βέλτιστου ΔΔΑ. Τότε, το βέλτιστο κόστος ΑΧΑ για το δέντρο με κόμβους $\{1, \ldots, n\}$ που το έχουμε συμβολίσει ως $\mathcal{O}(1, n)$ (δείτε Ορ. 2) γράφεται ως άθροισμα από τα παρακάτω τρία επιμέρους κόστη:

  1. (Όμοια με πριν.) Το κόστος 1 που πληρώνουμε για κάθε μία σύγκριση από τις $s(1, n) = \sum_{i=1}^{n} \Pr[i]$ συνολικά συγκρίσεις με την αξία v_k της ρίζας k του δέντρου με κόμβους $\{1, \ldots, n\}$. Αυτό μας δίνει συνολικό κόστος συγκρίσεων με την ρίζα ίσο με $1 \times s(1, n)$.
  2. Το κόστος ΑΧΑ $\mathcal{O}(1, k-1)$ του *βέλτιστου αριστερού* υποδέντρου της ρίζας k αποτελούμενο από τους κόμβους $\{1, \ldots, k-1\}$.
  3. Το κόστος ΑΞΑ $\mathcal{O}(k+1, n)$ του *βέλτιστου δεξιού* υποδέντρου της ρίζας k αποτελούμενο από τους κόμβους $\{k+1, \ldots, n\}$.

Με τη συμβουλή του σοφού ότι η ρίζα του δέντρου $\{1, \ldots, n\}$ είναι ο κόμβος k, όπου $1 \leq k \leq n$, καταλήγουμε στην Αναδρομή:

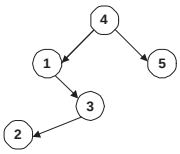
$$\mathcal{O}(1, n) = s(1, n) + \mathcal{O}(1, k-1) + \mathcal{O}(k+1, n) \tag{1.48}$$

Πρέπει να βρούμε τη γενική αναδρομή, ανεξαρτήτως της συμβουλής του σοφού. Είναι λογικό να επιλέξουμε ως ρίζα του δέντρου $\{1,\ldots,n\}$ τον κόμβο k που *ελαχιστοποιεί* το ΑΧΑ ως προς όλες τις δυνατές επιλογές κόμβου $k \in \{1,\ldots,n\}$ ως ρίζα του δέντρου $\{1,\ldots,n\}$:

$$\mathcal{O}(1,n) = s(1,n) + \min_{1\le k\le n}\{\mathcal{O}(1,k-1) + \mathcal{O}(k+1,n)\} \tag{1.49}$$

Για γενικό υποδέντρο με κόμβους $\{i,\ldots,j\}$ η Αναδρομή (1.49) γράφεται:

$$\begin{aligned}\mathcal{O}(i,j) &= s(i,j) + \min_{i\le k\le j}\{\mathcal{O}(i,k-1) + \mathcal{O}(k+1,j)\}\\ &\text{όπου } s(i,j) = \sum_{t=i}^{j} \Pr[t] \times comp(t)\end{aligned} \tag{1.50}$$



Σχήμα 1.27: Έχουμε $\Pr[1] = 30, \Pr[2] = 5, \Pr[3] = 8, \Pr[4] = 45, \Pr[5] = 12$. Η ρίζα είναι το πιο πιθανό στοιχείο, και παιδιά της είναι τα αμέσως πιο πιθανά στοιχεία. Φύλλο είναι το πιο σπάνιο στοιχείο.

1.10.3 Απομνημόνευση της Αναδρομής

Υπολογίζουμε τον Πίνακα $S_{\langle n\times n\rangle}$ (δείτε Πίνακα 1.1 του Παραδείγματος 1.10.4) σύμφωνα με το τύπο για κάθε $s(i,j)$ της Σχέσης (1.50)

```
for(i = 1; i <= n; i++)
    for(j = i; j <= n; j++)
        for(t = i; t <= j; t++)
            S[i,j] = S[i,j] + Pr[t];
```

Η βέλτιστη αξία υπολογίζεται με χρήση της αναδρομής (1.50) με τον αλγόριθμο (δείτε περιπτώσεις 1-5 στο Παράδειγμα 1.10.4):

```
for(t = 0; t <= n-1; t++)
    for(i = 1; i <= n-t; i++)
        j = i + t;
        M[i,j] = S[i,j] + min_{i<=k<=j}{M[i,k-1], M[k+1,j]};
return(M[1,n]);
```

1.10.4 Παράδειγμα

Αρχικά υπολογίζουμε τον Πίνακα 1.1 με τον τύπο στην Σχέση (1.50).
Στην συνέχεια, με την Αναδρομή (1.50) υπολογίζουμε το ΑΧΑ $\mathcal{O}(i,j)$ για κάθε βέλτιστο υποδέντρο, ως εξής:

$$S = \begin{pmatrix} 30 & 35 & 43 & 88 & 100 \\ & 5 & 13 & 58 & 70 \\ & & 8 & 53 & 65 \\ & & & 45 & 57 \\ & & & & 12 \end{pmatrix}$$

Πίνακας 1.1: Ο πίνακας S έχει στοιχεία τους αριθμούς $s(i,j), i,j \in \{1,\ldots,5\}$, σύμφωνα με τον τύπο την Σχέση (1.50). Τα στοιχεία $s(i,i), i = 1,\ldots,5$, της κύριας διαγωνίου είναι ίσα με τα πλήθη αναζήτησης των αξιών κάθε κόμβου: $\mathrm{Pr}[1],\ldots,\mathrm{Pr}[5]$. Εμφανίζονται αριθμοί μόνο πάνω από την κύρια διαγώνιο διότι κάθε $s(i,j)$ στην Σχέση (1.50) έχει νόημα όταν $1 \le i \le j \le 5$.

1. Θεωρούμε όλα τα υποδέντρα με διαδοχικούς κόμβους $\{i,\ldots,j\}$ όπου για τους ακραίους κόμβους i,j ισχύει $\mathbf{i-j=0}$. Παρατηρούμε ότι
$$\forall i = 1,\ldots,5 \text{ ισχύει: } \mathcal{O}(i,i) = \mathrm{Pr}[i] \tag{1.51}$$

2. Θεωρούμε όλα τα υποδέντρα με διαδοχικούς κόμβους $\{i,\ldots,j\}$ όπου για τους ακραίους κόμβους i,j ισχύει $\mathbf{i-j=1}$.
$$\begin{aligned} \mathcal{O}(1,2) &= s(1,2) + \min\{\mathcal{O}(1,0)+\mathcal{O}(2,2), \mathcal{O}(1,1)+\mathcal{O}(3,2)\} = 35 + \min\{5,30\} = 40 \\ \mathcal{O}(2,3) &= 18 \\ \mathcal{O}(3,4) &= 61 \\ \mathcal{O}(4,5) &= 69 \end{aligned} \tag{1.52}$$

3. Θεωρούμε όλα τα υποδέντρα με διαδοχικούς κόμβους $\{i,\ldots,j\}$ όπου για τους ακραίους κόμβους i,j ισχύει $\mathbf{i-j=2}$.
$$\begin{aligned} \mathcal{O}(1,3) &= s(1,3) + \min\{\mathcal{O}(1,0)+\mathcal{O}(2,3), \mathcal{O}(1,1)+\mathcal{O}(3,3), \mathcal{O}(1,2)+\mathcal{O}(4,3)\} \\ &= 43 + \min\{18,38,40\} = 61 \\ \mathcal{O}(2,4) &= s(2,4) + \min\{\mathcal{O}(2,1)+\mathcal{O}(3,4), \mathcal{O}(2,2)+\mathcal{O}(4,4), \mathcal{O}(2,3)+\mathcal{O}(5,4)\} \\ &= 58 + \min\{61,50,18\} = 76 \\ \mathcal{O}(3,5) &= s(3,5) + \min\{\mathcal{O}(3,2)+\mathcal{O}(4,5), \mathcal{O}(3,3)+\mathcal{O}(5,5), \mathcal{O}(3,4)+\mathcal{O}(6,5)\} \\ &= 65 + \min\{69,20,61\} = 85 \end{aligned}$$

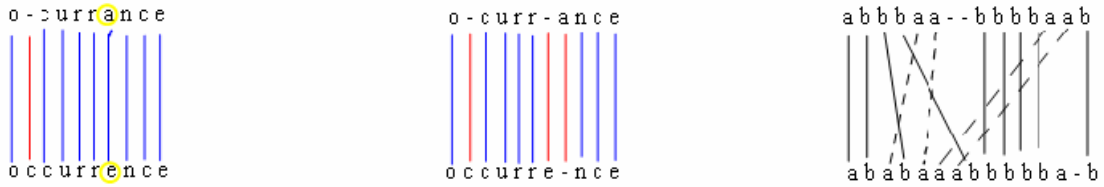
4. Θεωρούμε όλα τα υποδέντρα με διαδοχικούς κόμβους $\{i,\ldots,j\}$ όπου για τους ακραίους κόμβους i,j ισχύει $\mathbf{i-j=3}$.
$$\begin{aligned} \mathcal{O}(1,4) &= s(1,4) + \min\{\mathcal{O}(1,0)+\mathcal{O}(2,4), \mathcal{O}(1,1)+\mathcal{O}(3,4), \mathcal{O}(1,2)+\mathcal{O}(4,4), \mathcal{O}(1,3)+\mathcal{O}(5,4)\} \\ &= 88 + \min\{76,91,85,61\} = 149 \\ \mathcal{O}(2,5) &= s(2,5) + \min\{\mathcal{O}(2,1)+\mathcal{O}(3,5), \mathcal{O}(2,2)+\mathcal{O}(4,5), \mathcal{O}(2,3)+\mathcal{O}(5,5), \mathcal{O}(2,4)+\mathcal{O}(6,5)\} \\ &= 70 + \min\{85,74,30,76\} = 100 \end{aligned}$$

5. Θεωρούμε όλα τα υποδέντρα με διαδοχικούς κόμβους $\{i,\ldots,j\}$ όπου για τους ακραίους κόμβους i,j ισχύει $\mathbf{i-j=4}$.
$$\begin{aligned} \mathcal{O}(1,5) &= s(1,5) + \\ &\quad \min\{\mathcal{O}(1,0)+\mathcal{O}(2,5), \mathcal{O}(1,1)+\mathcal{O}(3,5), \mathcal{O}(1,2)+\mathcal{O}(4,5), \mathcal{O}(1,3)+\mathcal{O}(5,5), \mathcal{O}(1,4)+\mathcal{O}(6,5)\} \\ &= 100 + \min\{100,115,109,73,149\} = 173 \end{aligned}$$

Τελικά ο ΑΧΑ του βέλτιστου ΔΔΑ είναι 173, και το βέλτιστο δέντρο είναι στο Σχήμα 1.27.

1.11 Ταίριασμα λέξεων (Sequence alignment)

1.11.1 Περιγραφή του Προβλήματος



Σχήμα 1.28: **Αριστερά:** οι 9 μπλε γραμμές δηλώνουν *μονότονο ταίριασμα* γραμμάτων. Η 1 κόκκινη δηλώνει 1 κενό (μη ταίριασμα γράμματος). Η 1 μπλε γραμμή με κίτρινα άκρα δηλώνει ταίριασμα διαφορετικών γραμμάτων. **Κέντρο:** οι 8 μπλε γραμμές δηλώνουν *μονότονο ταίριασμα* γραμμάτων. Οι 3 κόκκινες δηλώνουν 3 κενά. **Δεξιά:** οι διακεκομμένες γραμμές δηλώνουν ότι το ταίριασμα δεν είναι *μονότονο*.

Θα μελετήσουμε ένα πρόβλημα με πολλές εφαρμογές στην *Βιοπληροφορική* (Bioinformatics). Μας δίνουν δύο λέξεις L_1, L_2 με αντίστοιχα μήκη n και m, οι οποίες έχουν σχηματιστεί με γράμματα ενός αλφαβήτου Σ. Μας ζητάνε να εκτιμήσουμε σε ποιο βαθμό οι λέξεις L_1, L_2 είναι *όμοιες*. Διαισθητικά, αν οι L_1, L_2 αντιπροσωπεύουν δύο μόρια, τότε ο βαθμός ομοιότητας τους αποτελεί σημαντική ένδειξη για γενετική συγγένεια των μορίων, προδιάθεση για ασθένειες, κτλ.
Μια άλλη εφαρμογή, που πρακτικά μας είναι ίσως πιο προσιτή, είναι σε ηλεκτρονικά *Λεξικά* και *Επεξεργαστές Κειμένου*. Πχ. κατά λάθος πληκτρολογούμε τη λέξη `ocurrance`, ενώ, θα θέλαμε να είχαμε γράψει `occurrence`. Άμεσα «διαβάζει» την σκέψη μας η ηλεκτρονική εφαρμογή, και μας προτείνει την λέξη `occurrence` που είχαμε στο μυαλό μας. Η εφαρμογή το επιτυγχάνει αυτό, εκτιμώντας το βαθμό ομοιότητας της λάθος λέξης που πληκτρολογήσαμε με τις δυνατές σωστές λέξεις.
Έχει καθιερωθεί η τεχνική του *μονότονου ταιριάσματος* για την εκτίμηση του βαθμού ομοιότητας δύο οποιοδήποτε λέξεων L_1 και L_2, δείτε Σχήμα 1.28 (αριστερά & κέντρο). Η τεχνική *μονότονου ταιριάσματος* βασίζεται στους εξής κανόνες.

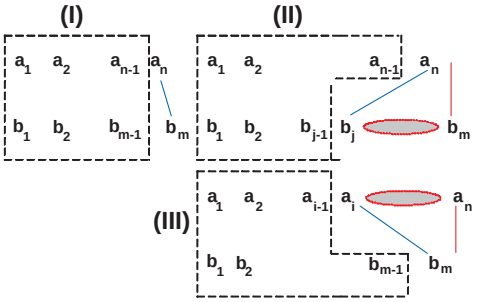
1. Κάθε γράμμα $a_i, i = 1, \ldots, n$, της λέξης L_1 το ταιριάζουμε με ≤ 1 γράμμα $b_j, j = 1, \ldots, m$, της λέξης L_2.
2. Ως μπλε ευθύγραμμο τμήμα $(i, j), i \in \{1, \ldots, n\}, i \in \{1, \ldots, m\}$ φαίνεται κάθε ταίριασμα γράμματος $a_i \in L_1$ με γράμμα $b_j \in L_2$, δείτε Σχ. 1.28 (αριστερά & κέντρο).
3. Κάθε ένα γράμμα που δεν το ταιριάζουμε με άλλο γράμμα το παριστάνουμε με κόκκινο ευθύγραμμο τμήμα, με το ένα άκρο στο εν λόγω γράμμα και το άλλο άκρο σε *κενό* χαρακτήρα, δείτε Σχ. 1.28 (αριστερά & κέντρο).
4. Δεν υπάρχουν διασταυρούμενα ταιριάσματα, όπως στο Σχ. 1.28 (δεξιά).
5. Κάθε ταίριασμα (μπλε ευθ. τμ.) από γράμμα $a_i \in L_1$ σε γράμμα $b_j \in L_2$ έχει κόστος $\alpha_{a_i,b_j} \geq 0$, όπου $\alpha_{a_i,b_j} = 0$ όταν $a_i \equiv b_j$.
6. Κάθε κενό (κόκκινο ευθ. τμ.) έχει κόστος $\delta > 0$.

Οι τιμές για τα κόστη ταιριάσματος $\alpha_{x,y}$ για κάθε ζεύγος λέξεων x, y του αλφαβήτου Σ είναι σταθερές και τις θεωρούμε ως είσοδο, όπως επίσης η τιμή δ για το κόστος του κενού. Παρατηρούμε ότι το μονότονο ταίριασμα στο Σχ. 1.28 (αριστερά) είναι προτιμότερο από το αντίστοιχο ταίριασμα στο Σχ. 1.28 (κέντρο) αν και μόνο αν ισχύει

$$\delta + \alpha_{a,e} < 3\delta$$

Ορισμός Προβλήματος. Μας δίνουν ζεύγος λέξεων L_1 και L_2 σε ένα αλφάβητο Σ, το κόστος $\alpha_{x,y}, \forall x, y \in \Sigma^*$, και το κόστος δ όταν ταιριάζουμε ένα γράμμα με το κενό. Πρέπει να υπολογίσουμε ένα *μονότονο ταίριασμα* των γραμμάτων των λέξεων L_1 και L_2 με το ελάχιστο κόστος.

1.11.2 Κατασκευή της Αναδρομής



Σχήμα 1.29: **(I):** Έστω ότι ταιριάζουμε το γράμμα $a_n \in L_1$ με το $b_m \in L_2$. Προκύπτει το υποπρόβλημα $a_1 \ldots, a_{n-1} \subseteq L_1$ και $b_1 \ldots, b_{m-1} \subseteq L_2$. Το κόστος $\mathcal{O}(n, m)$ του αρχικού προβλήματος είναι ίσο με το κόστος $\mathcal{O}(n-1, m-1)$ του υποπροβλήματος συν το κόστος (μπλε) ταιριάσματος α_{a_n, b_m}.
(II): Το γράμμα $a_n \in L_1$ ταιριάζει με γράμμα $b_j \in L_2$, που είναι $\neq b_m$. Τότε τα γράμματα $b_{j+1}, \ldots, b_m \in L_2$ δεν είναι δυνατόν να ταιριάξουν με άλλο γράμμα της L_1 (αλλιώς, δημιουργούνται *διασχίζοντα* ταιριάσματα).
(III): Το γράμμα $b_m \in L_2$ ταιριάζει με γράμμα $a_i \in L_1$, που είναι $\neq a_n$. Τότε τα γράμματα $a_{i+1}, \ldots, a_n \in L_2$ δεν είναι δυνατόν να ταιριάξουν με άλλο γράμμα της L_2 (αλλιώς, δημιουργούνται *διασχίζοντα* ταιριάσματα).
Συμπέρασμα: Τα a_n, b_m είτε ταιριάζουν μεταξύ τους ή δεν μπορούν αμοιβαία να ταιριάξουν με άλλα γράμματα. Δηλαδή, είτε ταιριάζουμε το a_n με το b_m, ή ταιριάζουμε το a_n με το *κενό*, ή ταιριάζουμε το b_m με το *κενό*.

Έστω ότι μας δίνουν τις λέξεις $L_1 \equiv a_1 \ldots a_n$ και $L_2 \equiv b_1 \ldots b_m$.

ΟΡΙΣΜΟΣ **3** *Ορίζουμε ως $\mathcal{O}(i, j)$ όπου $1 \leq i \leq n, 1 \leq j \leq m$, το κόστος του βέλτιστου μονότονου ταιριάσματος των υπολέξεων $a_1 \ldots a_i \subseteq L_1$ και $b_1 \ldots b_j \subseteq L_2$*

Ο στόχος μας είναι να υπολογίσουμε το κόστος $\mathcal{O}(n, m)$ του βέλτιστου μονότονου ταιριάσματος. Όσον αφορά το βέλτιστο μονότονο ταίριασμα, υπάρχουν 2 δυνατότητες για τα γράμματα a_n και b_m ως προς το πως θα εμφανίζονται σε αυτό. Η πρώτη δυνατότητα είναι ότι τα γράμματα a_n και b_m θα είναι ταιριασμένα μεταξύ τους (μπλε γραμμή) στο βέλτιστο μονότονο ταίριασμα. Η δεύτερη δυνατότητα είναι ότι τα γράμματα a_n και b_m ΔΕΝ θα είναι ταιριασμένα μεταξύ τους στο βέλτιστο μονότονο ταίριασμα. Αναλύουμε αυτές τις 2 δυνατότητες με την βοήθεια ενός σοφού.

- (Σωστός τρόπος.) Ο σοφός μας λέει ότι στο βέλτιστο μονότονο ταίριασμα το γράμμα a_n είναι ταιριασμένο (μπλε τμήμα) με το b_m. Τότε, πληρώνουμε κόστος για αυτό το ταίριασμα ίσο με α_{a_n,b_m} (το οποίο είναι 0 αν $a_n \equiv b_m$, αλλιώς > 0). Αφού το γράμμα a_n ξέρουμε ότι ταιριάζει με το b_m, απομένει να βρούμε το βέλτιστο ταίριασμα των υπόλοιπων γραμμάτων, δηλαδή, στις υπολέξεις $a_1 \dots a_{n-1} \subseteq L_1$ και $b_1 \dots b_{m-1} \subseteq L_2$ (δείτε Σχ. 1.29-**I**). Συνεπώς, από Ορ. 3 το κόστος $\mathcal{O}(n,m)$ του βέλτιστου μονότονου ταιριάσματος των λέξεων $a_1 \dots a_n$ και $b_1 \dots b_m$ μπορεί να γραφεί:

$$\mathcal{O}(n,m) = \alpha_{a_n,b_m} + \mathcal{O}(n-1,m-1) \tag{1.53}$$

- (Λάθος τρόπος.) Ο σοφός μας λέει ότι στο βέλτιστο μονότονο ταίριασμα το γράμμα a_n ΔΕΝ είναι ταιριασμένο με το b_m. Αφού αυτά τα 2 γράμματα δεν είναι ταιριασμένα, τότε δεν πληρώνουμε κόστος για το ταίριασμα τους. Δηλαδή, η κατάσταση είναι όπως στο Σχ. 1.29-**I**, αλλά χωρίς το μπλε τμήμα a_n, b_m. Συνεπώς, από Ορ. 3 το κόστος $\mathcal{O}(n,m)$ του βέλτιστου μονότονου ταιριάσματος των λέξεων $a_1 \dots a_n$ και $b_1 \dots b_m$ μπορεί να γραφεί:

$$\mathcal{O}(n,m) = \mathcal{O}(n-1,m-1) \tag{1.54}$$

- (Σωστός τρόπος.) Οι περιπτώσεις (**II**) & (**III**) του Σχ. 1.29 μας βοηθούν να αντιληφθούμε το λάθος στον προηγούμενο συλλογισμό μας. Γίνεται φανερό ότι η συμβουλή του σοφού «*στο βέλτιστο μονότονο ταίριασμα το γράμμα a_n ΔΕΝ είναι ταιριασμένο με το b_m*» σημαίνει: *είτε* το γράμμα a_n συνδέεται με κενό (με κόκκινο τμήμα, αλλιώς προκύπτει μη μονότονο ταίριασμα), *είτε* το γράμμα b_n συνδέεται με κενό (με κόκκινο τμήμα, αλλιώς προκύπτει μη μονότονο ταίριασμα). Συνεπώς, η συμβουλή του σοφού είναι ισοδύναμη με τις παρακάτω επιπλέον συμβουλές.

  - Αν το γράμμα a_n συνδέεται με κενό τότε πληρώνουμε το κόστος κενού δ για το γράμμα a_n. Τα υπόλοιπα γράμματα της λέξης $L_1 \setminus \{a_n\}$ ορίζουν το υποπρόβλημα $a_1 \dots a_{n-1} \subseteq L_1$. Σε αυτή την περίπτωση, δεν γνωρίζουμε τίποτα άλλο για το γράμμα b_m της λέξης L_2 (δηλαδή, το b_m μπορεί να ανήκει ή όχι στο βέλτιστο μονότονο ταίριασμα). Συνεπώς, τα γράμματα της λέξης L_2 ορίζουν το υποπρόβλημα $b_1 \dots b_m \equiv L_2$. Συνεπώς, από Ορ. 3 το κόστος $\mathcal{O}(n,m)$ του βέλτιστου μονότονου ταιριάσματος των λέξεων $a_1 \dots a_n$ και $b_1 \dots b_m$ μπορεί να γραφεί:

$$\mathcal{O}(n,m) = \delta + \mathcal{O}(n-1,m) \tag{1.55}$$

  - Αν το γράμμα b_m συνδέεται με κενό τότε πληρώνουμε το κόστος κενού δ για το γράμμα b_m. Τα υπόλοιπα γράμματα της λέξης $L_2 \setminus \{b_m\}$ ορίζουν το υποπρόβλημα $b_1 \dots b_{m-1} \subseteq L_2$. Σε αυτή την περίπτωση, δεν γνωρίζουμε τίποτα άλλο για το γράμμα a_n της λέξης L_1 (δηλαδή, το a_n μπορεί να ανήκει ή όχι στο βέλτιστο μονότονο ταίριασμα). Συνεπώς, τα γράμματα της λέξης L_1 ορίζουν το υποπρόβλημα $a_1 \dots a_n \equiv L_1$. Συνεπώς, από Ορ. 3 το κόστος $\mathcal{O}(n,m)$ του βέλτιστου μονότονου ταιριάσματος των λέξεων $a_1 \dots a_n$ και $b_1 \dots b_m$ μπορεί να γραφεί:

$$\mathcal{O}(n,m) = \delta + \mathcal{O}(n,m-1) \tag{1.56}$$

Από τα παραπάνω αντιλαμβανόμαστε ότι η τελική αναδρομή, ως λογικό επακόλουθο των 3 δυνατοτήτων που προκύπτουν από τις συμβουλές του σοφού, είναι αυτή που μας δίνει το *μικρότερο* κόστος από τις δυνατές Σχέσεις (1.56), (1.55), (1.53). Η τελική αναδρομή είναι:

$$\mathcal{O}(n,m) = \min\{\alpha_{a_n,b_m} + \mathcal{O}(n-1,m-1), \delta + \mathcal{O}(n,m-1), \delta + \mathcal{O}(n-1,m)\} \tag{1.57}$$

1.11.3 Απομνημόνευση της Αναδρομής

Στις παρακάτω αρχικές τιμές $M[i,0] = i \times \delta, i = 0,\dots,n$ (καθώς και $M[0,j] = j \times \delta, j = 0,\dots,m$, αντίστοιχα) λαμβάνουμε υπόψη ότι αν η υπολέξη της L_2 (της L_1, αντίστοιχα) είναι μηδενικού μήκους τότε προφανώς το κόστος ταιριάσματος με μια οποιαδήποτε υπολέξη της L_1 (της L_2, αντίστοιχα) μήκους i (ή j, αντίστοιχα) είναι ίσο με το κόστος ταιριάσματος i (ή j, αντίστοιχα) φορές με το κενό.

```
for(i = 0; i <= n; i++)
    M[i, 0] = i × δ;
for(j = 0; j <= m; j++)
    M[0, j] = j × δ;
```

Η βέλτιστη αξία υπολογίζεται με χρήση της αναδρομής (1.57).

```
for(i = 1; i <= n; i++)
    for(j = 1; j <= m; j++)
        M[i, j] = min{α_{a_i,b_j} + M[i − 1, j − 1], δ + M[i, j − 1], δ + M[i − 1, j]};
                      (I)                            (II)              (III)
return(M[1, n]);
```

Για την κατανόηση του αλγορίθμου δείτε το Παράδειγμα 1.11.4. Ειδικότερα, στον άνω αλγόριθμο όταν υπολογίζουμε αναδρομικά την τιμή του $M[i, j]$ και:

- Το min είναι ίσο με τη περίπτωση (I) τότε στο Σχ. 1.30(δεξιά) έχουμε *διαγώνια* ακμή με *πέρας* την κορυφή (i, j) και *αρχή* την κορυφή $(i - 1, j - 1)$.

- Το min είναι ίσο με τη περίπτωση (II), τότε στο Σχ. 1.30(δεξιά) έχουμε *κάθετη* ακμή με *πέρας* την κορυφή (i, j) και *αρχή* την κορυφή $(i, j - 1)$.

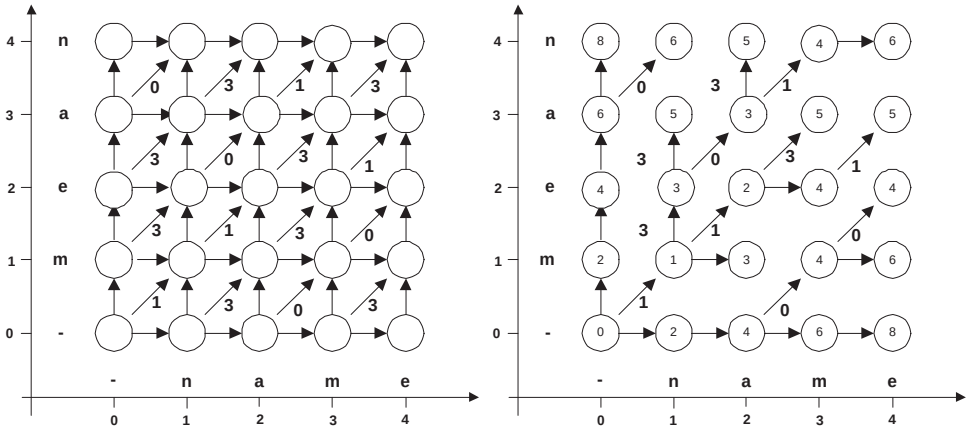
- Το min είναι ίσο με τη περίπτωση (III), τότε στο Σχ. 1.30(δεξιά) έχουμε *οριζόντια* ακμή με *πέρας* την κορυφή (i, j) και *αρχή* την κορυφή $(i - 1, j)$.

Από τα άνω, αντιλαμβανόμαστε ότι ο αλγόριθμος καθώς υπολογίζει αναδρομικά την τιμή $M[i, j]$, επιλέγοντας το μικρότερο κόστος με βάση τις προηγούμενες 3 τιμές $M[i - 1, j - 1], M[i - 1, j], M[i, j - 1]$, αυτό είναι ισοδύναμο με την επιλογή μιας ακμής στο γράφημα του Σχ. 1.30(αριστερά). Αυτές τις επιλεγμένες ακμές από τον αλγόριθμο θα δώσουν ένα μονοπάτι από την αρχή $(0, 0)$ του Σχ. και τέλος την κορυφή (i, j) στο γράφημα του Σχ. 1.30(δεξιά). Προφανώς, το μονοπάτι αυτό θα έχει το μικρότερο μήκος (shortest path), αλλιώς το κόστος της διαδρομής ως το πέρας $M[i, j]$ δεν θα είναι το *βέλτιστο*.

Οι άνω 3 περιπτώσεις, και ο τρόπος που εμφανίζονται οι ακμές γίνονται πιο κατανοητά με τους αναλυτικούς υπολογισμούς στις Σχέσεις (1.58) και (1.59) του συντομότερου μονοπατιού (shortest path) στο γράφημα του Σχ. 1.30 με αρχή την κορυφή $(0, 0)$ και πέρας την (i, j).

1.11.4 Παράδειγμα

Υπολογίζουμε το κόστος του βέλτιστου μονότονου ταιριάσματος για τις λέξεις $L_1 =$ name και $L_2 =$ naem. Μας δίνουν ότι: το κόστος ταιριάσματος με το κενό είναι $\delta = 2$, το κόστος ταιριάσματος σύμφωνου με συμφώνου είναι 1, το κόστος ταιριάσματος φωνήεν με φωνήεν είναι 1, το κόστος ταιριάσματος σύμφωνου



Σχήμα 1.30: **Αριστερά**: Έστω η λέξη $L_1 =$ **name** στον οριζόντιο άξονα και η λέξη $L_2 =$ **naem** στον αντίστοιχο κατακόρυφο. Η κορυφή με συντεταγμένες (i, j) αντιστοιχεί σε ταίριασμα του γράμματος a_i με συντεταγμένη $(i, 0)$ της L_1 με ταίριασμα του γράμματος b_j με συντεταγμένη $(0, j)$ της L_2. Κάθε οριζόντια ακμή με αρχή την κορυφή (i, j) συμβολίζει ότι το i-γράμμα της οριζόντιας λέξης L_1 ταίριαξε με το κενό. Κάθε κάθετη ακμή με αρχή την κορυφή (i, j) συμβολίζει ότι το αντίστοιχο j-γράμμα της κάθετης λέξης L_2 ταίριαξε με το κενό. Συνεπώς, κάθε μία από τις οριζόντιες/κάθετες ακμές έχει κόστος δ. Κάθε διαγώνια ακμή με τέλος την κορυφή με συντεταγμένες (i, j) (συμβολίζει ταίριασμα του γράμματος $(i, 0)$ της L_1 με το γράμμα $(0, j)$ της L_2) έχει κόστος α_{a_i,b_j}. **Δεξιά**: Το κόστος του βέλτιστου μονότονου ταιριάσματος $\mathcal{O}(4, 4)$ για τις λέξεις L_1 και L_2 είναι ίσο με το κόστος του *μικρότερου μονοπατιού* (shortest path) με αρχή στην κορυφή με συντεταγμένες $(0, 0)$ και τέλος στην κορυφή με συντεταγμένες $(4, 4)$.

με φωνήεν είναι 3. Το Σχ. 1.30 βοηθάει την κατανόηση των βημάτων του αλγορίθμου.

$$
\begin{aligned}
M[1,1] &= \min\{\alpha_{a_1,b_1} + M[1-1,1-1], \delta + M[1,1-1], \delta + M[1-1,1]\} = \min\{1+0, 2+2, 2+2\} \\
&= 1, \text{ Δεξιό Σχ. 1.30, διαγώνια ακμή προς κορυφή } (1,1). \\
M[1,2] &= \min\{\alpha_{a_1,b_2} + M[1-1,2-1], \delta + M[1,2-1], \delta + M[1-1,2]\} = \min\{3+2, 2+1, 2+4\} \\
&= 3, \text{ Δεξιό Σχ. 1.30, κατακόρ. ακμή προς κορυφή } (1,2). \\
M[1,3] &= \min\{\alpha_{a_1,b_3} + M[1-1,3-1], \delta + M[1,3-1], \delta + M[1-1,3]\} = \min\{3+4, 2+3, 2+6\} \\
&= 5, \text{ Δεξιό Σχ. 1.30, κατακόρ. ακμή προς κορυφή } (1,3). \\
M[1,4] &= \min\{\alpha_{a_1,b_4} + M[1-1,4-1], \delta + M[1,4-1], \delta + M[1-1,4]\} = \min\{0+6, 2+5, 2+8\} \\
&= 6, \text{ Δεξιό Σχ. 1.30, κατακόρ. ακμή προς κορυφή } (1,4). \\
M[2,1] &= \min\{\alpha_{a_2,b_1} + M[2-1,1-1], \delta + M[2,1-1], \delta + M[2-1,1]\} = \min\{3+2, 2+4, 2+1\} \\
&= 3, \text{ Δεξιό Σχ. 1.30, ορίζ. ακμή προς κορυφή } (2,1). \\
M[2,2] &= \min\{\alpha_{a_2,b_2} + M[2-1,2-1], \delta + M[2,2-1], \delta + M[2-1,2]\} = \min\{1+1, 2+3, 2+3\} \\
&= 2, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (2,2). \\
M[2,3] &= \min\{\alpha_{a_2,b_3} + M[2-1,3-1], \delta + M[2,3-1], \delta + M[2-1,3]\} = \min\{0+3, 2+2, 2+5\} \\
&= 3, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (2,3). \\
M[2,4] &= \min\{\alpha_{a_2,b_4} + M[2-1,4-1], \delta + M[2,4-1], \delta + M[2-1,4]\} = \min\{0+3, 2+2, 2+5\} \\
&= 5, \text{ Δεξιό Σχ. 1.30, κατακόρ. ακμή προς κορυφή } (2,4). \\
M[3,1] &= \min\{\alpha_{a_3,b_1} + M[3-1,1-1], \delta + M[3,1-1], \delta + M[3-1,1]\} = \min\{0+4, 2+6, 2+3\} \\
&= 4, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (3,1). \\
M[3,2] &= \min\{\alpha_{a_3,b_2} + M[3-1,2-1], \delta + M[3,2-1], \delta + M[3-1,2]\} = \min\{0+4, 2+6, 2+3\} \\
&= 4, \text{ Δεξιό Σχ. 1.30, ορίζ. ακμή προς κορυφή } (3,2). \\
M[3,3] &= \min\{\alpha_{a_3,b_3} + M[3-1,3-1], \delta + M[3,3-1], \delta + M[3-1,3]\} = \min\{0+4, 2+6, 2+3\} \\
&= 5, \text{ Δεξιό Σχ. 1.30, διάγ. (ή ορίζ.) ακμή προς κορυφή } (3,3). \\
M[3,4] &= \min\{\alpha_{a_3,b_4} + M[3-1,4-1], \delta + M[3,4-1], \delta + M[3-1,4]\} = \min\{1+3, 2+5, 2+5\} \\
&= 4, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (3,4). \qquad (1.58)
\end{aligned}
$$

$$
\begin{aligned}
M[4,1] &= \min\{\alpha_{a_4,b_1} + M[4-1,1-1], \delta + M[4,1-1], \delta + M[4-1,1]\} = \min\{3+6, 2+8, 2+4\} \\
&= 6, \text{ Δεξιό Σχ. 1.30, ορίζ. ακμή προς κορυφή } (4,1). \\
M[4,2] &= \min\{\alpha_{a_4,b_2} + M[4-1,2-1], \delta + M[4,2-1], \delta + M[4-1,2]\} = \min\{0+4, 2+6, 2+4\} \\
&= 4, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (4,2). \\
M[4,3] &= \min\{\alpha_{a_4,b_3} + M[4-1,3-1], \delta + M[4,3-1], \delta + M[4-1,3]\} = \min\{1+4, 2+4, 2+5\} \\
&= 5, \text{ Δεξιό Σχ. 1.30, διάγ. ακμή προς κορυφή } (4,3). \\
M[4,4] &= \min\{\alpha_{a_4,b_4} + M[4-1,4-1], \delta + M[4,4-1], \delta + M[4-1,4]\} = \min\{3+5, 2+5, 2+4\} \\
&= 6, \text{ Δεξιό Σχ. 1.30, οριζ. ακμή προς κορυφή } (4,4). \qquad (1.59)
\end{aligned}
$$

1.12 Μονοπάτι ελάχιστου κόστους (Shortest paths.) under construction!!!

1.12.1 Περιγραφή του Προβλήματος

1.12.2 Κατασκευή της Αναδρομής

1.12.3 Απομνημόνευση της Αναδρομής

1.13 Ένα πρόβλημα Περιοδεύοντος Πωλητή (A Traveling Salesman problem) under construction!!!

Όπως και στην προηγούμενη ενότητα, θα μελετήσουμε ένα πρόβλημα όπου το εκάστοτε πρόβλημα συνδέεται με αναδρομική σχέση με πολ/μου πλήθους υπο-προβλήματα.

1.13.1 Περιγραφή του Προβλήματος

Μας δίνουν n πόλεις σε ένα χάρτη και τις αμοιβαίες αποστάσεις τους. Ένας πωλητής πρέπει να επισκεφθεί όλες τις πόλεις, περνώντας 1 φορά από κάθε πόλη, διανύοντας την *ελάχιστη* συνολική διαδρομή. Αν ήθελε απλώς να επισκεφθεί όλες τις πόλεις από 1 φορά το πρόβλημα επιλύεται σε πολυωνυμικό χρόνο (μονοπάτι Hamilton). Επειδή όμως θέλει να διανύσει τη μικρότερη συνολική διαδρομή, ζητούμε το μονοπάτι μικρότερου μήκους από όλα τα Hamilton μονοπάτια (είναι εκθετικά πολλά). Όπως θα δούμε σε επόμενο κεφάλαιο, δεν έχει βρεθεί αλγόριθμος επίλυσης του προβλήματος σε πολυωνυμικό χρόνο. Ακριβέστερα, θα μάθουμε αργότερα ότι το πρόβλημα αυτό είναι πλήρες στην κλάση NP. Αυτό κάνει αδρές τις ελπίδες για επίλυση του προβλήματος με αλγόριθμο δυναμικού προγραμματισμού σε πολυωνυμικό χρόνο.

Αν περιορίσουμε τον πωλητή ώστε να διανύει διαδρομές όπου η x συνιστώσα των πόλεων που επισκέπτεται να μεταβάλλεται μονότονα τότε βρίσκουμε εφικτό αλγόριθμο δυναμικού προγραμματισμού. Πιο συγκεκριμένα, θα μελετήσουμε μονοπάτια όπου ο πωλητής (περνώντας 1 φορά από όλες τις πόλεις) θα ξεκινάει από την δεξιότερη πόλη, πηγαίνοντας ολοένα αριστερά φθάνει στην αριστερότερη πόλη και μετά πηγαίνοντας ολοένα δεξιά επιστρέφει στην δεξιότερη πόλη. Στην (1.60) έχουμε διατάξει τις πόλεις ως προς αύξουσα σειρά των x-συντεταγμένων τους (δηλαδή $x_1 \leq x_2 \leq \ldots \leq x_n$):

$$p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n) \qquad (1.60)$$

Στόχος μας είναι η εύρεση του μονοπατιού από το $p_n \to p_1$, ώστε η x συνιστώσα να φθίνει συνεχώς, και του μονοπατιού από το $p_1 \to p_n$, ώστε η x συνιστώσα να αυξάνει συνεχώς, περνώντας 1 φορά από όλες τις πόλεις και με το μικρότερο συνολικό μήκος. Η απόσταση μεταξύ οποιοδήποτε πόλεων p_i, p_j είναι δοσμένη και ίση με $d_{i,j}$.

1.13.2 Κατασκευή της Αναδρομής

Ο πωλητής αρχικά είναι στην δεξιότερη πόλη p_n έτοιμος να τολμήσει το μεγάλο ταξίδι προς την πόλη p_1 και να επιστρέψει. Έχει φοβερό δίλημμα ποια θα είναι η αμέσως επόμενη πόλη που θα επισκεφτεί. Όπως σε όλα τα προηγούμενα παραδείγματα που είδαμε, όταν έχουμε διλήμματα ζητάμε την συμβουλή του σοφού.
• Έστω ότι ο σοφός (που ξέρει τη βέλτιστη διαδρομή) λέει στο πωλητή να πάει από την p_n αρχικά στη πόλη p_i. Αυτό σημαίνει ότι το μήκος $\mathcal{O}(n,n)$ της βέλτιστης διαδρομής συμπεριλαμβάνει το μήκος μεταξύ των πόλεων p_n, p_i:

$$d_{n,i} \tag{1.61}$$

Τονίζουμε εδώ ότι δεν επιτρέπεται στον πωλητή να αυξήσει (πισωγυρίζοντας) την x συνιστώσα του (πρέπει να περνά διαδοχικά από πόλεις με ολοένα μικρότερη x συντεταγμένη) μέχρι να φθάσει στην πόλη p_1. Άρα όλες οι δεξιότερες πόλεις $p_{i+1}, \ldots, p_n$ της p_i *υποχρεωτικά* θα ανήκουν στο μονοπάτι της επιστροφής. Αυτό σημαίνει ότι το μήκος $\mathcal{O}(n,n)$ της βέλτιστης διαδρομής συμπεριλαμβάνει τα μήκη του μονοπατιού που περνά με τη σειρά τις πόλεις $p_{i+1}, \ldots, p_n$ που ισούται με

$$\sum_{j=i+1}^{n-1} d_{j,j+1} \tag{1.62}$$

Συνεπώς, αφού πλέον γνωρίζουμε το μονοπάτι που ανήκουν τα σημεία $p_{i+1}, \ldots, p_n$ μπορούμε να τα αγνοήσουμε.

Οι κρίσιμες παρατηρήσεις για την εξέλιξη του ταξιδιού είναι:

1. ο πωλητής *ξέρει* ότι βρίσκεται στην πόλη p_i.
2. επίσης *ξέρει* ότι από την πόλη p_{i+1} *υποχρεωτικά* αρχίζει το υπο-μονοπάτι της προς δεξιά επιστροφής του: $p_{i+1}, p_{i+2}, \ldots, p_n$ στην p_n.
3. *δεν ξέρει* ποια είναι η βέλτιστη διαδρομή από τη τωρινή πόλη p_i όλο αριστερά προς την p_1 και μετά όλο δεξιά προς την p_{i+1} (όπου αρχίζουν τα "ψιχουλάκια" που δείχνουν το δρόμο προς την p_n).

Αντιλαμβάνεται ότι το αρχικό του πρόβλημα (εύρεσης του μήκους $\mathcal{O}(n,n)$ μικρότερης διαδρομής που αρχίζει από τη πόλη p_n πάει όλο αριστερά στη p_1 και μετά όλο δεξιά στη p_n) έχει πλέον *αναχθεί* στην εύρεση του μήκους

$$\mathcal{O}(i, i+1) \tag{1.63}$$

μικρότερης διαδρομής που αρχίζει από τη πόλη p_i πάει όλο αριστερά στη p_1 και μετά όλο δεξιά στη p_{i+1}. Προσθέτοντας τα μήκη των (1.62), (1.61), (1.63) με τη σοφή συμβουλή λαμβάνουμε συμβολικά:

$$\mathcal{O}(n,n) = \mathcal{O}(i, i+1) + d_{n,i} + \sum_{j=i+1}^{n-1} d_{j,j+1} \tag{1.64}$$

Το μειονέκτημα της παραπάνω αναδρομής (1.64) είναι ότι προήλθε με την ουτοπική βοήθεια του σοφού. Όπως και στα προηγούμενα παραδείγματα, εύκολα και λογικά, μπορούμε να απεξαρτήσουμε την αναδρομή (1.64) από τη σοφή υπόδειξη της πόλης p_i.

Αρκεί να παρατηρήσουμε ότι οι δυνατές σοφές συμβουλές που υπάρχουν είναι όσες και οι δυνατές τιμές του $i = 1, \ldots, n$, δηλαδή $O(n)$. Λαμβάνουμε έτσι $O(n)$ διαφορετικά υποπροβλήματα, ένα για κάθε τιμή του i, με αντίστοιχα κόστη:

$$\left\{ \mathcal{O}(1, 1+1) + d_{n,1} + \sum_{j=1+1}^{n-1} d_{j,j+1} \right\}, \left\{ \mathcal{O}(2, 2+1) + d_{n,2} + \sum_{j=2+1}^{n-1} d_{j,j+1} \right\}, \ldots \tag{1.65}$$

Επειδή ο σκοπός μας είναι να ελαχιστοποιήσουμε το συνολικό μήκος πρέπει να επιλέξουμε εκείνο το $1 \leq i \leq n$ που θα μας δώσει εκείνο το υποπρόβλημα από τα (1.65) με το *μικρότερο* μήκος. Συμβολικά:

$$\mathcal{O}(n,n) = \min_{1 \leq i \leq n} \left\{ \mathcal{O}(i,i+1) + d_{n,i} + \sum_{j=i+1}^{n-1} d_{j,j+1} \right\} \tag{1.66}$$

Η ζητούμενη αναδρομή είναι η (1.66).

1.13.3 Απομνημόνευση της Αναδρομής