Κατ'οίκον Εργασία 2 – Σκελετοί Λύσεων

1. (α) Αλγόριθμος:

   Δημιούργησε το σύνολο P που αποτελείται από τα άκρα όλων των ευθυγράμμων τμημάτων.
   Βρες το κυρτό περίβλημα του P με τον αλγόριθμο του Graham.

   Ορθότητα:

   Έστω K το πολύγωνο που επιστρέφεται από τον αλγόριθμο. Το K είναι το ελάχιστο κυρτό πολύγωνο το οποίο περιέχει τα σημεία του P στο εσωτερικό ή το περίβλημα του. Ισχύει επίσης ότι κάθε σημείο των ευθυγράμμων τμημάτων βρίσκεται στο εσωτερικό ή το περίβλημα του K: Από τον ορισμό ενός κυρτού πολυγώνου, αν δύο σημεία p_1 και p_2 βρίσκονται στο εσωτερικό ή το περίβλημα του πολυγώνου τότε κάθε σημείο στο ευθύγραμμο τμήμα που τα ενώνει βρίσκεται στο εσωτερικό ή το περίβλημα του πολυγώνου. Επομένως το ζητούμενο έπεται.

   (β) Αλγόριθμος:
   Εφάρμοσε τον αλγόριθμο του Grahahm αφήνοντας πίσω το βήμα της ταξινόμησης των σημείων.

   (γ) Αλγόριθμος
   i = 0
   Εφόσον S≠∅
   Εφάρμοσε τον αλγόριθμο του Jarvis για να υπολογίσεις το κυρτό περίβλημα του S έστω K
   Για κάθε k∈ K
   θέσε βάθος(k)=i
   S=S-k
   i++

   Στην πρώτη εκτέλεση του αλγορίθμου του Jarvis υπολογίζουμε το κυρτό περίβλημα του αρχικού σημειοσυνόλου, και σε κάθε επόμενη εκτέλεση το κυρτό περίβλημα των σημείων που δεν ανήκουν σε κάποιο από τα προηγούμενα περιβλήματα. Υποθέτοντας ότι συνολικά δημιουργούνται m περιβλήματα και το i-οστό περίβλημα περιέχει h_i στοιχεία, τότε $h_1+\dots h_m = n$ και επιπλέον, η i-οστή εκτέλεση του αλγορίθμου του Jarvis απαιτεί χρόνο $O(h_i n)$ ενώ συνολικά ο αλγόριθμος απαιτεί χρόνο $O(h_1 n+\dots h_m n) = O(n(h_1+\dots h_m)) = (O(n^2)$.

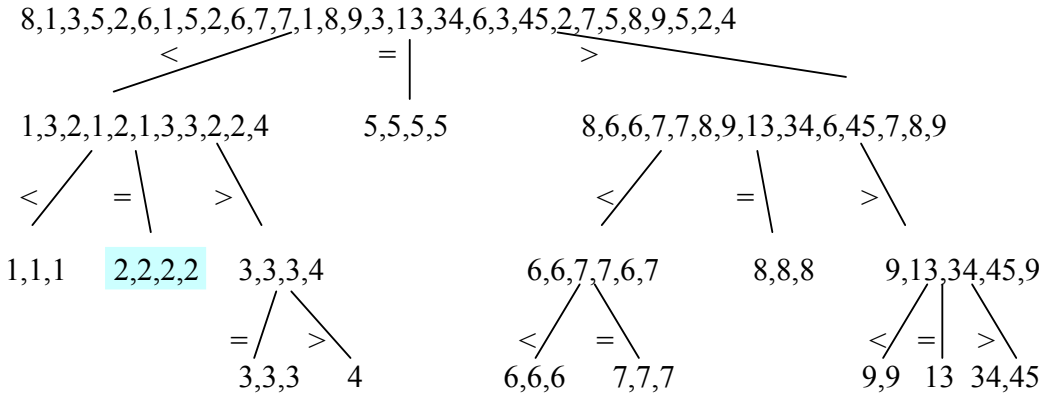
2. Ο αλγόριθμος ακολουθεί τις ίδιες γραμμές με τον διαίρει και βασίλευε αλγόριθμο closest pairs με αλλαγή του ορισμού της απόστασης από την ευκλειδική απόσταση στην απόσταση L_∞. Μόνη διαφορά είναι ότι τώρα κατά τη φάση συνδύασε, για κάθε σημείο στον τομέα L, χρειάζεται να ελέγξουμε μόνο τα 4 επόμενά του σημεία.

3. (α) Ταξινόμησε τη λίστα και εκτέλεσε μια διερεύνηση στη λίστα από τα αριστερά προς τα δεξιά συγκρίνοντας κάθε στοιχείο με το επόμενό του. Αν κανένα από τα στοιχεία της λίστας δεν είναι ίσο με το επόμενό του τότε απάντησε πως ο συρμός της λίστας είναι ίσος με 1, διαφορετικά απάντησε πως υπάρχουν στοιχεία στη λίστα με πολλαπλότητα μεγαλύτερη από 1.
Χρονική πολυπλοκότητα: O(n lg n)

(β) Παρατηρούμε ότι αν υπάρχει στοιχείο στη λίστα με πολλαπλότητα μεγαλύτερη του n/2 τότε το στοιχείο αυτό θα πρέπει οπωσδήποτε να καταλαμβάνει τη μεσαία θέση στην ταξινομημένη μορφή της λίστας. Επομένως ο αλγόριθμος έχει ως εξής:
Βρες το μεσαίο στοιχείο της λίστας (χρησιμοποιώντας τον αλγόριθμο επιλογής των Blum, Floyd, Pratt, Rivest και Tarjan) έστω x. Με μια διερεύνηση στη λίστα υπολόγισε την πολλαπλότητα του x. Αν η πολλαπλότητα του x είναι μεγαλύτερη του n/2 τότε απάντησε πως η πολλαπλότητα του συρμού της λίστας είναι μεγαλύτερη του n/2 διαφορετικά πως κάθε στοιχείο της λίστας έχει πολλαπλότητα μικρότερη του n/2.
Χρονική πολυπλοκότητα: O(n)

(γ) Ο Αλγόριθμος
Βρες το μεσαίο στοιχείο της λίστας και μετά κάλεσε τη διαδικασία partition με pivot το μεσαίο στοιχείο, έτσι ώστε η λίστα να χωριστεί σε τρεις υποπίνακες: τον L1 με στοιχεία μικρότερα του μεσαίου στοιχείου, τον L2 με στοιχεία ίσα με το μεσαίο στοιχείο και τον L3 με στοιχεία μεγαλύτερα του μεσαίου στοιχείου. Παρατηρούμε ότι τα L1 και L2 πρέπει να έχουν τουλάχιστον (n-1)/2 στοιχεία, αφού περιέχουν όλα τα στοιχεία που είναι μικρότερα ή ίσα του μεσαίου στοιχείου. Επαναλαμβάνουμε αναδρομικά τη διαδικασία στον πίνακα L1 και L3 μέχρι να έχουμε O(n/k) υποπίνακες. Κάθε ένας από τους υποπίνακες τύπου L2 αποτελείται αποκλειστικά από επαναλήψεις ενός στοιχείου. Αυτά τα στοιχεία είναι υποψήφιοι συρμοί. Οι υπόλοιποι υποπίνακες τύπου L1 και L3 έχουν μέγεθος μικρότερο από k. Υπολόγισε τις πολλαπλότητες των υποψήφιων συρμών και επέστρεψε το στοιχείο με τη μέγιστη πολλαπλότητα.

Παράδειγμα: Βρες συρμό με πολλαπλότητα τουλάχιστον 3 στον πίνακα



Ορθότητα: Κάθε αναδρομικό βήμα έχει ως αποτέλεσμα την εύρεση όλων των εμφανίσεων ενός στοιχείου και τοποθέτησή τους σε ένα πίνακα. Αφού όλοι οι πίνακες τύπου L1 και L3 έχουν μήκος μικρότερο του k στο τέλος της διαδικασίας είναι αρκετό να ελέγξουμε όλους τους πίνακες τύπου L2 για πιθανούς συρμούς.

Χρόνος εκτέλεσης: Αν το δεδομένο εισόδου περιέχει λιγότερα από k στοιχεία τότε σε σταθερό χρόνο μπορούμε να συμπεράνουμε ότι δεν υπάρχει στοιχείο με πολλαπλότητα μεγαλύτερη ή ίση του k. Αν $n \geq k$ τότε ο αλγόριθμος κάνει δύο αναδρομικές κλήσεις του εαυτού του σε υποπίνακες μεγέθους μικρότερου του n/2. Επομένως, ο χρόνος εκτέλεσης δίνεται από την πιο κάτω αναδρομική διαδικασία:

$T(n) = 1$ αν $n<k$
$T(n) \leq 2T(n/2) + \Theta(n)$ αν $n \geq k$

Επιλύνοντας την εξίσωση έχουμε $T(n) \in O(n \lg(n/k))$.

4. Ο χρόνος εκτέλεσης του αλγορίθμου επιλογής για μέγεθος υποπινάκων 3, 7 και 9 δίνεται αντίστοιχα από τις πιο κάτω αναδρομικές εξισώσεις

$T_1(n) = T_1(n/3) + T_1(2n/3+4) + n$
$T_2(n) = T_2(n/5) + T_2(7n/10+6) + n$
$T_3(n) = T_3(n/7) + T_3(5n/7+8) + n$

Αναλύοντας τις εξισώσεις συμπεραίνουμε ότι ενώ $T_2(n)$, $T_3(n) \in O(n)$, $T_1(n) \in \Omega(n)$ και ακόμη $T_1(n) \in \Omega(n \lg n)$. Ακολουθεί η απόδειξη $T_2(n) \in O(n)$.

Θα δείξουμε ότι $T_2(n) \in O(n)$, δηλαδή υπάρχει σταθερά c και n_0 τέτοια ώστε για κάθε $n \geq n_0$ $T_2(n) \leq cn$.

$T_2(n) \leq cn/5 + 7cn/10 + 6c + n$
$\leq 9cn/10 + 6c + n = cn - (cn/10 - 6c - n)$

Επομένως $T_2(n) \leq cn$ αν $cn/10 - 6c - n < 0$. Η ανισότητα ισχύει για κάθε n και c=1, και συμπεραίνουμε ότι $T_2(n) \leq n$ για κάθε $n \geq 1$.

5. (α) Έστω T(m) ο χρόνος εκτέλεσης της διαδικασίας Cheapest(i) όπου n-i = m, δηλαδή όταν η διαδικασία καλείται από τη στήλη n-m για να πετύχει μετακίνηση m στήλες προς τα δεξιά. Τότε

$T(0) = 1$
$T(m) = T(m-1) + T(m-2) + T(m-3) + 1$

<u>Λήμμα</u>: $T(m) \in \Omega((3/2)^m)$
Θα δείξουμε ότι για κάθε $n \geq 1$ $T(m) \geq (3/2)^m$.
Προφανώς για m=1 η πρόταση ισχύει ($T(0) = 1 \geq (3/2)^0$)
Υποθέτουμε ότι η πρόταση ισχύει για κάθε $k \leq m$. Τότε
$T(m) \geq (3/2)^{m-1} + (3/2)^{m-2} + (3/2)^{m-3} + 1$

$= (3/2)^{m-3}((3/2)^2+(3/2)+1) + 1$
$= (3/2)^{m-3}(19/4) \geq (3/2)^{m-3}(3/2)^3 = (3/2)^m$
και το ζητούμενο έπεται.

(β) Έστω K(m) το ελάχιστο κόστος για μετακίνηση από τη στήλη m της σχάρας μέχρι τη στήλη n. Το K(m) εκφράζεται αναδρομικά ως εξής:

$$K(m) = \begin{cases} \infty & \text{if } m > n \\ B[m,1] & \text{if } m = n \\ B[m,1] + K(m+1) & \text{if } B[m,2] = 1 \\ B[m,1] + \min(K(m+1), K(m+2)) & \text{if } B[m,2] = 2 \\ B[m,1] + \min(K(m+1), K(m+2), K(m+3)) & \text{if } B[m,2] = 3 \end{cases}$$

Από αυτή την αναδρομική σχέση παρατηρούμε ότι για υπολογισμό κάθε K(m) απαιτείται γνώση κάποιων K(m+i). Η βασική περίπτωση δίνεται από το K(n) και το ζητούμενο είναι το K(1). Επομένως χρησιμοποιούμε ένα πίνακα K μήκους n και υπολογίζουμε τις θέσεις του ξεκινώντας με τα K[n] και προχωρώντας προς τα K[n-1], K[n-1], …, και K[1].

Σε ψευδοκώδικα ο αλγόριθμος έχει ως εξής:

```
Cheapest2(int K[n], int B[n,2])

  K[n]= B[n,1];

  for (m = n – 1 ; m >= 1; m--)
     a = B[m,1] + K[m+1];
     if (B[m,2]>1 και m < n-1 και a < B[m+1]+K[m+2])
          a = B[m+1] + K[m+2]
     if (B[m,2]>2 και m < n-2 και a < B[m+1]+K[m+3])
          a = B[m+1] + K[m+3];

  return K[1];
```

Ο χρόνος εκτέλεσης του αλγορίθμου είναι O(n).

(γ) Για υπολογισμό όχι μόνο του ελάχιστου κέρδους αλλά και των κινήσεων του παιχνιδιού που αντιστοιχούν στο κέρδος αυτό, πρέπει να επεκτείνουμε τον αλγόριθμό μας έτσι ώστε όταν προσδιορίζει τη βέλτιστη πρώτη μετακίνηση από κάθε στήλη και να την σημειώνει. Φυλάγουμε αυτές τις πληροφορίες σε ένα νέο πίνακα A[n] και επεκτείνουμε τον αλγόριθμο ως εξής:

```
Cheapest3(int K[n], int A[n], int B[n,2])

  K[n]= B[n,1];
```

```
    for (m = n – 1 ; m >= 1; m--)
       a = B[m,1] + K[m+1];
       A[m]=1;
       if (B[m,2]>1 και m < n-1 και a < B[m+1]+K[m+2]))
            a = B[m+1] + K[m+2];
            A[m] = 2;
       if (B[m,2]>2 και m < n-2 και a < B[m+1]+K[m+3]))
            a = B[m+1] + K[m+3]);
            A[m] = 3;

    return K[1];
```

Για επιστροφή του βέλτιστου παιχνιδιού χρησιμοποιούμε την πιο κάτω διαδικασία.

```
m=1;
while (m != n)
   printf m;
   m=m+A[m];
printf n;
```

Ο χρόνος εκτέλεσης της διαδικασίας παραμένει στην τάξη O(n).