1η Σειρά γραπτών άσκησεων

Μανός Αραπίδης

ΑΜ: el16071 Εξάμηνο:7ο

1η Άσκηση

1)

Η σειρά κατάταξης είναι:

$\sum_{k=1}^{n} k2^{-k}$ $\frac{\log n!}{(\log n!)^3}$ $n2^{2^{2^{100}}}$ $\log\binom{2n}{n}$

$2^{(\log n)^4}$ $\sqrt{n!}$ $\sum_{k=0}^{n}\binom{n}{k}$ = Θ(n∗ 2^n) $\sum_{k=1}^{n} k2^k$ =Θ(n∗ 2^n)

- $\sum_{k=1}^{n} k2^{-k}$=$\sum_{k=1}^{n} k\frac{1}{2}^k$ = = $\frac{-(n+1)*\frac{1}{2}^{n+1}+n*\frac{1}{2}^{n+2}+\frac{1}{2}}{(\frac{1}{2}-1)^2}$ = Θ($\frac{n}{2^n}$)

Γνωρίζουμε $\log n!$=Θ($n\log n$)

- Οπότε: $\frac{\log n!}{(\log n)^3}$ = Θ($\frac{n}{(\log n)^2}$)
- $n2^{2^{2^{100}}}$ = Θ(n)
- $\log\binom{2n}{n}$ = $\log\frac{(2n)!}{n!*n!}$ = $\log\frac{(2n)!}{n!}\frac{1}{n!}$ = $\log\frac{(2n)!}{n!}$ + $\log\frac{1}{n!}$ ≤ $\log 2^n$ - $\log n!$ = Θ($n\log n$)
- $2^{(\log n)^4}$ = $(2^{\log n})^{(\log n)^3}$=Θ($n^{(\log n)^3}$)

Από τον τύπο του Stirling έχουμε

n! = Θ($\sqrt{2\pi n}(\frac{n}{e})^n$ = Θ($\sqrt{n}\frac{n^n}{e^n}$) = Θ($\frac{2^{0,5\log n+n\log n}}{2^{n\log e}}$) = Θ($2^{0,5\log n+n\log n-n\log e}$)

Οπότε για $\sqrt{n!}$ =Θ($2^{0,25\log n+0,5n^{0.5}\log n-n^{0.5}\log e}$)

- $\sum_{k=0}^{n}\binom{n}{k}$ = Θ(n∗ 2^n) = Θ($2^{n+\log n}$)
- $\sum_{k=1}^{n} k2^k$ =Θ(n∗ 2^n)

2)

1. T(n) = 3T(n/2) + $n^2\log n$

   f(n)= $n^2\log n$ = Ω($n^{\log_2 3}$) = Ω($n^{1.58}$)

   Οπότε απο το Master Theorem προκύπτει:

   T(n) = Θ($n^2\log n$)

2. T(n) = 4T(n/2) + $n^2 \log n$

   f(n)= $n^2 \log n$ = Ω($n^{\log_2 4}$) = Ω(n^2)

   Οπότε απο το Master Theorem προκύπτει:

   T(n) = Θ($n^2 \log n$)

3. T(n) = 5T(n/2) + $n^2 \log n$
   f(n)= $n^2 \log n$ = O($n^{\log_2 5}$) = O($n^{2.3}$)

   Οπότε απο το Master Theorem προκύπτει:

   T(n) = Θ($n^{2,3}$)

4. T(n) = T(n/2) + T(n/3) + n

   $\frac{1}{2} + \frac{1}{3} < 1$

   Οπότε απο την ειδίκη περίπτωση του Master Theorem προκύπτει:

   T(n) = Θ(n)

5. T(n) = T(n/2) + T(n/3) + T(n/6) + n

   Σύμφωνα με την μέθοδο Akra-Bazzi :
   $\sum_{i=0}^{2}(a_i * b_i)^p$ = 1 $\Leftrightarrow \frac{1}{2^p} + \frac{1}{3^p} + \frac{1}{6^p}$=1 => p = 1 (μοναδική λύση)

   T(n) =Θ($n * (1 + \int_1^n \frac{x}{x^{1+1}} dx$)) = Θ($n * (1 + \int_1^n \frac{x}{x^2} dx$)) =
   = Θ($n * (1 + \int_1^n \frac{1}{x} dx$)) = Θ($n + n \log n$) = Θ($n \log n$)

6. T(n) = T(n/4) + $\sqrt{n}$

   f(n) = $\sqrt{n}$ = Ω($n^{\log_4 1}$) = Ω(1)

   Οπότε απο το Master Theorem προκύπτει:

   T(n) = Θ($\sqrt{n}$)

2η Άσκηση

a)

$A_1=[1...n_1]$ $A_2=[1...n_2]$

Έστω ότι οι πίνακες είναι ταξινομοιμένοι σε αύξουσα σειρά.

Έστω ότι τα μέγεθη των πινάκων A_1, A_2 είναι N_1 και N_2 αντοίστιχα.

Ο αλγόριθμος μου για να υπολογίζω την ελάχιστη διαφορά δυο στοιχείων απο διαφορετικούς πίνακες είναι ο εξής:

Έστω ότι ο δείκτες i και j αντιστοιχούν στα στοιχεία των πινάκων A_1, A_2.

Αρχικοποιήσε τους δείκτες να δείχνουν στα πρώτα στοιχεία των πινάκων.

1) Επανέλαβε μέχρι ένας από τους δυο δείκτες να φτάσει στο τέλος του πίνακα.
2) Διάβασε τα στοιχεία που δείχνουν οι δείκτες.
3) Σύγκρινε τα και βρές το μικρότερο.
4) Αφαιρεσε το από το μεγαλύτερο και αποθήκεσε το αποτέλεσμα σε μια πρωσορινή μεταβλητή. Σύγκρινε την με την προηγούμενη τιμη, αν υπάρχει(που είναι αποθηκευμένη σε άλλη μεταβλητή). Αν η νεά τιμή είναι μικρότερη απο την προηγούμενη, ανανεώσε την παλία με την νέα μαζί με πληροφορίες για τους δείκτες.
5) Στον πίνακα με το με το μικρότερο στοιχείο απο τα δύο, αύξησε τον δείκτη κατά ένα. Τον αλλον κράτα τον σταθερό. Στην περίπτωση που οι τιμές των δυο στοιχείων είναι ίδιες αύξησε οποιοδήποτε από τους δυο δείκτες.

Η ορθότητα του αλγορίθμου αποδεικνύεται από

Στην περίπτωση που αποφασίζαμε να προχωράμε στον πίνακα με το μεγαλύτερο στοιχείο,η διαφορά ολο και θα μεγάλωνε. Για αυτό με τον αλγόριθμο που ακολουθήσαμε η διαφορά για το μικρότερο στοιχείο θα είναι πάντα η ελάχιστη δυνατή. Έτσι αποδεικνύεται ότι θα βρούμε την ελάχιστη διαφορά μεταξύ δύο στοιχείων στους πίνακες.

Ο αλγόριθμος είναι γραμικού χρόνου $\Theta(N)$ καθώς τελειώνει όταν έχει διατρέξει ολόκληρο ένα πίνακα. Στην χειρότερη περίπτωση θα χρειαστεί να διατρέξει και τους δύο πίνακες, οπότε θα έχουμε πολυπλοκότητα $O(N_1+N_2)$.

b)

Ο αλγόριθμος για m στοιχεία είναι παρόμοιος με τον παραπάνω και είναι ο εξης:

1) Επανέλαβε μέχρι ένας από τους m δείκτες να φτάσει στο τέλος του πίνακα.
2) Διάβασε τα στοιχεία που δείχνουν οι δείκτες.

3) Σύγκρινε τα και βρές το μικρότερο και το μεγαλύτερο.
4) Αφαιρεσε το από το μεγαλύτερο και αποθήκεσε το αποτέλεσμα σε μια πρωσορινή μεταβλητή. Σύγκρινε την με την προηγούμενη τιμη, αν υπάρχει(που είναι αποθηκευμένη σε άλλη μεταβλητή). Αν η νεά τιμή είναι μικρότερη απο την προηγούμενη, ανανεώσε την παλία με την νέα μαζί με πληροφορίες για τους δείκτες.
5) Στον πίνακα με το με το μικρότερο στοιχείο απο τους m, αύξησε τον δείκτη κατά ένα. Τους άλλους κράτα τους σταθερούς. Στην περίπτωση που οι τιμές των δυο ή περισσοτέρων στοιχείων είναι ίδιες αύξησε οποιοδήποτε από τους δείκτες.

Αντίστοιχα με την πριν, ο αλγόριθμος μας εγγυάται ότι θα βρεί την ελάχιστη διαφορά μετάξυ μέγιστου και ελάχιστου από m στοιχεία διαφορετικών m πινάκων.

Κάθε φορά που θα τρέχει ο αλγόριθμος θα χρειάζεται να βρίσκει το νεό ελάχιστο και μέγιστο από τα m στοιχεία. Για να βρούμε το μέγιστο αρκεί συγκρίνουμε να την προηγούμενη τιμή του μεγίστου με το νέο στοιχείο που πήραμε και να την ανανεώσουμε αν είναι μεγαλύτερη, O(1). Όμως για να βρούμε το νέο μικρότερο στοιχείο θα χρειαστεί να διατρέξουμε ξανά τα m στοιχεία O(m). Στην καλύτερη περίπτωση ο αλγόριθμος θα τρέξει $N_κ$ (κ=1,2..m) φορές οσο το μέγεθος ενός πίνακα, δηλαδή το μικρότερο στοιχέιο θα βρισκεται πάντα στον ίδιο πίνακα Ω(m*$N_κ$). Στην χειρότερη περίπτωση ο αλγόριθμος θα τρέξει N = $\sum_{κ=1}^{m} Nk$ φορές, δηλαδή θα χρειαστεί διαβαστούν και οι m πίνακες. Οπότε προκύπτει πολυπλοκότητα O(m*N).

c)

Η λειτουργία που καθορίζει τον χρόνο εκτέλεσης έιναι η εύρεση του ελάχιστου στοιχείου O(m). Προκειμένου να μειώσουμε τον χρόνο αυτό θα χρησιμοποιήσουμε μια δομή δεδομένων, τα AVL binary trees. Στην πρώτη επανάληψη να θα κάνουμε insert() τα πρώτα στοιχεία από τους m πίνακες. Στη συνέχεια βρίσκουμε το μικρότερο και το μεγαλύτερο στοιχείο, τα οποία βρίσκονται στην κάτω αριστερή και κάτω δεξία θέση του δέντρου αντίστοιχα, σε χρόνο O($\log m$). Τα παραπάνω τα κάνουμε μόνο για την πρώτη επανάληψη και συνεχιζουμε τον αλγόριθμο όπως περιγράφω παρακάτω.

Σε κάθε επανάληψη του αλγορίθμου συγκρίνουμε να την προηγούμενη τιμή του μεγίστου με το νέο στοιχείο που πήραμε και την ανανεώνουμε αν είναι μεγαλύτερη, O(1). Μετά κάνουμε delete() το προηγούμενο ελάχιστο στοιχείο O($\log m$), κανουμε insert() το νέο στοιχείο O($\log m$) και τέλος βρίσκουμε το νέο μικρότερο στοιχείο O($\log m$).

Οπότε με αυτόν τον τρόπο καταφέραμε να μειώσουμε την εύρεση του ελάχιστου στοιχείου σε O($\log m$). Οπότε η πολυπλοκότητα του αλγορίθμου μεώνεται σε O($N * \log m$).

3η Άσκηση

1.

Ο αλγόριθμος στην γενική του μορφή του θα συγκρίνει την κάρτα που μόλις διαβάσαμε με το πάνω στοιχείο την κάθε στοιχείο κάθε στοίβας και όσα από αυτά έχουν μεγαλύτερη τιμή απο αυτήν της νέας κάρτας,θα υπολογίζει την διαφορά τους. Από αυτες τις διαφορές θα επιλέγει την στοίβα με την ελάχιστη τιμή διαφοράς και εκεί θα τοποθετεί την νέα κάρτα.

Ο αλγόριθμος αποδεικνύεται ότι πάντα θα δημιουργεί τον ελάχιστο αριθμό στοιβών σύμφωνα με τα παρακάτω:

Έστω ότι έχουν δημιουργηθεί κ στόιβες με το πάνω στοιχείο της καθεμίας να είναι n_1, n_2, … , n_k, με $n_1<n_2<$ … $< n_k$. Έστω το νέο στοιχείο που διαβάζουμε έχει τιμή h και βρίσκουμε ότι το στοιχείο με την ελάχιστη διαφορά απο το h είναι το $n_{κ-1}$ και το αμέσως επόμενο είναι το $n_κ$, δηλαδή h < $n_{κ-1}$, $n_κ$ και $n_{κ-1}$ < $n_κ$. Οι στοίβες n_{k-1} και n_k είναι οι μόνες στις οποίες μπορούμε να βάλουμε το στοιχείο. Τοποθετώντας το h πάνω από το $n_{κ-1}$ και όχι από το $n_κ$, δίνουμε την δυνατότητα στα στοιχεία που είναι ανάμεσα του $n_{κ-1}$ και του $n_κ$ να μπορούν πάνε πάνω από το $n_κ$ και να μην χρειαστεί να δημιουργηθεί νέα στοίβα.

Προκειμενου ο αλγόριθμος να πετύχει την καλύτερη πολυπλοκότητα θα χρησιμοποιήσω μια δομή δεδομένων, τα AVL binary trees. Στους κόμβους του δέντρου θα έχουμε το πάνω στοιχείο των υπάρχοντων στοιβών, μαζί με πληροφορια για την στοίβα που ανήκει . Αν έχουμε m στοίβες θα έχουμε m κόμβους. Πιο συγκεκρυμένα ο αλγόριθμος θα είναι ο παρακάτω:

- Επανέλαβε μέχρι η αρχική στοίβα να αδείασει.
- Διάβαζε κάθε φορά μία κάρτα από την στοίβα.
- Για την πρώτη επανάληψη, δημιούργησε ένα AVL δέντρο και κάνε insert() το στοιχείο που μόλις διάβασες. Παρέλυψε τα επόμενα βήματα και συνέχισε με την δεύτερη επανάληψη.
- Το νέο στοιχείο που διαβάσαμε το κάνουμε insert() στο δέντρο, O($\log m$). Σε κάθε κόμβο που περνάμε, μέχρι το στοιχείο να βρεί την θέση του στο δέντρο, κάνουμε τα παρακάτω. Αν το στοιχείο του κόμβου είναι μεγαλύτερο από αυτο που κάνμε insert(), τότε υπολογίζουμε την διαφορά τους και κρατάμε πληροφορίες για την στοίβα στην οποία ανήκει. Κάθε φορά που βρίσκουμε διαφορά μικρότερη απο την παλια, ανανεώνουμε την παλιά. Στο τέλος, θα έχουμε την ελάχιστη διαφορά, αρα και την στοίβα που θα τοποθετήσουμε το στοχείο. Άν δεν συναντήσουμε κανένα στοιχείο που να είναι μεγαλύτερο, σημαίνει ότι πρέπει να δημιουργήσουμε μια νέα στοιβα και παραλείπουμε τα επόμενα βήματα του αλγορίθμου.
- Βάζουμε την κάρτα στην στόιβα που βρήκαμε.

- Κάνουμε delete() από το δέντρο το στοιχείο που έιχαμε υπολογίσει την ελάχιστη διαφορά , προκειμένου το δέντρο να έχει κάθε φορά την καινούργια εικόνα για τα στοιχεία τον στοιβών, O($\log m$).

Τον παραπάνω αλγόριθμο θα χρειαστεί να το εκτελέσουμε n φορές, δηλαδή όσες και τα στοιχεία της αρχικής στόιβας. Για μια επανάληψη έχει χρόνο εκτέλεσης O($\log m$), όπου m ο αριθμός των στοιβών. Οπότε συνολικά έχει O($n\log m$).

Στην χειρότερη περίπτωση, όπου τα στοιχεία είναι ήδη ταξινομημένα αλλά σε αύξουσα σειρά, θα δημιουργηθούν n στοίβες. Οπότε η πολυπλοκότητα της χειρότερης περίπτωσης είναι O($n \log n$).

2.

Μετά την ολοκλήρωση του παραπάνω αλγορίθμου θα έχουμε m ταξινομημένες στοίβες και ένα AVL δέντρο γεμάτο με τα πάνω στοιχεία των στοιβών. Ο γενικός αλγόριθμος που μπορούμε να χρησιμοποιήσουμε για να ενώσουμε τις στοίβες σε μια ταξινομημένη είναι ο παρακάτω:

- Επανάλαβε μέχρι όλες οι στοίβες να αδειάσουν.
- Βρίσκουμε το μικρότερο στοιχείο , που βρίσκεται στην κάτω αριστερή θέση, σε O($\log m$). Στην πρώτη επανάληψη δημιουργούμε μια αδεία στοίβα και στη συνέχεια τοποθετούμε όλα τα στοιχεία εκεί.
- Βαλε το στοιχείο στην παραπάνω στόιβα.
- Κάνουμε delete() από το δέντρο, το στοιχείο που βρήκαμε.
- Από την στοίβα που άνηκε το στοιχείο που αφαιρέσαμε από το δέντρο,κάνουμε pop(), πέρνουμε το επόμενο στη σειρά(αν υπάρχει) και το κάνουμε insert() στο δέντρο, O($\log m$). Όποια στοίβα αδειάσει σταματάμε να την χρησιμοποιούμε στην επανάληψη του αλγόριθμου. Αν στο τελος μείνει μια στοίβα την τοποθετούμε αυτούσια στην στοίβα ταξινόμησης.

Τον παραπάνω αλγόριθμο θα χρειαστεί να το εκτελέσουμε n φορές, δηλαδή όσα είναι τα στοιχεία στις αρχικές στόιβες. Για μια επανάληψη έχει χρόνο εκτέλεσης O($\log m$), όπου m ο αριθμός των στοιβών. Οπότε συνολικά έχει O($n\log m$).

Στην χειρότερη περίπτωση, όπου κάθε στοίβα αποτελειται από ένα στοιχείο, θα έχουμε n στοίβες. Οπότε η πολυπλοκότητα της χειρότερης περίπτωσης είναι O($n \log n$).

3.

Έχουμε είσοδο 3,2,4,7,8,1,5,6 ο αλγόριθμος της 1) είναι ο παρακάτω βήμα βήμα:

1. S_1={3}

Tree={3}

2. S_1={3,2}
   Tree={2}

3. S_1={3,2} S_2={4}
   Tree={2,4}

4. S_1={3,2} S_2={4} S_3={7}
   Tree={2,4,7}

5. S_1={3,2} S_2={4} S_3={7} S_4={8}
   Tree={2,4,7,8}

6. S_1={3,2,1} S_2={4} S_3={7} S_4={8}
   Tree={4,7,8,1}

7. S_1={3,2,1} S_2={4} S_3={7,5} S_4={8}
   Tree={4,8,1,5}

8. S_1={3,2,1} S_2={4} S_3={7,5} S_4={8,6}
   Tree={4,1,5,6}

Συνολικά δημιουργήθικαν 4 στοίβες.

Ο αλγόριθμος της 2) είναι ο παρακάτω βήμα βήμα:

1. S_1={3,2,1} S_2={4} S_3={7,5} S_4={8,6}
   Tree={1,4,5,6}, min = 1
   S={}

2. S_1={3,2} S_2={4} S_3={7,5} S_4={8,6}
   Tree={4,5,6,2}, min = 2
   S={1}

3. S_1={3} S_2={4} S_3={7,5} S_4={8,6}
   Tree={4,5,6,3}, min = 3
   S={1,2}

4. S_1={} S_2={4} S_3={7,5} S_4={8,6}
   Tree={4,5,6}, min = 4
   S={1,2,3}

5. S_1={} S_2={} S_3={7,5} S_4={8,6}
   Tree={5,6}, min = 5
   S={1,2,3,4}

6. S_1={} S_2={} S_3={7} S_4={8,6}
   Tree={6,7}, min = 6
   S={1,2,3,4,5}

7. S_1={} S_2={} S_3={7} S_4={8}
   Tree={7,8}, min = 7
   S={1,2,3,4,5,6}

8. S_1={} S_2={} S_3={} S_4={8}
   Tree={8}, root = 8
   S={1,2,3,4,5,6,7}

9. S_1={} S_2={} S_3={} S_4={}
   Tree={}
   S={1,2,3,4,5,6,7,8}

Η μέγιστη υπακολουθία είναι η {2,4,7,8} με μήκος 4.

4.

Έστω ότι η μέγιστη αύξουσα υπακολουθία της εισόδου έχει m στοιχεία. Τα στοιχεία αυτα επείδη βρίσκονται σε αύξουσα σειρά δεν μπορούν να τοποθετηθούν στην ίδια στοίβα. Για αυτό χρειάζονται m διαφορέτικες στοίβες στις οποίες να μπορούμε να βαλουμε όλα τα στοιχεία. Τα στοιχεία μπορούν να τοποθετηθούν σε ήδη υπάρχουσες στοίβες, αν γίνεται ή θα πρεπεί να δημιοργηθούν νέες για όσα δεν γίνεται το προηγούμενο. Οπότε φαίνεται οτι χρειάζονται τουλαχιστον m στοίβες, για την μέγιστη αύξουσα υπακολουθία και ανάλογα με την είσοδο και τον αλγόριθμο μπορεί να πρέπει να δημιουργηθούν και άλλες. Άρα, ο αριθμός που τελικά έχουμε είναι πάντα ίσος ή μεγαλύτερος από το μήκος της μέγιστης υπακολούθιας.

5.

Γνωρίζω ότι το πάνω στοιχείο της κ-οστής στόιβας πρέπει να είναι μεγαλύτερο από τα πάνω στοιχεία των στοιβών αριστερά του. Άρα είναι μεγαλύτερο τουλάχιστον από κ-1 στοιχεία.

Για να λύσουμε το πρόβλημα της μέγιστης αύξουσας ακολουθίας θα κάνουμε τα πρακάτω. Για κάθε στοιχείο που βάζουμε στις στοίβες, θα βρίσκουμε το αμέσος μικρότερο, από τα πάνω στοιχεία των στοιβών και θα βάζουμε ένα δείκτη να

δείχνει σαυτό. Αν το στοιχείο που τοποθετήσαμε είναι το μικρότερο, τότε το βάζουμε να δείχνει στον εύατο του.

Αξιωποιόντας την παρατήρηση για την κ-οστή στοίβα και το γεγονός ότι το πλήθος των στοιβών που δημιουργούνται είναι μεγαλύτερο ή ίσο από μήκος της μέγιστης υπακολουθίας, διαλέγω το στοιχείο της δεξίας στοίβας, δηλαδή το μεγαλύτερο από τις πάνω κάρτες, για να κάνω back tracing τους δείκτες. Έτσι θα βρώ την μέγιστη αύξουσα ακολουθία ανεστραμμένη, οπότε θα χρειαστεί να την αντιστρέψω. Δεν είναι απαραίτητο η ακολουθία που θα βρω να είναι η μοναδική λύση στο πρόβλημα αλλα δεν θα υπάρχει καμία μεγαλύτερη.

Για την μέγιστη φθίνουσα ακολουθία μπορούμε να συμπεράνουμε οτι το μήκος της θα είναι μεγαλύτερο ή ίσο από μήκος της μεγαλύτερης στοίβας. Δηλαδή όλες οι στοίβες έχουν μήκος μικρότερο ή ισο της φθίνουσας. Το ελάχιστο μήκος που μπορεί να έχει μια στόιβα, είναι ίσο με το μήκος της μικρότερης συνεχόμενης φθίνουσας ακόλουθίας. Ενώ το μέγιστο μήκος που μπορεί να έχει, είναι ίσο με της μέγιστης φθίνουσας ακολουθίας, αν εκείνη ήταν συνεχόμενη. Οπότε αποδεικνύεται η αρχική πρόταση.

6.

Η τράπουλα έχει $n * m + 1$ στοιχεία.

Έστω ότι,μετά την ολοκλήρωση του αλγορίθμου (1), έχουν δημιουργηθεί το πολύ n στοίβες. Τότε θα πρέπει να υπάρχει τουλάχιστον μια στοίβα με

$\frac{n*m}{n} + 1 = m + 1$ στοιχεία.

Έστω ότι,μετά την ολοκλήρωση του αλγορίθμου (1), κάθε στοίβα μπορεί να έχει το πολυ m στοιχεία. Τότε θα πρέπει να υπάρχουν τουλάχιστον $\frac{n*m}{m} + 1 = n + 1$ στοίβες.

Οπότε απο τα παραπάνω αποδεικνύεται ότι θα για μια τράπουλα με $n * m + 1$ στοιχεία θα δημιουργηθούν είτε τουλάχιστον $n + 1$ στοίβες είτε τουλάχιστον μια στοίβα θα έχει $m + 1$ στοιχεία.

Το προηγούμενο μπορεί να συνδιαστεί με την Αρχή του Περιστερώνα.

Έστω η ακολουθία :

S = {5, 10, 15, 20, 25, 4, 9, 14, 19, 24, 3, 8, 13, 18, 23, 2, 7, 12, 17, 22, 1, 6, 11, 16, 21}

1) S_1 = {5}
2) S_1 = {5} S_2 = {10}

3) $S_1 = \{5\}$ $S_2 = \{10\}$ $S_3 = \{15\}$

4) $S_1 = \{5\}$ $S_2 = \{10\}$ $S_3 = \{15\}$ $S_4 = \{20\}$

5) $S_1 = \{5\}$ $S_2 = \{10\}$ $S_3 = \{15\}$ $S_4 = \{20\}$ $S_5 = \{25\}$

6) $S_1 = \{5,4\}$ $S_2 = \{10\}$ $S_3 = \{15\}$ $S_4 = \{20\}$ $S_5 = \{25\}$

7) $S_1 = \{5,4\}$ $S_2 = \{10,9\}$ $S_3 = \{15\}$ $S_4 = \{20\}$ $S_5 = \{25\}$

8) $S_1 = \{5,4\}$ $S_2 = \{10,9\}$ $S_3 = \{15,14\}$ $S_4 = \{20\}$ $S_5 = \{25\}$

9) $S_1 = \{5,4\}$ $S_2 = \{10,9\}$ $S_3 = \{15,14\}$ $S_4 = \{20,19\}$ $S_5 = \{25\}$

10) $S_1 = \{5,4\}$ $S_2 = \{10,9\}$ $S_3 = \{15,14\}$ $S_4 = \{20,19\}$ $S_5 = \{25,24\}$

11) $S_1 = \{5,4,3\}$ $S_2 = \{10,9\}$ $S_3 = \{15,14\}$ $S_4 = \{20,19\}$ $S_5 = \{25,24\}$

12) $S_1 = \{5,4,3\}$ $S_2 = \{10,9,8\}$ $S_3 = \{15,14\}$ $S_4 = \{20,19\}$ $S_5 = \{25,24\}$

13) $S_1 = \{5,4,3\}$ $S_2 = \{10,9,8\}$ $S_3 = \{15,14,13\}$ $S_4 = \{20,19\}$
$S_5 = \{25,24\}$

14) $S_1 = \{5,4,3\}$ $S_2 = \{10,9,8\}$ $S_3 = \{15,14,13\}$ $S_4 = \{20,19,18\}$
$S_5 = \{25,24\}$

15) $S_1 = \{5,4,3\}$ $S_2 = \{10,9,8\}$ $S_3 = \{15,14,13\}$ $S_4 = \{20,19,18\}$
$S_5 = \{25,24,23\}$

16) $S_1 = \{5,4,3,2\}$ $S_2 = \{10,9,8\}$ $S_3 = \{15,14,13\}$ $S_4 = \{20,19,18\}$
$S_5 = \{25,24,23\}$

17) $S_1 = \{5,4,3,2\}$ $S_2 = \{10,9,8,7\}$ $S_3 = \{15,14,13\}$ $S_4 = \{20,19,18\}$
$S_5 = \{25,24,23\}$

18) $S_1 = \{5,4,3,2\}$ $S_2 = \{10,9,8,7\}$ $S_3 = \{15,14,13,12\}$ $S_4 = \{20,19,18\}$
$S_5 = \{25,24,23\}$

19) $S_1 = \{5,4,3,2\}$ $S_2 = \{10,9,8,7\}$ $S_3 = \{15,14,13,12\}$ $S_4 = \{20,19,18,17\}$
$S_5 = \{25,24,23\}$

20) $S_1 = \{5,4,3,2\}$ $S_2 = \{10,9,8,7\}$ $S_3 = \{15,14,13,12\}$ $S_4 = \{20,19,18,17\}$
$S_5 = \{25,24,23,22\}$

21) $S_1 = \{5,4,3,2,1\}$ $S_2 = \{10,9,8,7\}$ $S_3 = \{15,14,13,12\}$
$S_4 = \{20,19,18,17\}$ $S_5 = \{25,24,23,22\}$

22) $S_1 = \{5,4,3,2,1\}$ $S_2 = \{10,9,8,7,6\}$ $S_3 = \{15,14,13,12\}$
$S_4 = \{20,19,18,17\}$ $S_5 = \{25,24,23,22\}$

23) $S_1 = \{5,4,3,2,1\}$ $S_2 = \{10,9,8,7,6\}$ $S_3 = \{15,14,13,12,11\}$
$S_4 = \{20,19,18,17\}$ $S_5 = \{25,24,23,22\}$

24) $S_1 = \{5,4,3,2,1\}$ $S_2 = \{10,9,8,7,6\}$ $S_3 = \{15,14,13,12,11\}$
$S_4 = \{20,19,18,17,16\}$ $S_5 = \{25,24,23,22\}$

25) $S_1 = \{5,4,3,2,1\}$ $S_2 = \{10,9,8,7,6\}$ $S_3 = \{15,14,13,12,11\}$
$S_4 = \{20,19,18,17,16\}$ $S_5 = \{25,24,23,22,21\}$

Οπότε, για την ακολουθία S προκύπτουν 5 στοίβες των 5 στοιχείων.

<u>4η Άσκηση</u>

a.

Έχουμε n σωματίδια από κάθε στρατόπεδο.

Προκειμένου να βρούμε την πρώτη χρονική στιγμή της σύγκρουσης θα εφαρμόσουμε τον παρακάτω αλγόριθμο για κάθε σωματίδιο α ,από τα n, μέχρι να βρούμε αυτό για το οποίο έγινε η σύγκρουση.

Κάνουμε δυαδική αναζήτηση στο πεδίο του χρόνου. Η πρώτη συγκρούση θα γίνει στο χρονικό διάστημα [T_{min}, T_{max}], όπου $T_{min} = \frac{L}{2*Vmax}$ και $T_{max} = \frac{L}{2*Vmin}$. Το T_{min} προκύπει αν τα δύο σωματίδια κινούνταν με σταθερή ταχύτητα V_{max} , αντίστοιχα για το $T_{maχ}$ αν κινούνταν σταθερά με V_{min}.

Την σύγκρουση μπορούμε να την αντιληφθούμε αν δεν έχει μεταβληθεί η θέση του σωματιδίου.

Παίρνουμε κάθε φορά, για το διάστημα που εξετάζουμε [t_1, t_2], την μέση τιμή median=$\frac{t2+t1}{2}$. Στην 1η επανάληψη αρχίζουμε από το [T_{min}, T_{max}]. Αν την στιγμή του median δεν έχει παρατηρηθεί σύγκρουση απορρύπτουμε το αριστερό κομμάτι του χρονικού διαστήματος και συνεχίζουμε αναδρομικά στα δεξία του median. Αντίθετα, αν έχει γίνει σύγκρουση τότε απορρύπτουμε το δεξία κομμάτι και συνεχίζουμε αναδρομικά στα αρίστερα. Η αναδρομή τελειώνει όταν φτάσουμε σε ένα διάστημα [t_k, t_{k+1}] με χρονική διάρκεια $\Delta t \leq \tau$, όπου τ η ελάχιστη χρονική διάρκεια, η οποίη μπορεί να ελέχξει ο υπερπολογιστής. Αν στο παραπάνω διάστημα έχει παρατηρηθεί σύγκρουση τότε το επιστρέφουμε και η ακριβής χρονική στιγμή βρίσκεται ενδιάμεσα αυτού με ακρίβεια τ. Αν στο τέλος της αναδρομής για το συγκεκριμένο σωματίδιο δεν έχει παρατηρηθεί σύγκρουση, τότε εκτελουμε τον αλγόριθμο για το επόμενο.

Για κάθε σωματίδιο ο χρόνος εκτέλεσης είναι $O(\log \frac{Tmax-Tmin}{\tau})$ =

= $O(\log \frac{\frac{L}{2*Vmin}-\frac{L}{2*Vmax}}{\tau}) = O(\log(\frac{L}{\tau} * \frac{Vmax-Vmin}{2*Vmax*Vmin}))$. Οποτε στην χειρότερη περίπτωση, που θα χρειαστεί να ελέχξουμε και τα n σωματίδια, η πολυπλοκότητα αλγορίθμου προκύπτει: $O(n*\log(\frac{L}{\tau} * \frac{Vmax-Vmin}{2*Vmax*Vmin}))$.

b)

Για να λύσουμε το ίδιο πρόβλημα για την αλλην πλευρά του στρατοπέδου, που εκτοξεύδει σωματίδια β, θα ακολουθήσουμε τον παρακάτω αλγόριθμο.

Έχουμε απο n σωματίδια α και β.

Για κάθε ένα σωματίδιο β θα το χρησιμοποιήσουμε τον υπερπολογιστή για να βρούμε το σωματίδιο α με το οποίο συγκρούται, καθώς και την χρονική στιγμή που γίνεται αυτό. Το παραπάνω μπορούμε να το βρούμε υποθέτοντας ότι για κάθε ζευγάρι σωματιδίων ο υπερπολογιστής μας επιστρέφει είτε μια χρονική στιγμή είτε ότι δεν συγκρούονται. Για το πρώτο σωματίδιο β θα χρειαστούν να γίνουν το πολυ n έλεγχοι μέχρι να βρούμε το α με το οποίο συγκρούεται. Για το επόμενο σωματίδιο,

δεν συμπεριλαμβάνουμε στους ελέγχους το σωματίδιο α που βρίκαμε στην προηγούμενη επανάληψη του αλγορίθμου. Οπότε για το δεύτερο σωματίδιο χρειάζονται να γίνουν το πολύ n-1 έλεγχοι. Επαγωγικά για το m-οστό σωματίδιο, χρειάζονται το πολύ n-m έλεγχοι. Άρα, συνολικά για τα n σωματίδια β θα χρειαστούν το πολύ $\sum_{i=1}^{n} i = \frac{n*(n+1)}{2}$ = O(n^2) έλεγχοι. Από τις χρονικές στιγμές που θα έχει επιστρέψει ο αλγόριθμος βρίσκουμε την μικρότερη, όπου και είναι αυτή της πρώτης σύγκρουσης, O(n).Κάθε έλεγχος έχει σταθερό χρόνο Θ(1), οπότε η πολυπλοκότητα του αλγορίθμου είναι O($n^2 + n$) = O(n^2).

Αν δε μπορούμε να κάνουμε την παραπάνω υπόθεση για τον υπερπολογιστή, απλά θα χρειαστεί να κανουμε n ελέγχους για κάθε ένα απο τα n σωματίδια β. Αρα θα χρειαστούν παλί O(n^2) έλεγχοι και ο αλγόριθμος προκύπτει με την ίδια πολυπλοκότητα όπως προηγουμένως.

5η Άσκηση

Η θέση του θησαυρού βρίσκεται στη θέση |χ|, δηλαδή είναι είτε -χ είτε χ. Ο ανιχνευτής βρίσκεται αρχικά στην θέση 0.Προκειμένου να βρούμε τον θησαυρό διανύοντας την ελάχιστη απόσταση εκτελούμε τον παρακάτω αλγόριθμο:

- Επανέλαβε μέχρι να βρείς τον θησαυρό.
- Στην πρώτη επανάληψη διάλεξε τυχαία την αρχική κατεύθυνση του ανιχνευτή, πχ δεξιά. Σε κάθε επόμενη επανάληψη κινήσου προς την αντίθετη κατευθυνση.
- Στην πρώτη επανάληψη θα κινηθείς προς την κατεύθυνση που διάλεξες με βήμα 1. Σε κάθε επόμενη επανάληψη θα διανύεις απόσταση με βήμα διπλάσιο απο το προηγούμενο.

Κατά την διάρκεια του αλγορίθμου θα περάσουμε από τις θέσεις:

2^0,-2^1, 2^2, ..., -2^{m-1}, 2^m, ..., |x|= $2^{\log|x|}$

Στην m-οστη επαναληψη, που δεν θα θα έχουμε έχουμε βρεί τον θησαυρο, θα βρίσκομαστε στη θέση |2^m| και θα έχουμε διανήσει απόσταση

S= $2 * 2^0 + 2 * 2^1$+...+ $2 * 2^{m-1} + 2^m = 2 * \sum_{i=0}^{m-1} 2^i + 2^m$

Στην τελευταία επανάληψη, οπου και θα περάσουμε απο τη θέση x, θα έχουμε κάνει $\lceil \log|x| \rceil$ κινήσεις και θα έχουμε διανύσει

S = $2 * \sum_{i=0}^{\lceil \log|x| \rceil} 2^i$ + |x| < $2 * \sum_{i=0}^{\log|x|+1} 2^i$ + |x| = $2 * (2^{(\log|x|+1)+1}$- 1) + |x| =

= $2 * 2^2 * 2^{\log|x|} + |x| - 2 = 8 * |x| + |x| - 2 = 9|x| - 2$ => S <9|x|

Οπότε προκύπτει ότι αλγόριθμος έχει πολυπλοκότητα O(|χ|) και θα χρειαστεί να περπατήσουμε το πολύ 9|χ| βήματα.