Τσώλας Λεωνίδας – 03112422 – ΣΗΜΜΥ ΕΜΠ – 5ο ΕΤΟΣ

Αλγόριθμοι και Πολυπλοκότητα

2η σειρά ασκήσεων

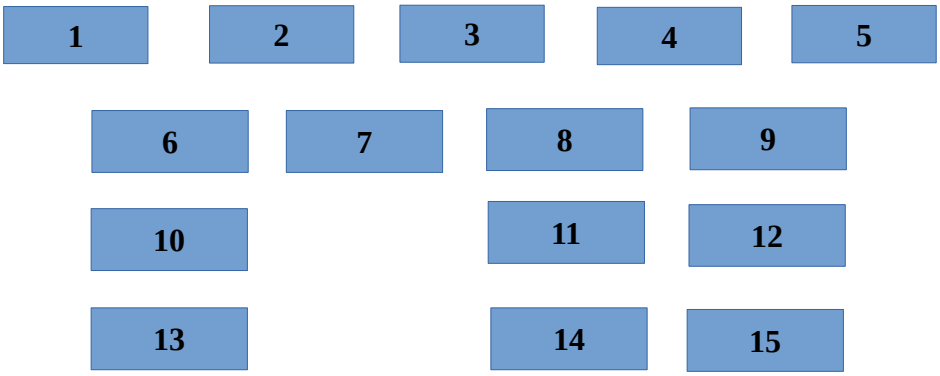
Άσκηση 1

1.α.1.

Θα επιχειρήσω να αποδείξω ότι είναι ψευδείς οι οι τρεις ζητούμενες περιπτώσεις με χρήση αντιπαραδειγμάτων. Εάν βρω αντιπαράδειγμα έχω αποδείξει ότι δεν είναι αληθείς οι αλγόριθμοι στη γενική περίπτωση, χωρίς σαφώς να ισχύει το αντίστροφο.

S1. Λιγότερες επικαλύψεις: Επιλέγουμε το μάθημα με τις λιγότερες επικαλύψεις με άλλα μαθήματα, αφαιρούμε όλα τα μαθήματα που επικαλύπτονται με αυτό, και επαναλαμβάνουμε. **Λάθος**

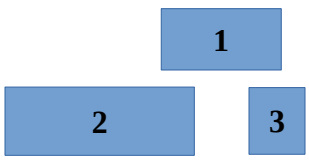
Αντιπαράδειγμα:



Λύση κριτηρίου S1: Διαγράφει τα {2,3,6,10,13,9,12,15,8,11,14}. Μένουν τα {7,1,5,4} →|S1| = 4
Βέλτιση λύση Optimal: {1,2,3,4,5} → |Optimal = 5

S2.Μεγαλύτερη διάρκεια: Αν δεν υπάρχουν επικαλύψεις, επιλέγουμε όλα τα μαθήματα. Διαφορετικά αφαιρούμε το μάθημα με τη μεγαλύτερη διάρκεια και επαναλαμβάνουμε. **Λάθος**

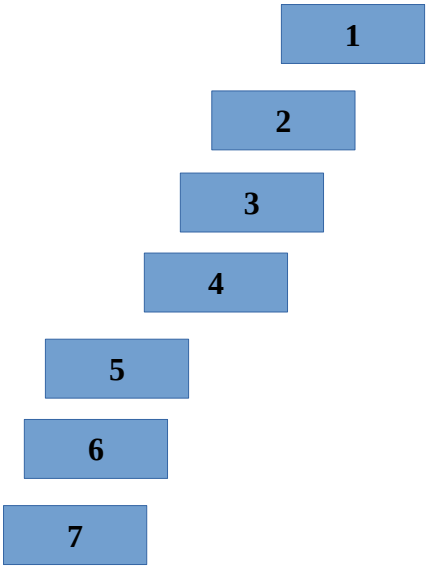
Αντιπαράδειγμα:



Λύση κριτηρίου S2: Διαγράφει τα {2,1}. Μένει το {3} → |S2|=1.
Βέλτιση λύση Optimal: {2,3} → |Optimal| = 2.

S3. Περισσότερες επικαλύψεις: Αν δεν υπάρχουν επικαλύψεις, επιλέγουμε όλα τα μαθήματα. Διαφορετικά, αφαιρούμε το μάθημα που έχει τις περισσότερες επικαλύψεις με άλλα μαθήματα και επαναλαμβάνουμε. **Λάθος**

Αντιπαράδειγμα:



Λύση κριτηρίου S3: Διαγράφει τα {4,3,5,2,6} →Μένουν τα {1,7} → |S3| = 2
Βέλτιση λύση Optimal: {1,4,7} → |Optimal| = 3

1.α.2.

- Ταξινομούμε τα χρονικά διαστήματα σε αύξουσα σειρά με βάση το χρόνο ολοκλήρωσής τους, έστω με σειρά $f_1, f_2, \ldots, f_n$.
- Συμβολίζουμε με $C(i)$ το μέγιστο σύνολο διδακτικών μονάδων μιας βέλτιστης λύσης για τα μαθήματα {1,2,…,i}.
- Προφανώς η ζητούμενη λύση είναι η $C(n)$ και η σχέση αρχικοποιείται στο $C(1)=w_1$.
- Για την κατασκευή της αναδρομικής σχέσης, χρειαζόμαστε ένα πίνακα μήκους n, κάθε στοιχείο $previous(i)$ του οποίου θα περιέχει το δείκτη του μαθήματος που τελειώνει αργότερα και δεν έχει επικάλυψη με το i .
- Η αναδρομική σχέση είναι $C(i)=max[w_i+C(previous(i)), C(i-1)]$.
- Ταξινόμηση διαστημάτων και κατασκευή πίνακα previous: $O(nlogn)$.
- Εύρεση ενός εκ των n στοιχείων του πίνακα της C: $O(1)$.
- Άρα συνολικά έχουμε πολυπλοκότητα $O(nlogn)$.

1.β.

- Ταξινομούμε τα μαθήματα κατ' αύξουσα σειρά χρόνου λήξεως, δηλαδή κατά f_i .
- Λαμβάνουμε το πρώτο στοιχείο, έστω f_{min} , δηλαδή εκείνο το οποίο αντιστοιχεί στο μάθημα που τελειώνει πρώτο.
- Προφανώς πρέπει να περάσουμε από αυτό το μάθημα και να ενημερώσουμε. Το διάστημα στο οποίο μπορεί να γίνει η ενημέρωση για το μάθημα αυτό είναι το [s_{min}, f_{min}).
- Αν περάσουμε από το μάθημα τη χρονική στιγμή s_{min} θα έχουμε κάνει κακή επιλογή, καθώς θα ενημερώσουμε μόνο για αυτό το μάθημα. Αντίθετα, αν περάσουμε πιο αργά υπάρχει το ενδεχόμενο να ενημερώσουμε και μαθητές άλλων μαθημάτων. Σε κάθε περίπτωση είναι μια κίνηση από την οποία έχουμε μόνο να κερδίσουμε (να χρειαστεί να περάσουμε λιγότερες φορές).
- Ο αλγόριθμος είναι ο εξής: Κρατάμε μία λίστα Unvisited με τα μαθήματα που δεν έχουμε επισκεφτεί ακόμα, αρχικά γεμάτη με όλα τα μαθήματα και μια αρχικά κενή λίστα Visited που περιέχει τα μαθήματα που έχουμε επισκεφτεί.
- Όπως ανέφερα ήδη η λίστα Unvisited αρχικοποιείται ταξινομημένη με αύξουσα σειρά ως προς το χρόνο ολοκλήρωσης f_i .
- Επιλέγω να περάσω όσο πιο κοντά γίνεται στο χρόνο που ολοκληρώνεται το πρώτο στοιχείο της λίστας Unvisited και στο σημείο που περνάω διαγράφω όλα τα σημεία που περιλαμβάνουν το χρόνο επίσκεψης. Προφανώς δεν χρειάζεται να ελέγξω όλη τη λίστα για να βρω τα σημεία αυτά, καθώς όταν βρίσκω το πρώτο διάστημα σταματάω, αφού η λίστα είναι ταξινομημένη.
- Διαγράφω τα διαστήματα που περιλαμβάνουν το χρόνο επίσκεψης από τη λίστα Unvisited και τα μεταφέρω στη λίστα Visited.
- Συνεχίζω μέχρι η λίστα Unvisited να είναι κενή ή ισοδύναμα η λίστα Visited να περιέχει όλα τα διαστήματα.
- Η ταξινόμηση μπορεί να θεωρηθεί πως γίνεται με χρήση Mergesort ή άλλης μεθόδου ταξινόμησης πολυπλοκότητας $O(nlogn)$.
- Έπειτα πρέπει να ελέγξουμε το πολύ n σημεία, στην περίπτωση που δεν υπάρχει καμία επικάλυψη μαθημάτων, οπότε πρέπει να περάσουμε μία φορά από κάθε μάθημα. Προφανώς, όσο περισσότερες είναι οι επικαλύψεις τόσο λιγότερες φορές χρειάζεται να περάσουμε. Άρα έχουμε πολυπλοκότητα $O(n)$ σε αυτό το βήμα.
- Συνολική πολυπλοκότητα: $O(nlogn)+O(n)=O(nlogn)$.

Άσκηση 2

2.α.

- Για την επίλυση του προβλήματος επιλέγεται η χρήση στοίβας. Διατρέχουμε την είσοδο από αριστερά προς τα δεξιά, έχοντας αρχικοποιήσει ένα μετρητή ζευγών $p=0$.
- Κάθε φορά που συναντάμε πομπό, τον βάζουμε στη στοίβα.
- Κάθε φορά που συναντάμε δέκτη, αν η στοίβα είναι μη κενή, βγάζουμε έναν πομπό και αυξάνουμε τον μετρητή p κατά ένα. Διαφορετικά, προχωράμε στο επόμενο στοιχείο.
- Μήκος μέγιστης ζητούμενης υπακολουθίας: $2p$.
- Πολυπλοκότητα: $O(n)$.

2.β.

Για την επίλυση του προβλήματος επιλέγεται δυναμικός προγραμματισμός.
Αναδρομική σχέση $C(i,j)=min[C(i-1,j+1)+R_i, C(i-1,j-1)+T_i]$.όπου $C(i,j)$ το κόστος της βέλτιστης λύσης μέχρι και την i-οστή κεραία, έχοντας j ≤ i πομπούς που δεν έχουν αντιστοιχηθεί σε δέκτες μέχρι στιγμής.

Αν έχουμε τη βέλτιστη λύση για i-1 κεραίες, τότε:
δεν έχουν αντιστοιχηθεί σε δέκτες προς το παρόν:

- Είτε υπάρχουν j+1 πομποί που δεν έχουν αντιστοιχηθεί σε δέκτες και η i-οστή κεραία γίνεται δέκτης οπότε μένουν j πομποί που δεν έχουν αντιστοιχηθεί σε δέκτες και προστίθεται R i στη συνολική κατανάλωση ισχύος.
- Είτε υπάρχουν j − 1 πομποί που δεν έχουν αντιστοιχηθεί σε δέκτες και η i-οστή κεραία γίνεται πομπός οπότε μένουν j πομποί που δεν έχουν αντιστοιχηθεί σε δέκτες και προστίθεται T i στη συνολική κατανάλωση ισχύος.
- Η ζητούμενη λύση είναι η C (n, 0).
- Πολυπλοκότητα: Υπολογισμός ενός εκ των $\Theta(n^2/2)$ στοιχείων του C σε $O(1)\rightarrow O(n^2)$.

Άσκηση 3

3.α.

- Δημιουργούμε ένα πίνακα n γραμμών με 2 στήλες έκαστη. Το πρώτο στοιχείο κάθε γραμμής είναι ίσο με το $rest(i)$ του αντίστοιχου κομματιού. Το δεύτερο είναι ίσο με το $score(i)$ μείον το άθροισμα των score των $rest(i)$ επόμενων κομματιών της λίστας.
- Ταξινομούμε τον πίνακα σε αύξουσα σειρά βάσει της δεύτερης γραμμής .
- Ουσιαστικά το πρόβλημα που έχουμε τώρα να λύσουμε είναι ίδιο με το πρόβλημα που επιλύσαμε στο υποερώτημα 1.α.2. Φτιάχνουμε μία λίστα με αντίστοιχο s_i το i , και αντίστοιχο f_i το $rest(i)+i$.
- Αντί του w_i , έχουμε τώρα σαν επιμέρους βάρους την τιμή που υπάρχει στη δεύτερη γραμμή του πίνακα που δημιουργήσαμε αρχικά.
- Με πανομοιότυπο τρόπο υπολογίζουμε την αντίστοιχη τιμή $C(n)$ σε χρόνο $O(nlogn)$ με την παρατήρηση ότι τα w_i δεν είναι απαραίτητο να έχουν θετικές τιμές. Η αναδρομική σχέση παραμένει ως έχει, καθώς μας ενδιαφέρουν οι μεγαλύτερες (όχι κατ' απόλυτη τιμή) τιμές.
- Ο υπολογισμός των αθροισμάτων για να συμπληρωθεί η δεύτερη γραμμή του αρχικού πίνακα έχει πολυπλοκότητα $O(n*i)$.
- Συνεπώς έχουμε συνολική πολυπλοκότητα $O(n*i)+O(n*logn)=(O(n*(i+logn)))$
  = $O(n*(max(i,logn)))$

3.β.

Εφαρμόζουμε τον αλγόριθμο που περιγράφηκε στο 3.α. με δύο τροποποιήσεις:

- Στο πρώτο στάδιο, η τιμή της δεύτερης γραμμής για κάθε στοιχείο i προκύπτει από την τιμή $score(i)$ μείον το άθροισμα των score των $rest(i)$ επόμενων κομματιών, μείον το άθροισμα των score των prep(i) προηγούμενων κομματιών της λίστας.τροποποιείται έτσι αντίστοιχα το w_i κάθε γραμμής.

- Σε κάθε στάδιο της αναδρομικής λύσης, ελέγχουμε πιθανή επικάλυψη μεταξύ των προηγούμενων κομματιών για ένα δεδομένο i και των επόμενων του i-1. Συγκεκριμένα, αντί της τιμής , υπολογίζουμε την τιμή

$$if\,[i+rest(i-1)<i-prep(i)]$$
$$then\,C(i)=max[w_i+C(previous(i)),C(i-1)]$$
$$else\,C(i)=max[w_i+C(previous(i-prep(i)+rest(i-1))),C(i-1)]$$

<u>Άσκηση 4</u>

Για τη λύση του προβλήματος επιλέγεται δυναμικός προγραμματισμός:
Αναδρομική σχέση $c(i,j)=max[min[c(i-1,j),c(i-1,j-1),c(i,j-1)],d(p_i,p_j)]$,
όπου $c(i,j)$ το ελάχιστο μήκος λουριού για είσοδο $(p_1,p_2,...,p_i)$ και $(q_1,q_2,...,q_j)$.

- Σίγουρα κάποια στιγμή θα βρεθείτε και οι δύο στις τελικές θέσεις p_i,q_j , άρα το λουρί πρέπει να έχει μήκος τουλάχιστον $d(p_i,q_j)$.
- Για να φτάσετε εκεί, είτε κινηθήκατε και οι δύο ταυτόχρονα, είτε μόνο ο σκύλος, είτε μόνο εσείς.
- Η ζητούμενη λύση είναι η $c(n,m)$ και αρχικοποιούμε $c(i,0)=c(0,j)=0,\forall i\in[n],j\in[m]$.
- Πολυπλοκότητα: $O(mn)$.