Αλγόριθμοι και Πολυπλοκότητα
2η Σειρά Γραπτών και Προγραμματιστικών Ασκήσεων

CoReLab

ΣΗΜΜΥ - Ε.Μ.Π.

Δεκέμβριος 2016

Outline

Άσκηση 1 (α.1): Δρομολόγηση Μαθημάτων

Το Πρόβλημα Δρομολόγησης Μαθημάτων

Είσοδος: Χρονικά διαστήματα διδασκαλίας μαθημάτων $[s_i, f_i)$, $i = 1, \ldots, n$.

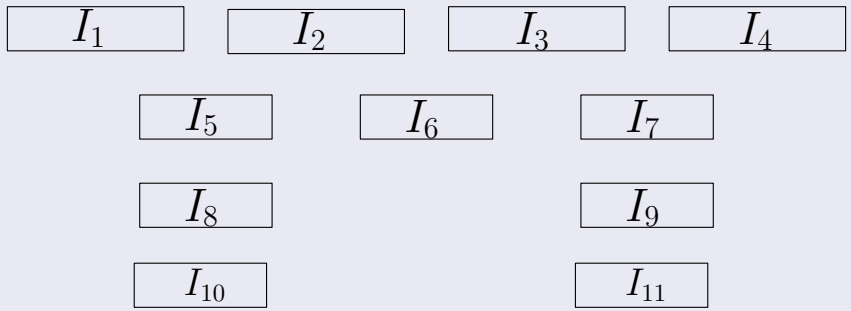
Ζητείται: Μεγίστου πλήθους σύνολο μαθημάτων χωρίς επικαλύψεις.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

1. Λιγότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



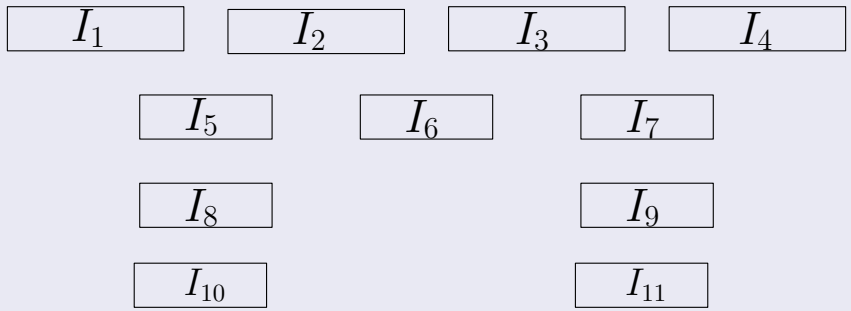
- Λύση Κριτηρίου, S_1: Επιλέγει τα I_6, I_1, I_4. $\Longrightarrow |S_1| = 3$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_2, I_3, I_4\} \Longrightarrow |OPT| = 4$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

1. Λιγότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



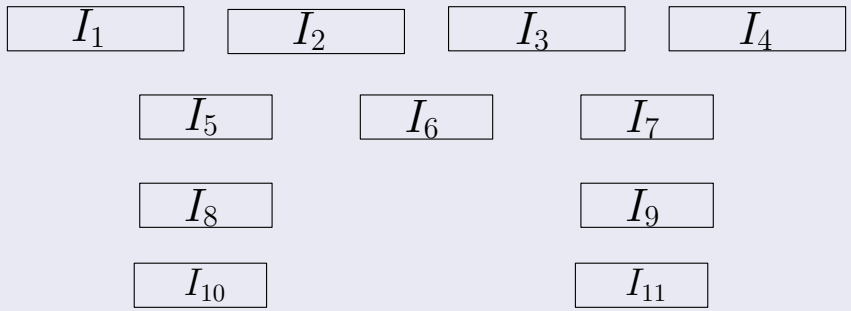
- Λύση Κριτηρίου, S_1: Επιλέγει τα I_6, I_1, I_4. $\Longrightarrow |S_1| = 3$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_2, I_3, I_4\} \Longrightarrow |OPT| = 4$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

1. Λιγότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



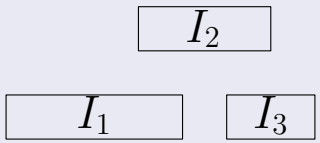
- Λύση Κριτηρίου, S_1: Επιλέγει τα I_6, I_1, I_4. $\Longrightarrow |S_1| = 3$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_2, I_3, I_4\} \Longrightarrow |OPT| = 4$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

2. Μεγαλύτερη Διάρκεια

Λάθος!

Αντιπαράδειγμα:



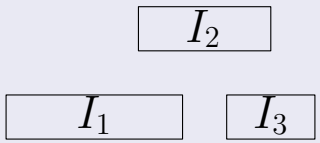
- Λύση Κριτηρίου, S_2: Διαγράφει πρώτα το I_1 και μετά το I_2 $\Longrightarrow |S_2| = |\{I_3\}| = 1$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_3\} \Longrightarrow |OPT| = 2$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

2. Μεγαλύτερη Διάρκεια

Λάθος!

Αντιπαράδειγμα:



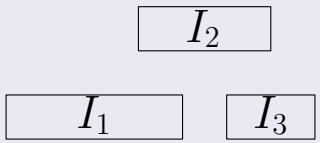
- Λύση Κριτηρίου, S_2: Διαγράφει πρώτα το I_1 και μετά το I_2 $\Longrightarrow |S_2| = |\{I_3\}| = 1$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_3\} \Longrightarrow |OPT| = 2$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

2. Μεγαλύτερη Διάρκεια

Λάθος!

Αντιπαράδειγμα:



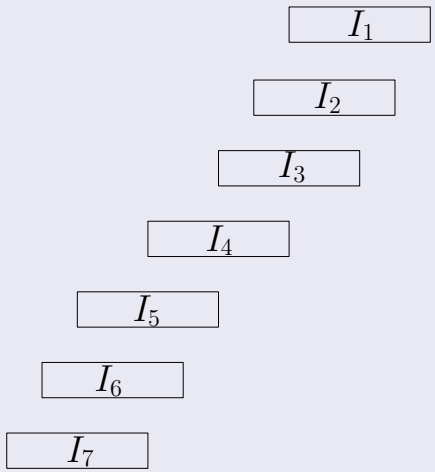
- Λύση Κριτηρίου, S_2: Διαγράφει πρώτα το I_1 και μετά το I_2 $\Longrightarrow |S_2| = |\{I_3\}| = 1$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_3\} \Longrightarrow |OPT| = 2$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

3. Περισσότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



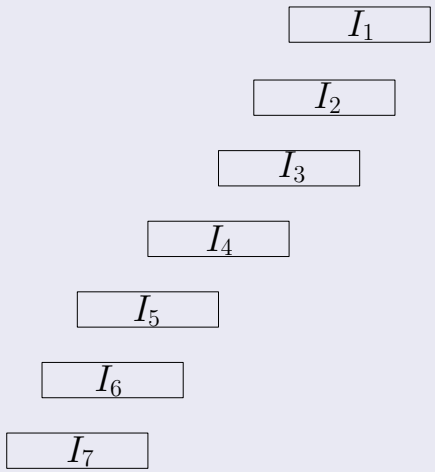
- Λύση Κριτηρίου, S_3: Διαγράφει πρώτα το I_4, έπειτα τα I_3, I_5, I_2, I_6. $\Longrightarrow |S_3| = |\{I_1, I_7\}| = 2$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_4, I_7\} \Longrightarrow |OPT| = 3$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

3. Περισσότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



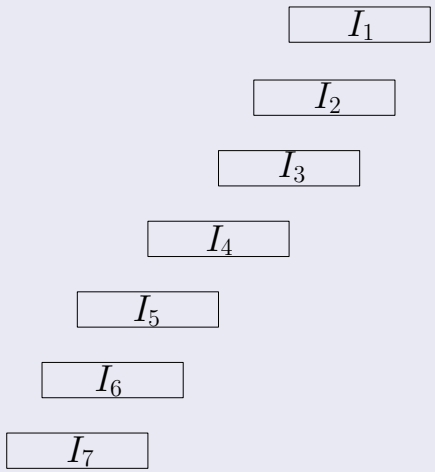
- Λύση Κριτηρίου, S_3: Διαγράφει πρώτα το I_4, έπειτα τα I_3, I_5, I_2, I_6. $\Longrightarrow |S_3| = |\{I_1, I_7\}| = 2$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_4, I_7\} \Longrightarrow |OPT| = 3$.

Άσκηση 1 (α.1): Άπληστα Κριτήρια Δρομολόγησης Μαθημάτων

3. Περισσότερες Επικαλύψεις

Λάθος!

Αντιπαράδειγμα:



- Λύση Κριτηρίου, S_3: Διαγράφει πρώτα το I_4, έπειτα τα I_3, I_5, I_2, I_6. $\Longrightarrow |S_3| = |\{I_1, I_7\}| = 2$.
- Βέλτιστη Λύση, OPT: $\{I_1, I_4, I_7\} \Longrightarrow |OPT| = 3$.

Άσκηση 1 (α.2): Δρομολόγηση Μαθημάτων με Βάρη

Είσοδος: Χρονικά διαστήματα μαθημάτων $[s_i, f_i)$, $i = 1, \ldots, n$ με αντίστοιχες διδακτικές μονάδες w_i.

Ζητείται: Σύνολο μαθημάτων χωρίς επικαλύψεις με μέγιστο συνολικό άθροισμα διδακτικών μονάδων.

Άσκηση 1 (α.2): Δρομολόγηση Μαθημάτων με Βάρη

Λύση:

- Ταξινομούμε τα χρονικά διαστήματα σε αύξουσα σειρά με βάση το χρόνο ολοκλήρωσής τους, έστω για ευκολία $f_1, \ldots, f_n$ αυτή η σειρά.
- Συμβολίζουμε με $C[i]$ το μέγιστο σύνολο διδακτικών μονάδων μίας βέλτιστης λύσης για τα μαθήματα $\{1, \ldots, i\}$.

Προφανώς, η ζητούμενη λύση είναι η $C[n]$ και η σχέση αρχικοποιείται στο $C[1] = w_1$.

Άσκηση 1 (α.2): Δρομολόγηση Μαθημάτων με Βάρη

Λύση:

Για την κατασκευή της αναδρομικής σχέσης, χρειαζόμαστε έναν πίνακα μήκους n, κάθε στοιχείο $\text{prev}[i]$ του οποίου θα περιέχει το δείκτη του μαθήματος που τελειώνει αργότερα και δεν έχει επικάλυψη με το i.
Η αναδρομική σχέση είναι:

$$C[i] = \max\{w_i + C[\text{prev}[i]], C[i-1]\}$$

Άσκηση 1 (α.2): Δρομολόγηση Μαθημάτων με Βάρη

- Ταξινόμηση διαστημάτων και κατασκευή πίνακα prev: $O(n \log n)$.
- Εύρεση ενός εκ των n στοιχείων του πίνακα της C : $O(1)$.

Άρα συνολικά:

Πολυπλοκότητα

$$O(n \log n)$$

Σημείωση: Η ανακατασκευή της λύσης γίνεται από τον πίνακα, σε $O(n)$.

Άσκηση 1 (β): Ανακοινώσεις

Είσοδος: Χρονικά διαστήματα μαθημάτων $[s_i, f_i)$, $i = 1, \ldots, n$.

Ζητείται: Σύνολο στιγμών που αγγίζουν όλα τα μαθήματα και έχουν ελάχιστο πλήθος.

Άσκηση 1 (β): Ανακοινώσεις

Λύση: Θεωρούμε ότι μία ανακοίνωση τη χρονική στιγμή f_i ενημερώνει και τους φοιτητές του μαθήματος i.

- Ταξινομούμε τα χρονικά διαστήματα σε αύξουσα σειρά με βάση το χρόνο ολοκλήρωσής τους, έστω για ευκολία $f_1, \ldots, f_n$ αυτή η σειρά.
- Βάζουμε στη λύση μας το σημείο f_i του πρώτου ακάλυπτου διαστήματος, σημειώνουμε ποιά διαστήματα έχουμε καλύψει και συνεχίζουμε με το επόμενο ακάλυπτο διάστημα. (βλ. "Άσκήσεις σε άπληστους αλγόριθμους και δυναμικό προγραμματισμό")

Πολυπλοκότητα

$$O(n \log n)$$

Άσκηση 1 (β): Ανακοινώσεις

Απόδειξη ορθότητας:

- Έστω P το σύνολο στιγμών που επέλεξε ο αλγόριθμός μας, O το σύνολο της βέλτιστης λύσης και έστω ότι στο πρώτο σημείο στο οποίο διαφέρουν ο αλγόριθμός μας επιλέγει $p = f_k - 1$ (για να καλύψει το διάστημα $[s_k, f_k)$) ενώ η βέλτιστη λύση έχει στην ίδια θέση τη στιγμή o.
- Αν $o > p$, τότε η βέλτιστη λύση δεν αγγίζει το διάστημα k (αφού μέχρι τώρα καλύπτουν τα ίδια διαστήματα) και οι φοιτητές του μαθήματος k μένουν ανενημέρωτοι στη βέλτιστη λύση! Άτοπο.
- Αν $o < p$: Μέχρι τώρα και οι δύο λύσεις έχουν καλύψει όλα τα διαστήματα που τελειώνουν πριν τη στιγμή f_k. Άρα αντικαθιστώντας το o με $p = f_k - 1$ η βέλτιστη λύση συνεχίζει να καλύπτει το $[s_k, f_k)$ και αποκλείεται να χάνει κάποιο άλλο διάστημα γιατί κάθε άλλο ακάλυπτο διάστημα τελειώνει μετά το f_k άρα αν το άγγιζε με το o θα το αγγίζει και με το p.

Outline

Άσκηση 2 (α): Πομποί και Δέκτες

Είσοδος: Ακολουθία $a = a_1, \ldots, a_n$ όπου $\forall i \ a_i = r$ ή $a_i = t$.

Ζητείται: Μέγιστη υπακολουθία όπου κάθε πομπός αντιστοιχεί σε ένα δέκτη προς τα δεξιά του.

Άσκηση 2 (α): Πομποί και Δέκτες

Λύση: Με 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 το μετρητή $p++$, διαφορετικά προχωράμε στο επόμενο στοιχείο.
- Μήκος μέγιστης ζητούμενης υπακολουθίας: $2p$.

Πολυπλοκότητα

$O(n)$

Άσκηση 2 (α): Πομποί και Δέκτες

Λύση: Με 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 το μετρητή $p++$, διαφορετικά προχωράμε στο επόμενο στοιχείο.
- Μήκος μέγιστης ζητούμενης υπακολουθίας: $2p$.

Πολυπλοκότητα

$O(n)$

Άσκηση 2 (α): Πομποί και Δέκτες

Λύση: Με 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 το μετρητή $p++$, διαφορετικά προχωράμε στο επόμενο στοιχείο.
- Μήκος μέγιστης ζητούμενης υπακολουθίας: $2p$.

Πολυπλοκότητα

$O(n)$

Άσκηση 2 (α): Πομποί και Δέκτες

Λύση: Με 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 το μετρητή $p++$, διαφορετικά προχωράμε στο επόμενο στοιχείο.
- Μήκος μέγιστης ζητούμενης υπακολουθίας: $2p$.

Πολυπλοκότητα

$$O(n)$$

Άσκηση 2 (β): Πομποί και Δέκτες

Είσοδος: Ακολουθία n (άρτιος) ζευγών (T_i, R_i), όπου T_i η κατανάλωση ισχύος της κεραίας i όταν λειτουργεί ως πομπός και $R_i \leq T_i$ όταν λειτουργεί ως δέκτης.

Ζητείται: Ελάχιστη συνολική κατανάλωση ισχύος, όπου για κάθε i που λειτουργεί ως πομπός υπάρχει ένα και μοναδικό $j > i$ που λειτουργεί ως δέκτης.

Άσκηση 2 (β): Πομποί και Δέκτες

Λύση: 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 $j \leq i$ πομπούς που δεν έχουν αντιστοιχηθεί σε δέκτες προς το παρόν.

Άσκηση 2 (β): Πομποί και Δέκτες

Λύση:

$$C[i,j] = \min\{C[i-1,j+1] + R_i, C[i-1,j-1] + T_i\}$$

Αν έχουμε τη βέλτιστη λύση για $i-1$ κεραίες τότε:

- 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.

Άσκηση 2 (β): Πομποί και Δέκτες

Λύση:
Η ζητούμενη λύση είναι η $C[n, 0]$.

Πολυπλοκότητα

Υπολογισμός ενός εκ των $\Theta(n^2/2)$ στοιχείων του C σε $O(1) \Longrightarrow$

$$O(n^2)$$

Υπάρχει πιο αποδοτικός αλγόριθμος;

Άσκηση 2 (β): Πομποί και Δέκτες

Λύση:
Η ζητούμενη λύση είναι η $C[n, 0]$.

Πολυπλοκότητα

Υπολογισμός ενός εκ των $\Theta(n^2/2)$ στοιχείων του C σε $O(1) \Longrightarrow$

$$O(n^2)$$

Υπάρχει πιο αποδοτικός αλγόριθμος;

Άσκηση 2 (β): Πομποί και Δέκτες

Καλύτερη Λύση: Για κάποια ανάθεση πομπών-δεκτών έστω $S[i]$ το πλήθος πομπών μείον το πλήθος δεκτών στο διάστημα $[1, i]$. Μία ανάθεση είναι έγκυρη αν και μόνο αν για κάθε $i \in \{1, ..., n\}$ ισχύει $S[i] \geq 0$ και $S[n] = 0$.

Θεωρούμε ότι έχουμε μια βέλτιστη έγκυρη ανάθεση στο διάστημα $[1, i]$. Αν ο i είναι περιττός, για να την επεκτείνουμε βάζουμε ένα δέκτη στο $i + 1$ ($S[i + 1] = 0$). Αν είναι άρτιος, βάζουμε ένα δέκτη στο $i + 1$, αλλά έχουμε $S[i + 1] = -1$, άρα κάποιος δέκτης στο $[1, i + 1]$ πρέπει να μετατραπεί σε πομπό: Μετατρέπουμε αυτόν που ελαχιστοποιεί την κατανάλωση, δηλαδή αυτόν με το ελάχιστο $T_j - R_j$. Τώρα έχουμε $S[i + 1] = 0$.

Για να βρίσκουμε το δέκτη με το ελάχιστο $T_j - R_j$ θα χρειαστούμε ένα min-heap στο οποίο οι λειτουργίες εισαγωγής και εξαγωγής ελαχίστου στοιχείου γίνονται το πολύ σε $O(\log n)$.

Άσκηση 2 (β): Πομποί και Δέκτες

Αλγόριθμος:

- Αρχικοποίησε ένα κενό min-heap Q
- Για $i = 1...n$
  - Θεώρησε τον i ως δέκτη, πρόσθεσε R_i στο συνολικό κόστος και βάλε το ζεύγος $(T_i - R_i, i)$ στο Q
  - Αν $S[i] < 0$, αφαίρεσε από το Q το δέκτη με το ελάχιστο $T_j - R_j$, μετάτρεψέ τον σε πομπό και αύξησε το συνολικό κόστος κατά $T_j - R_j$.

Άσκηση 2 (β): Πομποί και Δέκτες

Χρονική Πολυπλοκότητα

$$O(n \log n)$$

Μένει να δείξουμε το επαγωγικό βήμα, δηλαδή ότι η λύση στο διάστημα $[1, i+1]$ είναι βέλτιστη, δεδομένου ότι η λύση στο $[1, i]$ είναι βέλτιστη.

- Αν i περιττός ($S[i] = 1$), τότε προφανώς η θέση $i+1$ πρέπει να έχει δέκτη για να είναι έγκυρη η λύση και γιατί είναι το φθηνότερο. Άρα η λύση μας είναι βέλτιστη.
- Αν i άρτιος ($S[i] = 0$) τότε αν προσθέσουμε δέκτη στη θέση $i+1$ για να είναι έγκυρη η λύση πρέπει να μετατρέψουμε ένα δέκτη σε πομπό. Προφανώς αφού η λύση είναι μέχρι στιγμής ελαχίστου κόστους, προσθέτοντας το μικρότερο $T_j - R_j$ λόγω της αλλαγής (και R_{i+1} αντί T_{i+1}) έχουμε λύση μικρότερου δυνατού κόστους άρα ακόμα βέλτιστη.

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$								
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 0

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	-1							
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $0 + 6 = 6$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1							
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $6 + (8\text{-}6) = 8$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0						
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $8 + 3 = 11$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	-1					
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $11 + 1 = 12$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1					
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 12 + (5-1)=16

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1	0				
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $16 + 4 = 20$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1	0	-1			
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: $20 + 3 = 23$

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1	2	1			
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 23 + (6-4)=25

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1	2	1	0		
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 25+7=32

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	0	1	2	1	0	-1	
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 32+5=37

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	2	3	4	3	2	1	
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 37+(9-3)=43

Άσκηση 2 (β): Πομποί και Δέκτες

Παράδειγμα:

i	1	2	3	4	5	6	7	8
$S(i)$	1	2	3	4	3	2	1	0
T_i	8	9	5	6	10	30	12	2
R_i	6	3	1	4	3	7	5	1

Κατανάλωση: 43+1=44

Outline

Άσκηση 3 (α): Διαγωνισμός Χορού

Είσοδος: Βαθμολογία $score[i]$ και χρόνος ξεκούρασης $rest[i]$ για κάθε τραγούδι $i \in \{1, \ldots, n\}$.

Ζητείται: Επιτρεπόμενο σύνολο τραγουδιών που μεγιστοποιεί τη συνολική βαθμολογία.

Λύση: Δυναμικός Προγραμματισμός

Άσκηση 3 (α): Διαγωνισμός Χορού

Λύση:

Έστω $S[i]$ η μέγιστη συνολική βαθμολογία από το i μέχρι και το n.
Έχουμε δύο επιλογές:

- Επιλέγουμε το i και μένει η μέγιστη συνολική βαθμολογία από το $i + rest[i] + 1$ και μετά.
- Δεν επιλέγουμε το i και κρατάμε τη μέγιστη συνολική βαθμολογία από το επόμενο κιόλας τραγούδι και μετά.

Άσκηση 3 (α): Διαγωνισμός Χορού

Λύση:

Επομένως, η αναδρομική σχέση είναι:

$$S[i] = max\{score[i] + S[i + rest[i] + 1], S[i + 1]\}$$

Κατασκευάζουμε τον πίνακα από το n προς το 1 και το ζητούμενο είναι το $S[1]$.

Άσκηση 3 (α): Διαγωνισμός Χορού

Χρονική Πολυπλοκότητα

- Εύρεση ενός εκ των n στοιχείων του πίνακα: $O(1)$.

Άρα συνολικά:

$$O(n)$$

Άσκηση 3 (β): Διαγωνισμός Χορού

Είσοδος: Βαθμολογία $score[i]$, χρόνος ξεκούρασης $rest[i]$ και χρόνος προετοιμασίας $prep[i]$ για κάθε τραγούδι $i \in \{1, \ldots, n\}$.

Ζητείται: Επιτρεπόμενο σύνολο τραγουδιών που μεγιστοποιεί τη συνολική βαθμολογία.

Λύση: Δυναμικός Προγραμματισμός

Άσκηση 3 (β): Διαγωνισμός Χορού

Λύση:

Έστω $S[i]$ η μέγιστη συνολική βαθμολογία από το i μέχρι και το n.
Έχουμε δύο επιλογές:

- Επιλέγουμε το i και μένει η μέγιστη συνολική βαθμολογία από το επόμενο επιτρεπόμενο τραγούδι (λόγω προετοιμασίας του και ξεκούρασης του i) και μετά.
- Δεν επιλέγουμε το i και κρατάμε τη μέγιστη συνολική βαθμολογία από το επόμενο επιτρεπόμενο (λόγω προετοιμασίας του) τραγούδι και μετά.

Άσκηση 3 (β): Διαγωνισμός Χορού

Λύση:

Επομένως, η αναδρομική σχέση είναι:

$$S[i] = max\{score[i] + S[next[1, i]], S[next[0, i]]\}$$

όπου $next[x, i] = min\ j \in \{i + rest[i] * x + 1, \ldots, i + M + 1\}$ τέτοιο ώστε $j - prep[j] - 1 \geq i$.

Άσκηση 3 (β): Διαγωνισμός Χορού

Χρονική Πολυπλοκότητα

- Εύρεση ενός εκ των n στοιχείων του πίνακα: $O(M)$.

Άρα συνολικά:

$$O(nM)$$

Outline

Άσκηση 4: Βγάζοντας Βόλτα το Σκύλο

Είσοδος: Ακολουθίες σημείων $(p_1, \ldots, p_n)$ και $(q_1, \ldots, q_m)$.

Ζητείται: Αν $p(t) \in \{p_1, \ldots, p_n\}$ και $q(t) \in \{q_1, \ldots, q_m\}$ τα σημεία στα οποία βρίσκεστε εσείς και ο σκύλος σας αντίστοιχα τη χρονική στιγμή t, ζητείται το $\min_{p,q} \max_t d(p(t), q(t))$.
(Δηλαδή για την 'καλύτερη' παράλληλη διάσχιση, η μέγιστη απόσταση σημείων στα οποία βρίσκεστε ταυτόχρονα.)

Άσκηση 4: Βγάζοντας Βόλτα το Σκύλο

Λύση: Δυναμικός Προγραμματισμός

Αναδρομική Σχέση

$$c[i,j] = \max\{\min\{c[i-1,j], c[i-1,j-]), c[i,j-1]\}, d(p_i, p_j)\}$$

όπου $c[i,j]$ το ελάχιστο μήκος λουριού για είσοδο $(p_1, \ldots, p_i)$ και $(q_1, \ldots, q_j)$

- 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.

Άσκηση 4: Βγάζοντας Βόλτα το Σκύλο

Λύση:

$$c[i,j] = \max\{\min\{c[i-1,j], c[i-1,j-1], c[i,j-1]\}, d(p_i, p_j)\}$$

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

$$O(nm)$$

Outline

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Είσοδος: Ακολουθία τιμών $p(1), \ldots, p(N)$ και επιτρεπόμενο πλήθος αγοραπωλησιών K.

Έξοδος: Μέγιστο κέρδος, αν οι ενέργειες αγοράς και πώλησης πρέπει να γίνονται εναλλάξ.

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Ιδέα Δυναμικού Προγραμματισμού

Αν j είναι το πλήθος των αγοραπωλησιών, μπορούμε να χρησιμοποιούμε το $j\%2$ για να ξεχωρίζουμε αν βρισκόμαστε σε σημείο πώλησης ή αγοράς. Συμβολίζουμε με $C[i, j\%2]$ το μέγιστο κέρδος που μπορούμε να πετύχουμε μέχρι και την i-οστή μέρα με j αγοραπωλησίες. Η αναδρομική σχέση είναι:

$C[i, j\%2] = max\{C[i-1, j\%2], max_{1 \leq z \leq i-1}\{C[z, (j-1)\%2] + p(i) - p(z)\}\}$

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Λύση

$$C[i, j\%2] = max\{C[i-1, j\%2], max_{1 \leq z \leq i-1}\{C[z, (j-1)\%2] + p(i) - p(z)\}\}$$

- Είτε δεν πραγματοποιούμε αγοραπωλησία τη μέρα i
- Είτε επιλέγουμε να πουλήσουμε, αγοράζοντας κάποια από τις προηγούμενες μέρες (τη φθηνότερη)

Συμπληρώνουμε τον πίνακα στήλη-στήλη και η λύση βρίσκεται στο $C[N, (K-1)\%2]$.

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Χρονική Πολυπλοκότητα

Για κάθε στοιχείο από τα $O(KN)$ στοιχεία που πρέπει να υπολογίσουμε κάνουμε χρόνο $O(i)$ άρα συνολικά η λύση μας έχει πολυπλοκότητα $O(KN^2)$.

Μπορούμε καλύτερα;

Παρατήρηση

Αν έχουμε υπολογίσει τη φθηνότερη τιμή μέχρι ένα σημείο δε χρειάζεται να ξανακοιτάξουμε αυτές τις τιμές. Αντί αυτού την κρατάμε και την ανανεώνουμε όποτε χρειαστεί συγκρίνοντάς την με νέες τιμές.

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Ιδέα για λύση $O(KN)$

$$C[i, j\%2] = max\{C[i-1, j\%2], cheaper + p(i)\}$$

όπου $cheaper = \max\{cheaper, C[i-1, (j-1)\%2] - p(i-1)\}$

1η Προγραμματιστική: Εμπόριο Σοκολάτας

Χρονική Πολυπλοκότητα

Για κάθε ένα εκ των $O(KN)$ στοιχείων που πρέπει να υπολογίσουμε χρειάζεται να κάνουμε δύο συγκρίσεις άρα $O(1)$ χρόνο. Συνολικά επομένως η λύση έχει πολυπλοκότητα $O(KN)$.

Outline

2η Προγραμματιστική: Χημικά Απόβλητα

Είσοδος: Ένας πίνακας $A[i, j]$ με διαστάσεις $N \times N$ όπου η θέση (i, j) περιέχει το ποσό ενέργειας που εκλύεται αν αντιδράσει η ουσία i με την ουσία j. Επίσης δίνεται το πλήθος K των φιαλών.

Έξοδος: Το ελάχιστο ποσό ενέργειας που μπορεί να εκλυθεί αν τοποθετήσουμε της χημικές ουσίες σε φιάλες με κατάλληλο τρόπο (οι ουσίες πρέπει να τοποθετηθούν στις φιάλες με την ίδια σειρά με την οποία είναι αριθμημένες).

Ιδέα Δυναμικού Προγραμματισμού

- Συμβολίζουμε με $dp[i, j]$ την ελάχιστη ενέργεια που μπορεί εκλυθεί αν τοποθετήσουμε τις i πρώτες χημικές ουσίες σε j φιάλες.
- Η λύση που αναζητούμε είναι η $dp[N, K]$.
- Μπορούμε να υπολογίσουμε το $dp[i, j]$ με την εξής αναδρομική σχέση:

$$dp[i, j] = \min_{1 \leq t \leq i-1} \{dp[t, j-1] + \text{energy}(t+1, i)\}$$

όπου $\text{energy}(x, y)$ είναι η ενέργεια που εκλύεται αν τοποθετηθούν στην ίδια φιάλη όλες οι ουσίες l για τις οποίες $x \leq l \leq y$.

- Τις τιμές $\text{energy}(x, y)$ για κάθε x, y μπορούμε να υπολογίσουμε από πριν σε χρόνο $O(NK)$ (πώς;)

2η Προγραμματιστική: Χημικά Απόβλητα

Χρονική Πολυπλοκότητα

Για κάθε θέση του πίνακα dp χρειάζεται να το ελέγξουμε $O(N)$ τιμές για το t και έτσι για να γεμίσει όλος ο πίνακας χρειάζεται χρόνος:

$$O(N^2K)$$

Μπορούμε καλύτερα;

Ιδέα (για bonus)

Για σταθερό j (πλήθος φιαλών) θα προσπαθήσουμε να υπολογίσουμε τα $dp[i,j]$ για κάθε $i > j$ σε χρόνο $O(N \log N)$ (αντί για $O(N^2)$ που ήταν πρίν) με χρήση Διαίρει και Βασίλευε.

2η Προγραμματιστική: Χημικά Απόβλητα

Ιδέα

Μαζί με τον πίνακα dp θα χρειαστούμε κι έναν πίνακα opt ο οποίος στη θέση (i, j) κρατάει την τιμή του t για την οποία επιτυγχάνεται η βέλτιση λύση κατά τον υπολογισμό του $\text{dp}[i, j]$, δηλαδή:

$$\text{opt}[i, j] = \text{argmin}_{1 \leq t \leq n-1}\{\text{dp}[t, j-1] + \text{energy}(t+1, i)\}$$

Αν υπάρχουν πολλά τέτοια t κρατάμε το μικρότερο.

Παρατήρηση

Αν σταθεροποιήσουμε το j τότε ο opt είναι αύξουσα συνάρτηση του i, δηλαδή:

$$\text{opt}[1, j] \leq \text{opt}[2, j] \leq \ldots \leq \text{opt}[j, j]$$

Απόδειξη;

2η Προγραμματιστική: Χημικά Απόβλητα

Βελτιστοποίηση Διαίρει και Βασίλευε (παράδειγμα)

- Έστω ότι $n = 100$ και για σταθερό j θέλουμε να υπολογίσουμε όλα τα $dp[i, j]$ έχοντας ήδη υπολογίσει όλο τον πίνακα dp για τα μικρότερα j.
- Μπορούμε να υπολογίσουμε πρώτα το $dp[50, j]$ (ελέγχοντας όλα τα t) και μαζί το $\text{opt}[50, j] = x$.
- Από την προηγούμενη Παρατήρηση προκύπτει ότι το x είναι ένα άνω φράγμα για τα t που πρέπει να ελέγξουμε όταν θα υπολογίζουμε τα $dp[i, j]$ για $i < 50$ και αντίστοιχα είναι κάτω φράγμα για τα t όταν είμαστε σε $i > 50$.
- Μπορούμε να συνεχίσουμε αυτή τη διαδικασία αναδρομικά, δηλαδή για να υπολογίσουμε τα $dp[i, j]$ για $i = 1, \ldots, 49$ βρίσκουμε πρώτα το $dp[25, j]$ (ελέγχοντας πλέον μόνο $t \leq x$). Αντίστοιχα για $i > 50$ και κάθε φορά ανανεώνουμε τα φράγματα που έχουμε βρεί.

2η Προγραμματιστική: Χημικά Απόβλητα

```
1: procedure SOLVE(j, L, R, optL, optR)
2:     if L == R then
3:         COMPUTE_DP_OPT(L, j, optL, optR)
4:     else
5:         M ← (L + R)/2
6:         COMPUTE_DP_OPT(M, j, optL, optR)
7:         SOLVE(j, L, M − 1, optL, opt[M][j])
8:         SOLVE(j, M + 1, R, opt[M][j], optR)
```

όπου η διαδικασία Compute_dp_opt$(i, j, \text{optL}, \text{optR})$ υπολογίζει το $\text{dp}[i, j]$ και το $\text{opt}[i, j]$ με την αναδρομική σχέση του δυναμικού προγραμματισμού περιορίζοντας τη αναζήτηση για το κατάλληλο t στο διάστημα $[\text{optL}, \text{optR}]$.

2η Προγραμματιστική: Χημικά Απόβλητα

Χρονική Πολυπλοκότητα

- Κάθε κόμβος του δέντρου αναδρομής χαρακτηρίζεται από δύο διαστήματα: το $[L, R]$ και το $[\text{optL}, \text{optR}]$.
- Χρεώνουμε σε κάθε κόμβο το υπολογιστικό κόστος της γραμμής 6 (ή της 3 αν είναι φύλλο).
- Κάθε αναδρομική κλήση μειώνει το διάστημα $[L, R]$ στο μισό. Συνεπώς το δέντρο αναδρομής θα έχει ύψος $\Theta(\log N)$.
- Το κόστος ενός κόμβου με διάστημα $[\text{optL}, \text{optR}]$ είναι $optR - optL + 1$.
- Κάθε κόμβος του δέντρου αναδρομής διαμερίζει το διάστημα $[optL, optR]$ στα δύο παιδιά του (πιθανώς όχι ισόποσα).
- Σε κάθε περίπτωση το αθροιστικό κόστος κάθε επιπέδου του δέντρου είναι $O(N)$.

Ανάλυση Χρονικής Πολυπλοκότητας

- Συνεπώς μια κλήση στην Solve για συγκεκριμένο j κάνει χρόνο $O(N \log N)$.
- Για την τελική λύση του προβλήματος καλούμε την Solve για όλα τα $j = 1, \ldots, K$ και έτσι ο αλγόριθμος έχει πολυπλοκότητα:

$$O(NK \log N)$$