Αλγόριθμοι και Πολυπλοκότητα

2η σειρά γραπτών ασκήσεων

Σχέδιο Λύσεων

CoReLab

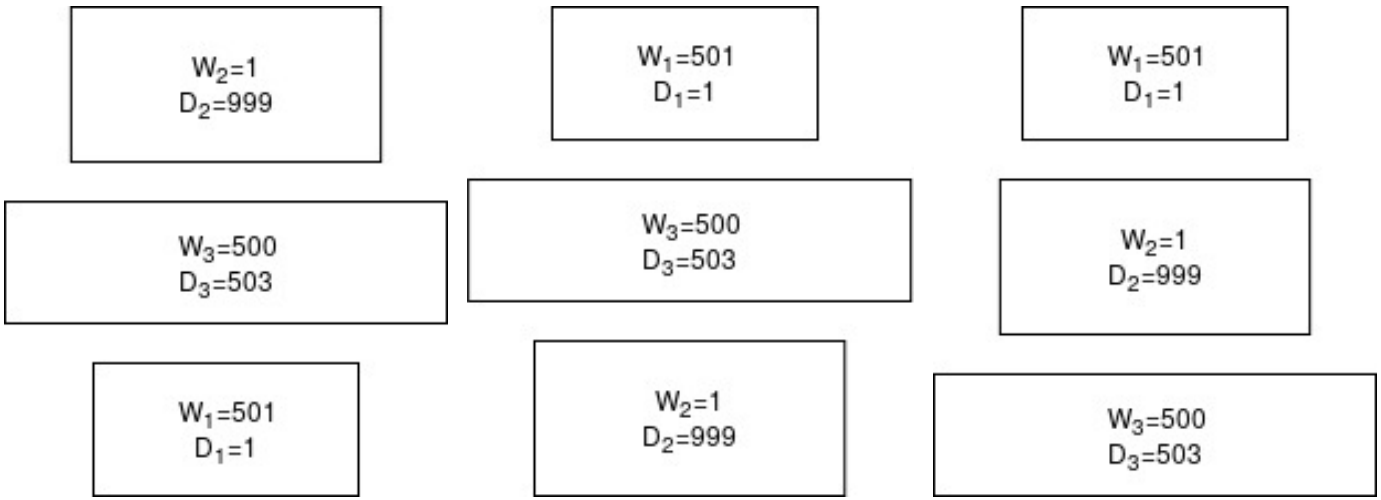
ΣΗΜΜΥ Ε.Μ.Π.

17 Νοεμβρίου 2019

Outline

Τα κριτήρια (ι),(ιι) δεν δίνουν απαραίτητα σωστή στοίβαξη.

Αντιπαράδειγμα: Για τρία δέματα με $w_1 = 501, d_1 = 1$, $w_2 = 1, d_2 = 999$ και $w = 500, d_3 = 503$.



Το κριτήριο (iii) δίνει σωστή στοίβαξη, εφόσον υπάρχει.

Απόδειξη: Έστω μια σωστή στοίβαξη, όπου το δέμα i είναι ακριβώς πάνω από το δέμα j και $w_i + d_i > w_j + d_j$. Τότε αν εναλλάξουμε τα i,j θα έχουμε και πάλι σωστή στοίβαξη. Πράγματι, αφού στην αρχή η στοίβαξη ήταν σωστή, τότε $W + w_i < d_j \implies W < d_j - w_i < d_i - w_j \implies W + w_j < d_i$.

Άρα απο κάθε σωστή στοίβαξη μπορούμε να φτιάξουμε μια σωστή στοίβαξη με τα $w_i + d_i$ διατεταγμένα με τη σωστή σειρά.

Άσκηση 1: Μεταφορά Δεμάτων

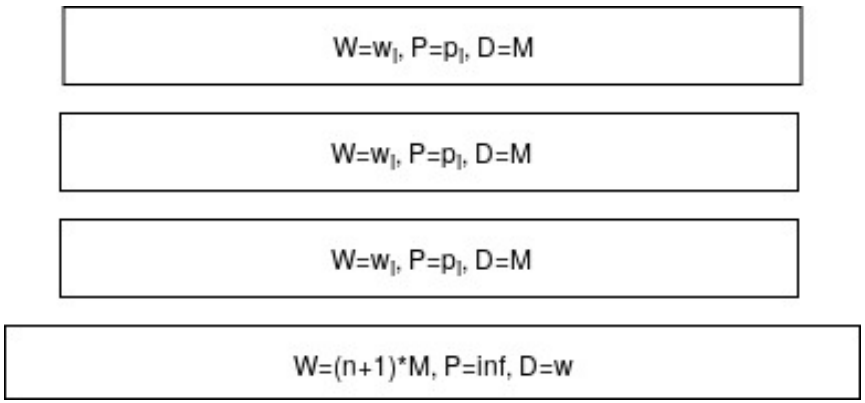
Αλγόριθμος: Παρόμοια με το 0-1 knapsack. Διατάσσουμε τα δέματα σε φθίνουσα σειρά $w_i + d_i$. Για κάθε (i, w), με $d_{i-1} \geq w$, έστω $K(i, w)$ το μέγιστο κέρδος από ασφαλή στοίβαξη δεμάτων από το n μέχρι και το i (από πάνω προς τα κάτω) και με συνολικό βάρος το πολύ w. Η αναδρομική είναι (συμπληρώστε τις αρχικές συνθήκες):
$K(i, w) = \max(p_i + K(i + 1, \min\{d_i, w - w_i\}), K(i + 1, w))$.

Όπως και στο knapsack η πολυπλοκότητα είναι $O(nlogn + nW)$ δηλαδή ψευδοπολυωνυμική.

Η ορθότητα του αλγορίθμου προκύπτει από το ότι όλες οι στοιβάξεις με μέγιστο κέρδος μπορούν να υλοποιηθούν με ασφαλή στοίβαξη.

Άσκηση 1: Μεταφορά Δεμάτων

Δε μπορούμε να κάνουμε κάτι καλύτερο από το παραπάνω (μάλλον), αφού μπορούμε να ανάξουμε το πρόβλημα των δεμάτων στο knapsack, όπου M αυθαίρετα μεγάλος αριθμός και p_i, w_i, w οι παραμέτροι του knapsack instance



Outline

Άσκηση 2: Αναμνηστικά

Είσοδος: n πόλεις, Προυπολογισμός C, Πίνακας p_{ij} συναισθηματικής αξίας, Πίνακας c_{ij} χρηματικής αξίας.

Έξοδος: Επιλογή αντικειμένων $(a_1, \ldots, a_n)$ ένα για κάθε πόλη, ώστε $\sum_{j=1}^{n} c_{ja_j} \leq C$ και να μεγιστοποιείται η συνολική συναισθηματική αξία.

Διαίσθηση

- Έστω $(a_1^*, \ldots, a_n^*)$ η βέλτιστη λύση του προβλήματος.
- Τότε, η $(a_1^*, \ldots, a_{n-1}^*)$ είναι βέλτιστη λύση του προβλήματος με $C' = C - c_{na_n}$ για τις $n-1$ πόλεις.
- Αρχή βελτιστότητας $\Longrightarrow$ δυναμικός προγραμματισμός!

Λύση

- Έστω A ένας $n \times C$ πίνακας με $A[i, j]$ η μέγιστη δυνατή συναισθηματική αξία για τις πόλεις 1 έως i με προυπολογισμό j.
- Ακραίες περιπτώσεις:
  - $A[i, j] = -1$ αν τα χρήματα j δεν επαρκούν για να αγοραστεί αντικείμενο από την πόλη i
  - $A[1, j]$ υπολογίζεται εύκολα.
- Αναδρομική σχέση :
  $A[i, j] = \max_{1 \leq u \leq k_i, c_{iu} \leq j}(A[i-1, j-c_{iu}] + p_{iu})$

Λύση

- Έστω A ένας $n \times C$ πίνακας με $A[i, j]$ η μέγιστη δυνατή συναισθηματική αξία για τις πόλεις 1 έως i με προυπολογισμό j.
- Ακραίες περιπτώσεις:
  - $A[i, j] = -1$ αν τα χρήματα j δεν επαρκούν για να αγοραστεί αντικείμενο από την πόλη i
  - $A[1, j]$ υπολογίζεται εύκολα.
- Αναδρομική σχέση :
  $A[i, j] = \max_{1 \leq u \leq k_i, c_{iu} \leq j}(A[i-1, j-c_{iu}] + p_{iu})$

Λύση

- Έστω A ένας $n \times C$ πίνακας με $A[i, j]$ η μέγιστη δυνατή συναισθηματική αξία για τις πόλεις 1 έως i με προυπολογισμό j.
- Ακραίες περιπτώσεις:
  - $A[i, j] = -1$ αν τα χρήματα j δεν επαρκούν για να αγοραστεί αντικείμενο από την πόλη i
  - $A[1, j]$ υπολογίζεται εύκολα.
- Αναδρομική σχέση :
  $A[i, j] = \max_{1 \leq u \leq k_i, c_{iu} \leq j}(A[i-1, j-c_{iu}] + p_{iu})$

Αλγόριθμος

- Γεμίζουμε τον πίνακα ξεκινώντας από $i = 1$ μέχρι n χρησιμοποιώντας την αναδρομή.
- Η βέλτιστη επιλογή αντικειμένων προκύπτει από τις επιλογές του u που δίνει το max.
- Χρονική πολυπλοκότητα $O(nCK)$, όπου $K = \max_i k_i$.
- Δεν υπάρχει πολυωνυμικός αλγόριθμος, εκτός αν $P = NP$ (αναγωγή από Discrete Knapsack).

Αλγόριθμος

- Γεμίζουμε τον πίνακα ξεκινώντας από $i = 1$ μέχρι n χρησιμοποιώντας την αναδρομή.
- Η βέλτιστη επιλογή αντικειμένων προκύπτει από τις επιλογές του u που δίνει το max.
- Χρονική πολυπλοκότητα $O(nCK)$, όπου $K = \max_i k_i$.
- Δεν υπάρχει πολυωνυμικός αλγόριθμος, εκτός αν $P = NP$ (αναγωγή από Discrete Knapsack).

Outline

Άσκηση 3: Σοκολατάκια

Είσοδος: Ακολουθία n ζευγών (q_i, t_i), όπου q_i το πλήθος από σοκολατάκια που περιέχει το κουτί i και t_i ο τύπος τους, αρχική θέση (κουτί) p και ελάχιστο πλήθος Q σοκολατακίων που πρέπει να καταναλωθούν. Τα κουτιά είναι διατεταγμένα σε γραμμή και το κόστος μετακίνησης από το i στο j είναι $|i - j|$.

Ζητείται: Σε κάθε βήμα μπορούμε να φάμε μόνο περισσότερα σοκολατάκια και διαφορετικού τύπου από το προηγούμενο και ξεκινάμε από το κουτί p. Ποιος είναι ο ελάχιστος χρόνος (και αλληλουχία κουτιών) για να φάμε τουλάχιστον Q σοκολατάκια.

Άσκηση 3: Σοκολατάκια (γνησίως αύξουσα κατανάλωση)

Λύση: Δυναμικός Προγραμματισμός.
Αφού τελειώνουμε με κατανάλωση (δεν συμφέρει να μετακινηθούμε μόλις φάμε Q σοκολατάκια), μπορούμε να ασχοληθούμε με τα υπο-προβλήματα $c[i,j]$ όπου όπου $c[i,j]$ ο ελάχιστος χρόνος που χρειαζόμαστε για να φάμε i σοκολατάκια, τρώγοντας στο τέλος από το κουτί j.

Αναδρομική Σχέση

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

Αρχή βελτιστότητας: Αν η βέλτιστη ακολουθία κουτιών που τρώμε, για να φάμε i σοκολατάκια καταλήγοντας στο κουτί j είναι k_a, k_b, k_o, k_i, η βέλτιστη ακολουθία για να φάμε $i - q_j$ σοκολατάκια καταλήγοντας στο κουτί k_o είναι k_a, k_b, k_o.

Άσκηση 3: Σοκολατάκια (γνησίως αύξουσα κατανάλωση)

Λύση: Δυναμικός Προγραμματισμός.
Αφού τελειώνουμε με κατανάλωση (δεν συμφέρει να μετακινηθούμε μόλις φάμε Q σοκολατάκια), μπορούμε να ασχοληθούμε με τα υπο-προβλήματα $c[i,j]$ όπου όπου $c[i,j]$ ο ελάχιστος χρόνος που χρειαζόμαστε για να φάμε i σοκολατάκια, τρώγοντας στο τέλος από το κουτί j.

Αναδρομική Σχέση

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

Αρχή βελτιστότητας: Αν η βέλτιστη ακολουθία κουτιών που τρώμε, για να φάμε i σοκολατάκια καταλήγοντας στο κουτί j είναι k_a, k_b, k_o, k_i, η βέλτιστη ακολουθία για να φάμε $i - q_j$ σοκολατάκια καταλήγοντας στο κουτί k_o είναι k_a, k_b, k_o.

Άσκηση 3: Σοκολατάκια

Λύση:

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

- Τρέχουμε για q από 1 μέχρι Q. Σε κάθε βήμα, για κάθε κουτί j υπολογίζουμε το πόσο χρειαζόμαστε για να φάμε $q + q_j$ σοκολατάκια φτάνοντας στο j.
- Αφού $q_i > 0, \forall i$, αν αρχικοποιήσουμε σωστά τις αρχικές καταστάσεις, υπολογίζουμε τα υποπροβλήματα με τη σωστή σειρά.
- Αρχικοποίηση: $c[i][j] = \infty$ και $\forall j \in n, c[\min\{q_j, Q\}][j] = |p - j|$
- Επιστρέφουμε το $\min_{j \in \{1,...,n\}} c[Q][j]$.

Άσκηση 3: Σοκολατάκια

Λύση:

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

- Τρέχουμε για q από 1 μέχρι Q. Σε κάθε βήμα, για κάθε κουτί j υπολογίζουμε το πόσο χρειαζόμαστε για να φάμε $q + q_j$ σοκολατάκια φτάνοντας στο j.
- Αφού $q_i > 0, \forall i$, αν αρχικοποιήσουμε σωστά τις αρχικές καταστάσεις, υπολογίζουμε τα υποπροβλήματα με τη σωστή σειρά.
- Αρχικοποίηση: $c[i][j] = \infty$ και $\forall j \in n, c[\min\{q_j, Q\}][j] = |p - j|$
- Επιστρέφουμε το $\min_{j \in \{1,...,n\}} c[Q][j]$.

Άσκηση 3: Σοκολατάκια

Λύση:

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

- Τρέχουμε για q από 1 μέχρι Q. Σε κάθε βήμα, για κάθε κουτί j υπολογίζουμε το πόσο χρειαζόμαστε για να φάμε $q + q_j$ σοκολατάκια φτάνοντας στο j.
- Αφού $q_i > 0, \forall i$, αν αρχικοποιήσουμε σωστά τις αρχικές καταστάσεις, υπολογίζουμε τα υποπροβλήματα με τη σωστή σειρά.
- Αρχικοποίηση: $c[i][j] = \infty$ και $\forall j \in n, c[\min\{q_j, Q\}][j] = |p - j|$
- Επιστρέφουμε το $\min_{j \in \{1,...,n\}} c[Q][j]$.

Άσκηση 3: Σοκολατάκια

Λύση:

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

- Τρέχουμε για q από 1 μέχρι Q. Σε κάθε βήμα, για κάθε κουτί j υπολογίζουμε το πόσο χρειαζόμαστε για να φάμε $q + q_j$ σοκολατάκια φτάνοντας στο j.
- Αφού $q_i > 0, \forall i$, αν αρχικοποιήσουμε σωστά τις αρχικές καταστάσεις, υπολογίζουμε τα υποπροβλήματα με τη σωστή σειρά.
- Αρχικοποίηση: $c[i][j] = \infty$ και $\forall j \in n, c[\min\{q_j, Q\}][j] = |p - j|$
- Επιστρέφουμε το $\min_{j \in \{1,...,n\}} c[Q][j]$.

Άσκηση 3: Σοκολατάκια

Λύση:

$$c[min\{q_j + q, Q\}][j] = \min_{\substack{\{i | i \in \{1,..,n\}, \\ q_i < q_j, \\ t_i \neq t_j\}}} \{c[q][i] + |i - j|\}$$

- Τρέχουμε για q από 1 μέχρι Q. Σε κάθε βήμα, για κάθε κουτί j υπολογίζουμε το πόσο χρειαζόμαστε για να φάμε $q + q_j$ σοκολατάκια φτάνοντας στο j.
- Αφού $q_i > 0, \forall i$, αν αρχικοποιήσουμε σωστά τις αρχικές καταστάσεις, υπολογίζουμε τα υποπροβλήματα με τη σωστή σειρά.
- Αρχικοποίηση: $c[i][j] = \infty$ και $\forall j \in n, c[\min\{q_j, Q\}][j] = |p - j|$
- Επιστρέφουμε το $\min_{j \in \{1,\ldots,n\}} c[Q][j]$.

Άσκηση 3: Σοκολατάκια

Πολυπλοκότητα

$O(Qn^2)$ για την επίλυση των $Q * n$ υποπροβλημάτων του $c[i][j]$ (n συγκρίσεις στο καθένα) + $O(n)$ για την εύρεση του ελαχίστου $c[Q][j] =$

$$O(Qn^2)$$

.

Ορθότητα: Αφού σε κάθε επανάληψη, για κάθε κουτί j μελετάμε ως πιθανά 'προηγούμενα' μόνο κουτιά i με q_i αυστηρά μικρότερο του q_j, οποιαδήποτε αλληλουχία κουτιών για κάθε υποπρόβλημα του $c[i][j]$ είναι αδύνατον να περιέχει κύκλο (να χρησιμοποιεί το ίδιο κουτί πάνω από μια φορά).

Πολυπλοκότητα

$O(Qn^2)$ για την επίλυση των $Q * n$ υποπροβλημάτων του $c[i][j]$ (n συγκρίσεις στο καθένα) + $O(n)$ για την εύρεση του ελαχίστου $c[Q][j] =$

$$O(Qn^2)$$

.

Ορθότητα: Αφού σε κάθε επανάληψη, για κάθε κουτί j μελετάμε ως πιθανά 'προηγούμενα' μόνο κουτιά i με q_i αυστηρά μικρότερο του q_j, οποιαδήποτε αλληλουχία κουτιών για κάθε υποπρόβλημα του $c[i][j]$ είναι αδύνατον να περιέχει κύκλο (να χρησιμοποιεί το ίδιο κουτί πάνω από μια φορά).

Outline

Άσκηση 4: Απόσταση Επεξεργασίας για Αριθμητικές Εκφράσεις

Είσοδος: Μια συμβολοσειρά $x \in \{0, 1, .., 9, (,), +, -, \times, /\}^*$.
Έξοδος: Η απόσταση επεξεργασίας $d_L(x, E_S)$ της x από την γλώσσα *απλών αριθμητικών εκφράσεων* E_S, η οποία ορίζεται στο αλφάβητο $\{0, 1, .., 9, (,), +, -, \times, /\}$ και περιέχει τις συμβολοσειρές που προκύπτουν από τον αναδρομικό ορισμό:

- Αν $x \in \{1|2|...|9\}\{0|1|...|9\}^*|0$, τότε $x \in E_S$.
- Αν $x \in E_S$ τότε $(x) \in E_S$.
- Αν $x, y \in E_S$ τότε $x + y, x - y, x/y, x \times y \in E_S$.

Άσκηση 4: Απόσταση Επεξεργασίας για Αριθμητικές Εκφράσεις

Βασική ιδέα: Η συμβολοσειρά x αποτελείται από πέντε ήδη χαρακτήρων: 0, ψηφία διάφορα του μηδενός (d), $(,)$ και πράξεις ($\diamond$). Δηλαδή, το αλφάβητο μας είναι ουσιαστικά το $\{0, d, (,), \diamond\}$.

Παρατηρούμε ότι μπορούμε να αναλύσουμε αναδρομικά μια συμβολοσειρά $y \in E_S$ με $|y| = n$ ως εξής: η y θα είναι είτε το σύμβολο 0, είτε μια σειρά ψηφίων με $y_1 \neq 0$, είτε της μορφής $z \diamond l$ με $y_k = \diamond$ και $z, l \in E_S$, είτε της μορφής (z), με $y_1 = (, y_n =)$ και $z \in E_S$.
Άρα, οι πιθανές ελάχιστες διορθώσεις περιλαμβάνουν τις περιπτώσεις είτε της διαγραφής στοιχείων που δεν είναι ψηφία, για τη δημιουργία αριθμού, είτε την προσθήκη εξωτερικών παρενθέσεων δεδομένου ότι η εσωτερική συμβολοσειρά έχει προκύψει με ελάχιστο τρόπο, είτε την προσθήκη μιας πράξης ανάμεσα σε δυο συμβολοσειρές που έχουν προκύψει με ελάχιστο τρόπο.

Άσκηση 4: Απόσταση Επεξεργασίας για Αριθμητικές Εκφράσεις

Λύση: Με Δ.Π.: Έστω $dist[i,j]$ το κόστος μετατροπής της υποσυμβολοσειράς $x_{i,j}$ $(i \leq j)$. Φτιάχνουμε την αναδρομική σχέση:

Αναδρομική σχέση

$$dist[i,j] = min \begin{cases} |ND_{i+1,j}|, & x_i = d \\ |ND_{i+1,j}| + 1, & x_i \neq d \\ dist[i+1, j-1], & x_i = (, x_j =) \\ dist[i+1, j-1] + 1, & x_i \neq (, x_j =) \vee x_i = (, x_j \neq) \\ dist[i+1, j-1] + 2, & x_i \neq (, x_i \neq) \\ \min\limits_{k \in [i,j] \wedge x_k = \diamond} \{dist[i, k-1] + dist[k+1, j]\} \\ \min\limits_{k \in [i,j] \wedge x_k \neq \diamond} \{dist[i, k-1] + dist[k+1, j] + 1\} \\ \min\limits_{k \in [i,j]} \{dist[i, k] + dist[k, j] + 1\} \end{cases}$$

Άσκηση 4: Απόσταση Επεξεργασίας για Αριθμητικές Εκφράσεις

- Όπου $|ND_{i+1,j}|$ το πλήθος των συμβόλων που δεν είναι ψηφία από την θέση $i+1$ ως την θέση j της συμβολοσειράς ($ND_{i+1,j} = \{x_l | l \in \{i+1, .., j\} \wedge x_l \in \{\diamond, (,)\}$). Επιπλέον $dist[i, j] = 1$ για $i > j$ (κενή συμβολοσειρά).
- Παρατηρήστε ότι οι δυο πρώτες συνθήκες της αναδρομής αφορούν σε χειρισμό αριθμών, οι τρεις επόμενες σε χειρισμό παρενθέσεων και οι τρεις τελευταίες σε χειρισμό πράξεων (δηλ, πχ το κόστος του να προσθέσουμε μια πράξη σε κάποια θέση ή του να αποφασίσουμε να 'σπάσουμε' τη συμβολοσειρά σε δυο που χωρίζονται από μια πράξη).
- Επιπλέον, δεν χρειάζονται άλλες αρχικοποιήσεις για τις βασικές καταστάσεις. Προσέξτε ότι αν μια υπο-συμβολοσειρά είναι αριθμός που δεν αρχίζει από 0 η πρώτη συνθήκη θα δώσει 0, ενώ αν αρχίζει από 0, η πρώτη συνθήκη θα δώσει 1.

Άσκηση 4: Απόσταση Επεξεργασίας για Αριθμητικές Εκφράσεις

- Προσέξτε επίσης ότι η έκτη συνθήκη 'φροντίζει' να σπάσει την συμβολοσειρά σε δυο μέρη, αν χωρίζονται από μία πράξη που δεν συμφέρει να αντικατασταθεί από τις πρώτες δυο συνθήκες.
- Τέλος, παρατηρήστε ότι δεν χρειάζεται να αποθηκεύσουμε τις αλλαγές που χρειάζονται, παρά μόνο το κόστος τους. Π.χ,αν $x_{i+1,j-1} \in E_S$ και $x_j =), x_i \neq (,$ ξέρουμε ότι μπορούμε να κάνουμε αλλαγή κόστους 1 στην $x_{i,j}$ ώστε να ανήκει στην E_S. Δεν έχει σημασία αν έγινε με insertion η replacement.

Πολυπλοκότητα

Το ζητούμενο είναι στην $dist[1, n]$, όπου $n = |x|$ (προσέξτε ότι δεν χρειάζεται να υπολογίσουμε θέσεις $dist[i, j]$ όπου $i > j$). Έτσι, αφού σε κάθε κλήση της αναδρομής χρεαζόμαστε χρόνο $\mathcal{O}(n)$ συνολικά έχουμε πολυπλοκότητα τάξης $\mathcal{O}(n^3)$.

Outline

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Είσοδος: Δυαδικό δέντρο $T = (V, E)$ με $|V| = n$ κορυφές, ρίζα r του δέντρου και ακέραιος αριθμός k με $1 \leq k \leq n$.

Έξοδος: Ένα υποσύνολο $K \subseteq V$ των κορυφών του δέντρου με πλήθος στοιχείων $|K| = k$ που θα ελαχιστοποιεί τη μέγιστη απόσταση των κορυφών του δέντρου από τον κοντινότερο προγονό τους στο K. Συγκεκριμένα, θέλουμε να ελαχιστοποιήσουμε το κόστος $\text{cost}(K) = \max_{v \in V} \text{cost}(v, K)$ με

$$\text{cost}(v, K) = \begin{cases} 0, & v \in K \\ \infty, & v = r \text{ και } r \notin K \\ \text{cost}(\text{par}(v), K) + 1, & \text{αλλιώς} \end{cases}$$

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Λύση με Δυναμικό Προγραμματισμό: Έστω $f(v, d, i)$ το ελάχιστο κόστος κάλυψης από σύνολο μεγέθους i του υποδέντρου με ρίζα το v η οποία έχει απόσταση d.

Αναδρομική Σχέση

$$f(v, d, i) = \min\{A, B\}$$

A: να ανήκει η κορυφή v στο K

B: να μην ανήκει η κορυφή v στο K

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Αναδρομική Σχέση

Το κόστος του υποδέντρου με ρίζα την κορυφή v αν αυτή ανήκει στο K

$$A = \min_{j=0,\ldots,i-1} \{\max\{f(ch_l(v), 1, j), f(ch_r(v), 1, i-1-j)\}\}$$

Για το Α: Η λύση για το δέντρο είναι το μέγιστο κόστος των δύο υποδέντρων (όπου οι ρίζες τους απέχουν 1 από το σύνολο) για την καλύτερη μοιρασιά των $i-1$ εναπομηνάντων στοιχείων του συνόλου κάλυψης.

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Αναδρομική Σχέση

Το κόστος του υποδέντρου με ρίζα την κορυφή v αν αυτή ΔΕΝ ανήκει στο K

$$B = \min_{j=0,\ldots,i} \{\max\{f(ch_l(v), d+1, j), f(ch_r(v), d+1, i-j), d\}\}$$

Για το B: Η λύση για το δέντρο είναι το μέγιστο κόστος μεταξύ της ρίζας (δηλ d) και των λύσεων των δύο υποδέντρων (όπου οι ρίζες τους απέχουν $d+1$ από το σύνολο), για την καλύτερη μοιρασιά των i στοιχείων του συνόλου κάλυψης.

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Αναδρομική Σχέση

$$f(v,d,i) = \min\{A, B\}$$

$$A = \min_{j=0,\ldots,i-1}\{\max\{f(ch_l(v),1,j), f(ch_r(v),1,i-1-j)\}\}$$

$$B = \min_{j=0,\ldots,i}\{\max\{f(ch_l(v),d+1,j), f(ch_r(v),d+1,i-j), d\}$$

Αρχικοποίηση: Για v φύλλο: $f(v,d,i) = \begin{cases} d, & i = 0 \\ 0, & i > 0 \end{cases}$.

Σειρά των υπολογισμών: Ξεκινώντας από τα φύλλα, για κάθε επίπεδο υπολογίζουμε το f για κάθε $d \in \{0,1,\ldots,n-1\}$ και για κάθε $i \in \{0,1,\ldots,k\}$.Το ζητούμενο κόστος είναι $f(r,0,k)$, όπου στη ρίζα εκτελείται μόνο ο κανόνας A.

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Αναδρομική Σχέση

$$f(v,d,i) = \min\{A, B\}$$

$$A = \min_{j=0,\ldots,i-1}\{\max\{f(ch_l(v),1,j), f(ch_r(v),1,i-1-j)\}\}$$

$$B = \min_{j=0,\ldots,i}\{\max\{f(ch_l(v),d+1,j), f(ch_r(v),d+1,i-j), d\}$$

Τι επιστρέφει ο αλγόριθμος: Μέχρι στιγμής περιγράψαμε πως υπολογίζεται το βέλτιστο κόστος κάλυψης. Για να επιστρέψουμε το **σύνολο** κάλυψης αρκεί να αποθηκεύουμε επιπλέον (π.χ. σε έναν δεύτερο πίνακα $g(v,d,i)$) αν η καλύτερη επιλογή ήταν η κορυφή v να ανήκει στο σύνολο κάλυψης ή όχι καθώς και το καλύτερο μοίρασμα των i κορυφών στα υποδέντρα.

Άσκηση 5: Καλύπτοντας ένα Δέντρο

Πολυπλοκότητα

Έχουμε τους βρόχους επανάληψης:

$$\left.\begin{array}{l}\text{Για κάθε επίπεδο } l \text{ του δέντρου} \\ \quad \text{Για κάθε κορυφή } v \text{ στο επίπεδο } l\end{array}\right\} \Rightarrow \Theta(n)$$

$$\text{Για κάθε } d \in \{0, 1, \ldots, n-1\} \} \Rightarrow \Theta(n)$$

$$\text{Για κάθε } i \in \{0, 1, \ldots, k\} \} \Rightarrow \Theta(k)$$

$$\text{Υπολόγισε } f(v, d, i) \} \Rightarrow \Theta(k)$$

Άρα εκτελείται σε $O(n^2 k^2)$.

Άσκηση 5: Καλύπτοντας ένα Δέντρο (β)

Θα περιγράψουμε αλγόριθμο για την εξής εργασία:

Είσοδος: Δυαδικό δέντρο $T = (V, E)$ με $|V| = n$ κορυφές, ρίζα r του δέντρου και ακέραιος αριθμός z με $z \leq n$.

Έξοδος: Ελάχιστου μεγέθους συνόλο $K \subseteq V$ τέτοιο ώστε $\text{cost}(K) \leq z$.

Άσκηση 5: Καλύπτοντας ένα Δέντρο (β)

Greedy αλγόριθμος

Μέχρι να καλύψουμε το δέντρο:

- Ξεκινώντας από ένα φύλλο στο χαμηλότερο επίπεδο ανεβαίνουμε στο δέντρο μέχρι την κορυφή-πρόγονο σε απόσταση z.
- Προσθέτουμε την κορυφή αυτή στο σύνολο K.
- Διαγράφουμε το υποδέντρο με ρίζα την κορυφή αυτή.

Πολυπλοκότητα: Διερχόμαστε από κάθε κορυφή σταθερό πλήθος φορών, επομένως έχουμε χρονική πολυπλοκότητα $O(n)$.

Άσκηση 5: Καλύπτοντας ένα Δέντρο (β)

Απόδειξη Ορθότητας: Με επαγωγή που χρησιμοποιεί τα παρακάτω.

Αρχή Βελτιστότητας Υπολύσεων: Αν έχουμε μια βέλτιση λύση K^* για όλο το δέντρο T και ορίσουμε το T' ως το δέντρο που προκύπτει από το T αν αφαιρέσουμε το υποδέντρο που βρίσκεται κάτω από μια κορυφή που ανήκει στο σύνολο K, τότε τα εναπομείναντα στοιχεία του K καλύπτουν βέλτιστα το T'.
Άπληστη Επιλογή: Έστω v η κορυφή που διαλέγει πρώτη ο αλγόριθμός μας. Θα δείξουμε ότι υπάρχει βέλτιστη λύση που περιέχει την v:
Έστω ότι η K^* είναι βέλτιστη λύση που δεν περιέχει την v. Τότε αναγκαστικά θα περιέχει κάποια κορυφή-απόγονο της v. Έστω v' αυτή η κορυφή. Τότε, μπορούμε να κατασκευάσουμε λύση με ίσο (ή μικρότερο) πλήθος κορυφών κάλυψης απλά αντικαθιστώντας την v' με την v στο σύνολο κάλυψης.

Άσκηση 5: Καλύπτοντας ένα Δέντρο (β)

Θεωρώντας ως μαύρο κουτί τον αλγόριθμο $A(z)$ του δεύτερου ερωτήματος βρίσκουμε λύση για το πρώτο:

Κάνουμε δυαδική αναζήτηση ως προς z για να βρούμε το βέλτιστο κόστος z^* για το οποίο ο αλγόριθμος του δεύτερου ερωτήματος επιστρέφει σύνολο μεγέθους το πολύ k. Τυπικά ψάχνουμε το z^* τ.ω.

$$\forall z \geq z^* \quad A(z) \leq k$$
$$\forall z < z^* \quad A(z) > k$$

Αυτό είναι εφικτό γιατί το μέγεθος του βέλτιστου συνόλου είναι φθίνουσα συνάρτηση του z.

Πολυπλοκότητα: $O(n \log n)$ γιατί έχουμε $O(n)$ για τον αλγόριθμο που τρέχει σε κάθε ένα από τα $O(\log n)$ βήματα της δυαδικής αναζήτησης.