# Αλγόριθμοι - Εργασία 2

Δεληγιαννάκης Χαράλαμπος
4383

Μάιος 2024

# 1 Πρώτη άσκηση

Καταστευάστηκε άπλητσος αλγόριθμος που επιλύει το πρόβλημα. Σύμφωνα με αυτόν:

1. Αρχικοποίησε δομή που να αποθηκεύει το μονοπάτι της λύσης - visited (π.χ λίστα).
2. Αρχικοποίησε μεταβλητές που αφορούν την τρέχουσα κατάσταση (τρέχον χιλιόμετρο: $xCurrent$, τρέχουσες προμήθειες: $kCurrent$).
3. Ταξινόμηση των δεδομένων με βάση τις προμήθειες κάθε σημείου (σε φθίνουσα σειρά). (Τα δεδομένα θα μπορούσαν να είχαν μπει σε έναν πίνακα σαν δυάδες $(x, k)$ για να είναι ευκολότερη η ταξινόμηση)
4. Όσο δεν μπορείς να φτάσεις απευθείας από την τρέχουσα θέση στην τελική θέση $X$: (δηλαδή, έλεγχξε αν από το τρέχον σημείο μπορείς να φτάσεις στο τελικό με τις τρέχουσες προμήθειες)
   (a) Επίλεξε το σημείο με τη μεγαλύτερη προμήθεια, το οποίο πρέπει να είναι εντός εμβέλειας. 'Εστω ότι αυτό είναι το $\{x_i, k_i\}$. Αν δεν υπάρχει τέτοιο σημείο επίστρεψε -1 (άλυτο πρόβλημα).
   (b) Πήγαινε στο σημείο, πρόσθεσε το index του σε κάποια δομή που αποθηκεύει το μονοπάτι της λύσης - visited και ενημέρωσε τις μεταβλητές που αφορούν την τρέχουσα κατάσταση (αύξησε το τρέχον χιλιόμετρο κατά $x_i$, μείωσε τις τρέχουσες προμήθειες κατά $x_i$ αλλά και αύξησές τες κατά $k_i$.
5. Επίστρεψε το μονοπάτι της λύσης - visited και το μέγεθός του ως το πλήθος θέσεων που θα περάσει.

Σχετικά με την ορθότητα του αλγορίθμου, ο αλγόριθμος επιλέγει το σημείο με τις περισσότερες προμήθειες που είναι εντός της εμβέλειάς του. Αυτό σημαίνει ότι σε κάθε βήμα, διασφαλίζει ότι αυξάνει τις διαθέσιμες προμήθειες όσο το δυνατόν περισσότερο, δίνοντάς του τη μέγιστη δυνατή εμβέλεια για τα επόμενα βήματα. Επίσης, η επιλογή αυτή είναι τοπικά βέλτιστη και οδηγεί σε μια συνολικά βέλτιστη λύση γιατί μεγιστοποιεί τις προμήθειες που έχουμε διαθέσιμες για τα επόμενα βήματα. Μπορούμε να αποδείξουμε την ορθότητα με τη λογική ότι "Ο άπληστος αλγόριθμος υπερτερεί":

Δηλαδή, έστω ότι βρίσκουμε το σημείο $x_i$ με τις περισσότερες προμήθειες $k_i$ εντός της τρέχουσας εμβέλειάς μας. Αντί να επιλέξουμε το $x_i$, υποθέτουμε ότι επιλέγουμε ένα άλλο σημείο $x_j$ με λιγότερες προμήθειες $k_j$. Επιλέγοντας το $x_j$, αφήνουμε προμήθειες $k_i$ που είναι διαθέσιμες στο $x_i$. Αυτό μειώνει τις συνολικές διαθέσιμες προμήθειές μας για τα επόμενα βήματα, και συνεπώς μειώνει την πιθανότητα να φτάσουμε στον προορισμό. Συνεπώς, η

επιλογή του $x_i$ με τις περισσότερες προμήθειες είναι τοπικά βέλτιστη, και αφού μιλάμε για απληστεία, η τελική λύση θα κατασκευαστεί μυωπικά από τις τοπικές βέλτισες λύσεις.

Σχετικά με την πολυπλοκότητα του αλγορίθμου, αν $n$ είναι ο αριθμός των ζευγαριών (χιλιόμετρο-προμήθειες), η προταξινόμηση μπορεί να γίνει σε $O(nlogn)$ με έναν αποδοτικό αλγόριθμο όπως ο Merge Sort, ενώ η επανάληψη γίνεται σε $O(n)$. Επομένως, η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(nlogn + n)$, δηλαδή $O(nlogn)$.

## 2 Δεύτερη άσκηση

Καταστευάστηκε αλγόριθμος με τη μέθοδο δυναμικού προγραμματισμού που επιλύει το πρόβλημα. Με βάση την αρχή του ΔΠ, θα δημιουργήσουμε υποπροβλήματα (εύρεση μικρότερου κόστους διάσπασης μιας μικρότερης συμβολοσειράς) και συνδυάζοντας τις λύσεις τους θα φτάσουμε στην τελική λύση. Σύμφωνα με τον αλγόριθμο:

1. Βάλε τα σημεία διάσπασης σε έναν πίνακα και ταξινόμησέ τον κατά αύξουσα σειρά, και πρόσθεσε το σημείο 0 στην αρχή και το σημείο n (μέγεθος συμβολοσειράς) στο τέλος του πίνακα.

2. Αρχικοποίησε με 0 έναν δισδιάστατο πίνακα (m x m) όπου m ο αριθμός των σημείων διάσπασης συμπεριλαμβανόμενων των άκρων 0 και $n$. Ονόμασε αυτόν τον πίνακα $OPT(i, j)$, ο οποίος στο κελί $(i, j)$ θα περιλαμβάνει το ελάχιστο κόστος διάσπασης συμβολοσειράς που ξεκινάει από το $i$ και τελειώνει με $j$. Τα σημεία διάσπασης για αυτό (αν υπάρχουν) είναι τα σημεία διάσπασης του αρχικού προβλήματος τα οποία βρίσκοντα ανάμεσα από $i$ και $j$.

3. Η εξίσωση Bellman που θα χρησιμοποιηθεί για τον υπολογισμό του παραπάνω κόστους (έστω k σημείο διάσπασης) είναι:

$$OPT(i, j) = (j - i) + min_{i<k<j}(OPT(i, k) + OPT(k, j))$$

4. Με αυτόν τον τρόπο, γεμίζουμε όλα τα δυνατά κελιά του πίνακα $OPT(i, j)$. Για να γίνει αυτός ο υπολογισμός, χρειαζόμαστε 3 εμφωλευμένες επαναλήψεις, μία για κάθε δυνατή απόσταση $j - i$, μια για κάθε δυνατό σημείο έναρξης συμβολοσειρας $i$, και μία για την μεταβλητή $k$ που βρίσκεται ανάμεσα από τα $i$ και $j$.

5. Έτσι, αφού έχουμε συνδυάσει λύσεις υποσυμβολοσειρών για να λύσουμε το ίδιο πρόβλημα για μια μεγαλύτερη υποσυμβολοσειρά (δηλαδή συνδυάζοντας τις λύσεις των υποπροβλημάτων), θα καταλήξουμε να έχουμε λύσει το αρχικό πρόβλημα και η τιμή του συνολικού ελάχιστου κόστους θα είναι η $OPT(0, n)$, δηλαδή το ελάχιστο κόστος για τη διάσπαση όλης της συμβολοσειράς, με σημεία διάσπασης αυτά που βρίσκοντα

ανάμεσα από το 0 και το $n$, δηλαδή όλα.

Σχετικά με τον ορθότητα του παραπάνω αλγορίθμου, εξασφαλίζεται από το γεγονός ότι υπολογίζουμε το ελάχιστο κόστος για κάθε υποσυμβολοσειρά σύμφωνα με τις στις προηγούμενες λύσεις, δηλαδή τις προηγούμενες τιμές του πίνακα $OPT$. Έτσι, εξασφαλίζουμε ότι όλες οι πιθανές τομές εξετάζονται και ότι το κόστος υπολογίζεται βέλτιστα.

Σχετικά με την πολυπλοκότητα του αλγορίθμου, στα βήματα του αλγορίθμου αναφέρεται ότι χρησιμοποιούμε 3 εμφωλευμένες επαναλήψεις όπου κάθε επανάληψη είναι $O(n)$, επομένως αφού δεν υπάρχει κάτι πιο επιβαρυντικό, η συνολική πολυπλοκότητα είναι $O(n^3)$.

# 3 Τρίτη Άσκηση

## 3.1 Πρώτο ζητούμενο

Κατασκευάστηκε αλγόριθμος με τη μέθοδο δυναμικού προγραμματισμού που επιλύει το πρόβλημα. Θα χρησιμοποιήσουμε δυσδιάστατο πίνακα $OPT[j][j]$ για να αποθηκεύσουμε για κάθε κελί το μέγιστο κέρδος (μέγιστο άθροισμα) με την επιλογή του κελιού μέχρι εκείνο το σημείο. Επομένως, το ζητούμενο του αλγορίθμου ουσιαστικά είναι το μεγαλύτερο κελί από τα κελία της τελευταίας σειράς του πίνακα $OPT[j][j]$, δηλαδή το $OPT[n][k]$ για κάποιο $0 \leq k \leq n$, καθώς αυτό θα δίνει το μεγαλύτερο κέρδος που προκύπτει με την επιλογή αυτού του κελιού, αφού έχουμε περάσει πρώτα και από όλες τις προηγούμενες σειρές (διασφαλίζοντας ότι δεν παραβιάζουμε κάποιον περιορισμό).

Η εξίσωση Bellman για την εύρεση τιμής στον πίνακα $OPT[n][n]$ είναι:

$$OPT[i][j] = V[i][j] + max_{k \neq j} OPT[i-1][k]$$

όπου ο V[n][n] είναι ο πίνακας που δίνεται από την εκφώνηση και αποθηκεύει το βάρους του κάθε κελιού ξεχωριστά. Η περιγραφή του αλγορίθμου είναι η εξής:

1. Αρχικοποιούμε τον πίνακα $OPT[n][n]$ και βάζουμε στην πρώτη σειρά του τις ίδιες τιμές με την πρώτη σειρά του $V[n][n]$. Οι υπόλοιπες σειρές θα έχουν παντού 0.

2. Διατρέχουμε κάθε σειρά του πίνακα $OPT[n][n]$ ξεκινώντας από τη δεύτερη μέχρι και την τελευταία $(1 \leq i \leq n-1)$.

3. Για κάθε σειρά $OPT[i][n]$, διατρέχουμε και τις στήλες της με εμφωλευμένο βρόγχο $(0 \leq j \leq n-1)$.

4. Αρχικοποιούμε με -1 μία μεταβλητή $maximumValue$ που αντιπροσωπεύει την τιμή $max_{k \neq j} OPT[i-1][k]$ στην εξίσωση Bellman.

5. Για να την υπολογίσουε ξεκινάμε έναν τρίτο βρόγχο (εμφωλευμένο) που θα διατρέχει τις στήλες τις προηγούμενης σειράς (αν υπάρχει) εκτός από την στήλη $j$, διασφαλίζοντας το με ένα $if - statement$. $(0 \leq k \leq n-1$ για καθε $k \neq j)$

6. Θα υπολογίζουμε σε κάθε επανάληψη την μεταβλητή με τον εξής τρόπο:

$$maximumValue = max(maximumValue, OPT[i-1][k])$$

   ώστε να έχει τη μέγιστη τιμή της προηγούμενης σειράς εκτός από το κελί που βρίσκεται στη στήλη $j$.

7. Αφού τελειώσει ο τρίτος βρόγχος για το $k$, αναθέτουμε $OPT[i][j] = V[i][j] + maximumValue$.

8. Αφού τελειώσουν όλες οι επαναλήψεις, επιστρέφουμε την μέγιστη τιμή της τελευταίας σειράς του πίνακα $OPT[n][n]$.

Υποσημείωση: μπορούμε να χρησιμοποιήσουμε μια δομή για κρατάμε για κάθε κελί το path που ακολουθήσαμε για να υπολογίσουμε το μέγιστο κέρδος του.

Σχετικά με την ορθότητα, εξετάζονται κατάλληλα όλες οι δυνατές επιλογές, δηλαδή ελέγχονται τα αποτελέσματα για κάθε κελί χωρίς να παραβιάζεται ο περιορισμός. Ταυτόχρονα, υπολογίζει τη βέλτιστη λύση με τη βοήθεια της μεταβλητής $maximumValue$. Επομένως, είναι ο ορθός.

### 3.2 Δεύτερο ζητούμενο

Σχετικά με την πολυπλοκότητα του αλγορίθμου δυναμικού προγραμματισμού, έχουμε μία αρχικοποίηση (δύο αν έχουμε και δομή για το path) ενός δισδιάστατου πίνακα - $O(n^2)$, τρεις εμφωλευμένες επαναλήψεις του δισδιάστακου πίνακα - $O(n^3)$ και την εύρεση μεγίστου της τελευταίας γραμμής - $O(n)$. Άρα, η συνολική πολυπλοκότητα είναι $O(n^2 + n^3 + n)$, δηλαδή $O(n^3)$.

Σχετικά με την πολυπλοκότητα του άπλησ του αλγορίθμου με προταξινόμηση του δυσδιάστατου πίνακα σε φθίνουσα σειρά (κάνοντας αλλαγές παράλληλα και σε δεύτερο πίνακα που θα έχει τις συντεταγμένες κάθε κελιού για να μην χαθεί η σειρά), θα χρειαστούμε $O(n^2 log^2 n)$, δηλαδή $O(2n^2 logn)$, άρα $O(n^2 logn)$ για την ταξινόμηση χρησιμοποιώντας έναν γρήγρο αλγόριθμο όπως τον $mergesort$. Έπειτα θα διατρέξουμε κάθε κελί του πίνακα - $O(n^2)$ για να επιλέξουμε τις τιμές που πληρούν τα κριτήρια. Ο έλεγχος αν ένα κελί πληρεί τα κριτήρια γίνεται σε $O(1)$. Επομένως, η συνολική πολυπλοκότητα είναι $O(n^2 logn)$.

### 3.3 Τρίτο ζητούμενο

Ο άπληστος αλγόριθμος (όπως οι περισσότεροι άπληστοι αλγόριθμοι γενικά) δεν είναι σωστός για όλες τις πιθανές εισόδους. Για παράδειγμα, αν δοθεί ο παρακάτω πίνακας-σκακιέρα όπου κάθε κελί έχει για τιμή το βάρος του:

$$V = \begin{bmatrix} 3 & 5 \\ 1 & 4 \end{bmatrix}$$

τότε, θα ταξινομήσει τον πίνακα ως εξής [5, 4, 3, 1], θα επιλέξει πρώτα το μέγιστο στοιχείο (5) και μετά το αμέσως επόμενο μέγιστο στοιχείο που δεν παραβιάζει τους περιορισμούς (1), άρα θα παράγει έξοδο 5+1. Παρ'όλα αυτά, η σωστή έξοδος για το συγκεκριμένο πρόβλημα είναι 3+4 = 7. Επομένως, ο greedy αλγόριθμος δε βρίσκει πάντα τη βέλτιστη λύση.