Γιώργος Αγγέλης
el18030
ΣΗΜΜΥ ΕΜΠ
2021

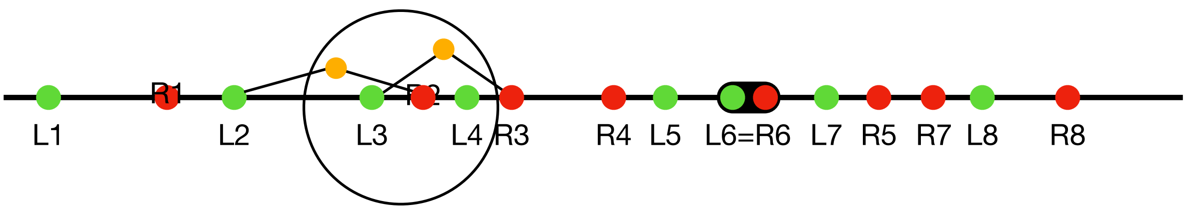
# Αλγόριθμοι και Πολυπλοκότητα Σειρά 2

## Άσκηση 1

### Αλγόριθμος

Με κέντρο κάθε σημείο i διαγράφουμε έναν κύκλο ακτίνας r και σημειώνουμε τα σημεία τομής του κύκλου αυτού με την ευθεία έστω Li το αριστερό και Ri το δεξί. Αν το σημείο i απέχει r από την ευθεία τότε Li=Ri.

Συνεπώς στην ευθεία έχουμε μία αλληλουχία left και right σημείων, π.χ.:




Ορίζουμε δύο κενά sets visited[ ] και fullset[ ] και έναν counter=0.

Μέχρις ότου όλα τα σημεία μπουν στο fullset[ ]:

> Ξεκινάμε από αριστερά προς τα δεξιά. Επιλέγουμε κάθε σημείο διαδοχικά. Αναζητούμε αποδεκτή αλληλουχία L-R.
>
> <u>Αν το σημείο που επιλέξαμε πρόκειται για L:</u>
>
> > Το i του L προστίθεται στο visited[ ].
> >
> > Πάμε στο επόμενο σημείο.
>
> <u>Αν το σημείο που επιλέξαμε πρόκειται για R:</u>
>
> > Σε περίπτωση που το αμέσως προηγούμενο σημείο είναι R, έχουμε διαδοχικά R-R οπότε πάμε στο επόμενο σημείο.
> >
> > Σε περίπτωση που το αμέσως προηγούμενο σημείο είναι L, έχουμε διαδοχικά L-R. Αν το k του Rk δεν ανήκει στο fullset, έχουμε L-R, αυτό είναι το ζητούμενο, counter+=1, στη fullset[ ] προστίθενται όλα τα στοιχεία της visited[ ] και κατόπιν η visited[ ] αδειάζει.
> >
> > Αν το k του Rk ανήκει στο fullset αγνοούμε πλήρως το Rk.
> >
> > Πάμε στο επόμενο σημείο.

Ορθότητα:

Έστω ότι σε κάθε διάστημα ο αλγόριθμος μας μάς επιστρέφει το σημείο x, και έστω ο ιδανικός αλγόριθμος μας επιστρέφει το σημείο y, και έστω Sx και Sy το πλήθος των x και y αντίστοιχα. Προφανώς ισχύει Sx≥Sy. Αρκεί να δείξουμε Sx=Sy.
Η περίπτωση Sy > Sx δεν έχει νόημα να εξεταστεί.

Έστω Sy < Sx:

Αυτό σημαίνει ότι ο άπληστος αλγόριθμος μας αναγκάστηκε να τοποθετήσει ένα και παραπάνω επιπλέον σημεία σε κάποιο διάστημα ή διαστήματα. Ωστόσο αυτό δεν μπορεί να συμβεί διότι ο αλγόριθμος μας τοποθετεί τον ελάχιστο αριθμό δυνατών σημείων καθώς προσθέτει σημεία μόνο μεταξύ διαδοχικών L και R, εξασφαλίζοντας ταυτόχρονα ότι για σημεία που έχουν καλυφθεί ήδη δεν θα υπάρξει περιττή προσθήκη σημείου.

Οπότε Sy = Sx.

Είναι προφανές ότι ο Αλγόριθμος έχει πολυπλοκότητα O(n).

Άσκηση 2

α)

Θέλουμε να ελαχιστοποιήσουμε τους επιμέρους βεβαρημένους χρόνους εξυπηρετώντας όσο το δυνατόν περισσότερους πελάτες στο μικρότερο χρονικό διάστημα.

Διατάσσουμε τα $x_k = w_k/p_k$ , k ε 0,…,n σε φθίνουσα σειρά και προκύπτει ο πίνακας:

$A = [x_{min}, x_{min+1}, \ldots, x_i, \ldots, x_{max-1}, x_{max}]$ όπου $w_i$ και $p_i$ τα αντίστοιχα w, p που δίνουν το $x_i$.

Οπότε ξεκινάμε από το μεγαλύτερο $x_i=x_{max}$ και υπολογίζουμε τους βεβαρημένους χρόνους:

Προσοχή: το $w_{max}$ δεν είναι το max των w αλλά εκεινο το w που δίνει $x_{max}$!

$w_{max}p_{max} + w_{max-1}(p_{max-1} + p_{max}) + \ldots + w_i(p_i + \ldots + p_{max-1} + p_{max}) + \ldots + w_{min}(p_{min} + \ldots + p_{max})$

Πολυπλοκότητα O(nlogn).

Ορθότητα:

Έστω ότι μπορούσαμε για κάποιο πελάτη να υπάρξει καλύτερος βεβαρυμένο χρόνος $t_{opt}$, δηλαδή:

$t_{opt} < t$ =>

$w_{max,opt}p_{max,opt}+w_{max-1,opt}(p_{max-1,opt}+p_{max,opt})+\ldots+w_{i,opt}(p_{i,opt}+\ldots+p_{max-1,opt}+p_{max,opt})<w_{max}p_{max}+w_{max-1}(p_{max-1}+p_{max})+\ldots+w_i(p_i+\ldots+p_{max-1}+p_{max})$=>

που δεν γίνεται γιατί μετά την ταξινόμηση των $x_i$ κάθε επιμέρους $w_j(p_j + \ldots + p_{max})$ είναι το ελάχιστο.

β)

Για το δεύτερο πρόβλημα διατάσσουμε τα $x_k = w_k/p_k$ , k ε 0,…,n σε φθίνουσα σειρά και προκύπτει ο πίνακας:

$A = [x_{min}, x_{min+1}, \ldots, x_i, \ldots, x_{max-1}, x_{max}]$ όπου $w_i$ και $p_i$ τα αντίστοιχα w, p που δίνουν το $x_i$.
Χρησιμοποιούμε ως κεντρική ιδέα Knapsack problem και δημιουργούμε την αναδρομή:

$$MinTime(i,m) = \begin{cases} max_{1<=j<=i}\,(w_j \cdot p_j), & m = i \\ min(max_{m<=j<=i}(\ MinTime(j-1, m-1)\ + w_j \cdot \sum\limits_{m<=k<=j} p_k)\ ), & m > 1, i > 0 \end{cases}$$

Για ίδιο αριθμό πελατών και υπαλλήλων ο βεβαρυμένος χρόνος ισούται με τον μέγιστο χρόνο εξυπηρέτησης.

Άσκηση 3

α)

Ξεκινάμε από δεξιά επισκεπτόμενοι τα σημεία του πεζοδρομίου με συντεταγμένες:

$$0 < \underline{x1 < x2 < \cdot \cdot \cdot < xn} < L$$

Για κάθε σημείο i που επισκεπτόμαστε ψάχνουμε το ελάχιστο κόστος για ένα σκέπαστρο που ξεκινά από ένα σημείο πριν και τελειώνει στο $x_i$. Υπολογίζουμε το ελάχιστο κόστος με τον τύπο:

- $M(i) = min_{0 \le k \le i-1}((x_{i-1}-x_k)^2 + C + M(k-1))$
- $M(-1) = 0$

Και έτσι υπολογίζουμε το $M(x_n)$ σε $O(n^2)$.

β)

Για την απάντηση σε αυτό το ερώτημα αξιοποίησα την πηγή:

https://github.com/algoholics-ntua/algorithms/blob/master/Advanced%20Topics/Convex%20Hull%20Trick/dp_cht.pdf?fbclid=IwAR2s7-yVhjnHbpD6ZYB46fu7X3_1BDRJiDpmttSXiurXsNmQ0YxKHpP9we0

Η συγκεκριμένη επίλυση βασίζεται στην τεχνική Convex Hull Trick. Αρχικά γράφουμε την παραπάνω σχέση ως:

$M(i) = min_{0 \le k \le i-1}((x_{i-1}-x_k)^2 + C + M(k-1)) = x_{i-1}^2 + C + min_{0 \le k \le i-1}(x_k^2 - 2x_{i-1}x_k + M(k-1))$ και θέτουμε:

$a(k) = -2x_{i-1}x_k$
$b(k) = M(k-1) + x_k^2$

Οι κλήσεις $a_n$ των ευθειών αποτελούν φθίνουσα ακολουθία και οι τιμές $x_m$ στις οποίες υπολογίζουμε το ελάχιστο αποτελούν αύξουσα ακολουθία. Στο πρόβλημά μας με τα στέγαστρα ικανοποιούνται και οι δύο αυτές προϋποθέσεις. Για να βρούμε το ελάχιστο κόστος M αναζητούμε την ευθεία που δίνει το ελάχιστο x για κάθε x.

Το περίβλημα αποτελεί δυναμική δομή αφού κάθε φορά που βρίσκουμε κατάλληλη ευθεία αυτή προστίθεται στο περίβλημα.

Η εισαγωγή ευθείας στο κυρτό περίβλημα γίνεται με την προϋπόθεση ότι τα $a_n$ αποτελούν φθίνουσα ακολουθία, και άρα αναζητούμε αν η ευθεία τέμνει το περίβλημα σε σημείο πιο κάτω την προτελευταία ευθεία. Αν αυτό συμβαίνει η ευθεία προστίθεται στο περίβλημα.

Λόγω της ιδιότητας των a να είναι σε φθίνουσα σειρά, αν μία ευθεία απορριφθεί τότε δεν ξαναελέγχεται, και άρα συνολικά το περίβλημα δημιουργείται σε O(n), δηλαδή σε O(1) για κάθε ευθεία.

Επίσης σε O(1) γίνεται η εύρεση του x που δίνει ελάχιστο M, γιατί γνωρίζουμε σε ποιο σημείο του Convex Hull βρίσκεται το x, καθώς και ότι μεγαλύτερα x θα απορρίπτουν σίγουρα ευθείες που έχει απορρίψει το ίδιο το x.

Οπότε το πρόβλημα λύνεται γραμμικά.

Έτσι, με τα παραπάνω στοιχεία προσαρμοσμένα στο πρόβλημά μας μπορούμε να βρίσκουμε σε χρόνο O(n) εκείνα τα x που θα μας οδηγήσουν στη κατάλληλη διαμέριση των στεγάστρων.

Άσκηση 4

Για την απάντηση σε αυτό το ερώτημα αξιοποίησα την πηγή:

https://github.com/algoholics-ntua/algorithms/blob/master/Advanced%20Topics/Convex%20Hull%20Trick/dp_cht.pdf?fbclid=IwAR2s7-yVhjnHbpD6ZYB46fu7X3_1BDRJiDpmttSXiurXsNmQ0YxKHpP9we0

α)

Η συγκεκριμένη επίλυση βασίζεται στην επίλυση του προβλήματος Leaves στο παραπάνω PDF.

$$MinRate(i,k) = \begin{cases} 0\,, & i=k,\ k>0, i>0 \\ \sum_{1<=\lambda<=i} \sum_{\lambda<m<=i} A_{\lambda m}, & k=1, i>0 \\ min_{k<=j<=i}(MinRate(j-1,k-1) + \sum_{j<=\lambda<=i} \sum_{\lambda<=m<=i} A_{\lambda m}), & k>1, i>0 \end{cases}$$

- Για i = k έχουμε ίδιο αριθμό φοιτητών με λεωφορεία οπότε ο συνολικός δείκτης ευαισθησίας παίρνει τιμή 0.

- Για k = 1 έχουμε ένα λεωφορείο, άρα όλοι οι φοιτητές θα μπουν σε αυτό, άρα ο συνολικός δείκτης ευαισθησίας είναι το άθροισμα όλων των επιμέρους δεικτών ευαισθησίας που έχει ο κάθε φοιτητής ως προς κάθε άλλο φοιτητή. Επίσης η περίπτωση αυτή λειτουργεί και ως σχέση βάσης για την αναδρομή που ακολουθεί στην περίπτωση (k > 1, i > 0).

- Για (k > 1, i > 0) η σχέση:

$$min_{k<=j<=i}(MinRate(j-1,k-1) + \sum_{j<=\lambda<=i} \sum_{\lambda<=m<=i} A_{\lambda m}), \quad k>1, i>0$$

  υπολογίζει τον ελάχιστο συνολικό δείκτη ευαισθησίας για τις περιπτώσεις που:

  - j=i, δηλαδή ο τρέχων φοιτητής i μπει μόνος του στο τελευταίο λεωφορείο,

  - j=i-1, δηλαδή ο τρέχων φοιτητής i και ο i-1 μπουν στο τελευταίο λεωφορείο

    …

  - j=k, δηλαδή ο τρέχων φοιτητής i και οι υπόλοιποι i-k+1 μπουν στο τελευταίο λεωφορείο.

Ελέγχουμε για j>=k γιατί σε άλλη περίπτωση οδηγούμαστε στην τετριμμένη περίπτωση όπου έχουμε τουλάχιστον ένα λεωφορείο ανά φοιτητή. Και κάθε φορά που καλείται η MinRate( ) ο αριθμός των λεωφορείων μειώνεται προκειμένου αναδρομικά να φτάσουμε στο ένα λεωφορείο και από εκεί και πέρα με τις κατάλληλες συγκρίσεις της min( ) αναδρομικά να προκύψει το ζητούμενο. Εμείς αναζητούμε το MinRate(n,k).

Η πολυπλοκότητα είναι $O(kN^3)$ και προκύπτει από το γινόμενο $kN^2(N-k)$.

Η Ορθότητα του αλγορίθμου μας αποδεικνύεται ως εξής: Για κάθε φοιτητή i έχουν δημιουργηθεί όλες οι δυνατές q-άδες φοιτητών που έχουν εισαχθεί στα προηγούμενα λεωφορεία και έχει επιλεγεί εκείνος ο συνδυασμός που να οδηγεί στον ελάχιστο συνολικό δείκτη ευαισθησίας.

β)

$$MinRate(i,k) = \begin{cases} 0, & i = k,\ k = 1, i > 0 \\ \sum_{1<=\lambda<=i} \sum_{\lambda<m<=i} A_{\lambda m}, & k = 1, i > 0 \\ min_{k<=j<=i}(\ max(\ MinRate(j-1, k-1), \sum_{j<=\lambda<=i} \sum_{\lambda<=m<=i} A_{\lambda m})\ ), & k > 1, i > 0 \end{cases}$$

Ο αλγόριθμος δεν αλλάζει πολύ και η πολυπλοκότητα μένει η ίδια. Πλέον δεν αναζητάμε άθροισμα δείκτη ευαισθησίας αλλά συγκρίνουμε τους αντίστοιχους δείκτες μεταξύ των λεωφορείων εξού και το «+» αντικαταστάθηκε με «,». Μεταξύ των δεικτών ευαισθησίας των λεωφορείων αναζητούμε τον μέγιστο δείκτη και με το min( ) βρίσκουμε την ελάχιστη τιμή του δείκτη αυτου καθώς το j παίρνει τιμές από k μέχρι i.

Άσκηση 5

(α)

Θεωρούμε ένα συνεκτικό μη κατευθυνόμενο γράφημα G(V, E) με n κορυφές και m ακμές και T1, T2 δύο διαφορετικά συνδετικά δέντρα του G. Θα δείξουμε ότι για κάθε ακμή e ∈ T1 \ T2 , υπάρχει ακμή e′ ∈ T2 \T1, τέτοια ώστε το (T1 \{e})∪{e′} είναι συνδετικό δέντρο.

Αλγόριθμος:

Έστω T1, T2 και ακμή e={a,b}, e∈T1\T2. Συνεπώς T2∪e περιέχει κύκλο, έστω K. Υπάρχει ακμή e’ του K που δεν ανήκει στο T1, διότι αν όλες οι ακμές του K ανήκαν στο T1 το T1 θα περιείχε τον κύκλο K. Οπότε αποδεικνύεται ότι δέντρο (T1 \{e})∪{e′} είναι συνδετικό. Στο T2 βρίσκουμε το μονοπάτι που ενώνει τους κόμβους a, b με DFS και πολυπλοκότητα O(n) και με έλεγχο σε σταθερό χρόνο ελέγχουμε για κάθε ακμή του μονοπατιού αν αυτή ανήκει στο T1. Αν δεν ανήκει βρέθηκε η e’.

(β)

Θα δείξουμε ότι το H είναι συνεκτικό και ότι η απόσταση (στο H) μεταξύ δύο συνδετικών δέντρων T1 και T2 του G είναι ίση με |T1 \T2|.

Έστω ότι το H είναι μη-συνεκτικό. Τότε από τον ορισμό του H υπάρχουν τουλάχιστον δύο διαφορετικά συνδετικά δέντρα $T_n$ και $T_m$ του G που δεν διαφέρουν κατά καμία ακμή, άτοπο. Άρα H συνεκτικό.

Η απόδειξη ότι η απόσταση (στο H) μεταξύ δύο συνδετικών δέντρων T1 και T2 του G είναι ίση με |T1 \T2| θα γίνει επαγωγικά.

- |T1 \ T2| = 1 αν και μόνο αν d(T1,T2) = 1.

- Έστω ότι |T1\T2|=k αν και μόνο αν d(T1,T2) = k.

- Αρκεί να δείξουμε ότι |T1\T2| = k+1 αν και μόνο αν d(T1,T2) = k+1.

- Έστω e ∈ T1 \ T2
- Από το (α), υπάρχει e΄ ∈ T2 \ T1 τ.ω. T1΄=(T1\{e})∪{e΄} ∈ H
- Αν d(T1΄,T2) = k => d(T1,T2) = k+1
- Άρα, d(T1,T2) = k+1 και T1΄ ∈ H με d(T1,T1΄) = 1 και d(T1΄,T2) = k
- Άρα d(T1΄,T2) = k => |T1΄ \T2| = k => |T1\T2| = k+1

Για να υπολογίσουμε ένα συντομότερο μονοπάτι (στο H) μεταξύ των T1 και T2 δουλεύουμε όπως στο (α) ερώτημα:

Έστω ότι ξεκινάμε από το T2 και θέλουμε να φτάσουμε στο T1, τα οποία απέχουν απόσταση μεγαλύτερη του 1. Από το T2 βρίσκουμε το T΄ με το οποίο το T2 απέχει απόσταση 1. Με το (α) ερώτημα βρίσκουμε κάθε ακμή e ∈ T2 \ T΄ και την προσθέτουμε στο δέντρο. Συνεχίζουμε να εκτελούμε τον αλγόριθμο του (α) ερωτήματος για διαδοχικά δέντρα μέχρι να φτάσουμε στο δέντρο T1. Επειδή τα δύο δέντρα χωρίζονται από απόσταση k και κάθε φορά οδηγούμαι σε ένα δέντρο για το οποίο ο έλεγχος όλων των ακμών του γίνεται σε O(n) η συνολική πολυπλοκότητα είναι O(nk).

(γ)

Δημιουργούμε το MST με τον αλγόριθμο του Kruskal σε O(nlogm). Κατόπιν και για κάθε ακμή που δεν ανήκει στο MST:
-προσθέτουμε την ακμή στο MST και δημιουργείται ένας κύκλος
-αφαιρούμε την ακμή με το μεγαλύτερο βάρος που ανήκει στον κύκλο
-προκύπτει το MST

<u>Ορθότητα:</u>
Ο παραπάνω αλγόριθμος είναι εξαντλητικός ως προς την αναζήτηση των επιμέρους MSTs και από τον ορισμό του MST διασφαλίζεται ότι για κάθε ακμή η παραπάνω αλληλουχία ενεργειών θα δημιουργήσει όντως το αντίστοιχο MST.

ΤΕΛΟΣ