2η Σειρά Γραπτών Ασκήσεων Αλγορίθμων

Παπαρρηγόπουλος Θοδωρής
el18040

Άσκηση 1

Αρχικά έστω ότι έχουμε ένα σημείο (x,y) και ότι κάνουμε έναν κύκλο ακτίνας r στο (χ0,0). Τότε από εξισώσεις κύκλου πρέπει $(x-x_0)^2+y^2\leq r^2$ για το σημείο (x,y).
Συνεπώς, $x^2-2\cdot x\cdot x_0+x_0^2+y^2\leq r^2\Rightarrow x_0^2-(2\cdot x)x_0+(y^2-r^2)\leq 0$, και προκύπτει λύνοντας την ανυσότητα ένα εύρος σημείων που μπορούμε να σχεδιάσουμε κύκλο ακτίνας r και να καλύπτει το σημείο μας. $\Delta=4x^2-4\cdot(y^2-r^2)$ και $x_{0_{1,2}}=x\pm\sqrt{\frac{\Delta}{4}}\Rightarrow x-\sqrt{\frac{\Delta}{4}}\leq x_0\leq x+\sqrt{\frac{\Delta}{4}}$
Έτσι, ουσιαστικά το πρόβλημα μας είναι ισοδύναμο με το να βρούμε τα βέλτιστα σημεία που ικανοποιούν όσο τον δυνατόν περισσότερα τμήματα.

Σορτάρω λοιπόν από το start σημείο του κάθε διαστήματος.

Επειδή πρέπει να τοποθετηθούν όλα τα σημεία τότε πρέπει ένας κύκλος να πιάνει όσο τον δυνατότερο μεγαλύτερο αριθμό συμπεριλαμβάνοντας τον πρώτο (στην ακολουθία που κοιτάμε).

Για λόγους ευκολίας θα θεωρούμε ως αρχή του διαστήματος i τον συμβολισμό si και ως τέλος του το fi.

Για αυτό ξεκινώντας παίρνουμε το πρώτο σαν βέλτιστο διάστημα [s1, f1]

Τώρα, αν s2 > f1, τότε κρατάμε έναν κύκλο ήδη και συνεχίζουμε ξεκινόντας πάλι τον αλγόριθμο από το 2ο διάστημα-σημείο.

Αν s2 = f1 τότε, χρειάζεται ένας μόνο κύκλος για τα 2 σημεία ο οποίο θα περιλαβμάνει επίσης όσα άλλα διαστήματα ξεκινάνε από το f1=s2 και συνεπώς μεταπηδάμε στο επόμενο στοιχείο με s > f1.

Τέλος, αν s2 < f1 τότε έχουμε επικαλυπτόμενο διάστημα έτσι το βέλτιστο παράθυρο μας είναι το [s2, min(f1,f2)] και συνεχίζουμε την παραπάνω διαδικασία. Δηλαδή ψάχνουμε τις 3 περιπτώσεις για το s3.
Μόλις βρούμε το μέγιστο τέτοιο διάστημα αυξάνουμε τον αριθμό των συνολικών κύκλων και ξεκινώντας από το επόμενο στοιχείο.

Απόδειξη (με επαγωγή):
Άν είχαμε 1 μόνο σημείο προφανώς ισχύει.
Υποθέτω ότι για κάθε n < k ο greedy αλγόριμος δίνει την ίδια λύση με το βέλτιστο. Θα δείξω ότι θα ισχύει και για n = k.
Έστω ότι μπορούμε να σχεδιάσουμε ένα διάστημα [si, fj] για τα διαδοχικά σημεία με βάση το παραπάνω (σε κ σημεία) (αν δεν υπήρχε πάλι ο αλγόριθμος βγάζει την βέλτιστη λύση με 2 σημεία). Αν τοποθετήσω ένα σημείο μου στο min(f1,f2,…,fk) = fj, προκειμένου να βρώ το μέγιστο τότε το υποπρόβλημα που δημιουργείται έχει λύση σίγουρα μικρότερη από οποιοδήποτε άλλο αφού αποτελεί υποσύνολο του. Για το υποπρόβλημα ισχύει η επαγωγική υπόθεση από όπου προκύπτει και η βελτιστότητα του αλγορίθμου.

Άσκηση 2

1. Στο πρόβλημα αυτό ένας άπληστος αλγόριθμος είναι αρχικά να ταξινομήσει τους λόγους wi/pi σε φθίνουσα σειρά. Στη συνέχεια θα διαλέγει σύμφωνα με τη σειρά ταξινόμησης τους πελάτες. Δηλαδή εάν είχαμε (w1,p1),(w2,p2),(w3,p3) με w2/p2 >= w3/p3 >= w1/p1 τότε πρακτικά ο συνολικός βεβαρυμένος χρόνος εξυπηρέτησης θα ήταν:

w2p2 + w3(p2+p3) + w1(p1+p2+p3) (δηλαδή θα παίρναμε απλά πρώτα το 2 μετά το 3 και τέλος το 1, σύμφωνα πάντα με την ταξινόμηση).

θα δείξω ότι η συγκεκριμένη προσέγγιση είναι βέλτιστη. Ειδικότερα θα χρησιμοποιήσω το ακόλουθο αν έχω 2 αντικείμενα τότε υπάρχουν 2 πιθανές διατάξεις 1,2 η 2,1 όπου ο βεβαρημένος χρόνος εξυπηρέτησης είναι
w1 * p1 + w2(p1 + p2), w2 * p2 + w1 * (p1 + p2) αντίστοιχα.

Είναι φανερό ότι για να κρατήσω αυτό με τον μικρότερο βεβαρυμένο χρόνο αρκεί να συγκρίνω το w2 * p1 με το w1 * p2 άρα για να έχω το πρώτο αντικείμενο πρώτο πρέπει w2 * p1 < w1 * p2 => p1 / w1 < p2 / w2.

Έστω τώρα η τελική διάταξη που ελαχιστοποιεί το συγκεκριμένο κριτήριο και 2 διαδοχικά αντικείμενα για τα οποία ισχύει pk / wk > pk+1 / wk+1.
Από την παραπάνω παρατήρηση για το πρόβλημα με μόνο 2 αντικείμενα ότι αν αυτά τα 2 αλλάξουν θέση η μετρική μας θα μειωθεί καθώς με αφαίρεση κατά μέλη των 2 σχέσεων που θα προκύψουν θα δούμε ότι η διαφορά εξαρτάται μόνο από τον όρο
Pk * wk+1, Pk+1 * wk (αυτό που θέλαμε δηλαδή)
Έτσι, για κάθε 2 αντικείμενα στην βέλτιστη λύση πρέπει να ισχύει ότι Pi / wi < Pi+1 / wi + 1 που είναι ισοδύναμο με το να έχω την ταξινόμηση που αναφέρθηκε παραπάνω.

Προκύπτει λοιπόν, πως με διαδοχικές μετατοπίσης εν τέλη θα προκύψει μπροστά ο μεγαλύτερος λόγος P/W και θα ακολουθούν σορταρισμένοι οι υπολοιποι.

Θ(N log N)

2. Έστω ότι έχω τα αντικείμενα με την σειρά που είχα προηγουμένως. Οι οντότητες που με ενδιαφέρουν είναι ο συνολικός βεβαρημένος χρόνος και για αυτό με νοιάζει το άθροισμα P

dp[i][P] = min(dp[i+1][P + pi] + wi * (P+pi), T(i+1,P) + wi * (prefix[i] – P)) όπου ο prefix πίνακας που περιλαμβάνει όλα τα αθροίσματα των P μέχρι την i θέση και dp[i][P] ο βέλτιστος χρόνος που μπορώ να επιτύχω τοποθετώντας τον i-οστό πελάτη σε εργαζόμενο με ήδη υπάρχουντα συνολικό χρόνο P.
Ουσιαστικά εδώ έχουμε 2 επιλογές, ή θα τον επιλέξη ο πρώτος ή ο δεύτερος. Θα έχει πολυπλοκότητα $O(N \cdot \sum p_i)$
Για παραπάνω εργαζόμενους αρκεί να προσθέσω μια διάσταση κάθε φορά. Θα έχουμε δηλαδή P1, P2, P3, … Pm τα οποία θα αντιστοιχούν στα αντίστοιχα αθροιστικά βάρη του κάθε εργαζόμενου κάθε φορά.
Πχ dp[i][P1][P2] = min(

```
        dp[i+1][P1+pi][P2] + wi * (P1+pi),
        dp[i+1][P1][P2 + pi] + wi * (P2+pi),
        dp[i+1][P1][P2] + wi * (prefix[i] – P1 – P2)
)
```

Εδώ η πολυπλοκότητα αυξάνεται σε $O(N \cdot (\sum p_i)^m)$

Άσκηση 3

α) Έστω dp[n] η τιμή που προκύπτει αν σπάσουμε βέλιστα τα στέγαστρα (το τελευταίο διάστημα περιλαμβάνει το n-οστό στοιχείο)

Έστω ότι για κάποιο n γνωρίζω όλα τα βέλτιστα που προκύπτουν πριν από αυτό και κρατώντας το μικρότερο που προκύπτει αν για κάθε m < n, σπάσω το το ολικό στο βέλτιστο όπως ήταν μέχρι το m και προσθέσω ένα υπόστεγο από το m+1 μέχρι το n.

Έτσι, dp[n] = min(dp[m] + (x_n – x_{m+1})^2 + C) for each m < n

β) Αρχικά αφαιρώ κάθε ευθεία που το a της είναι ίδιο με άλλης ευθείας και έχει μεγαλύτερο b. Έτσι προκύπτει α1 > α2 > … > αk
Τοποθετώ την πρώτη ευθεία στο καρτεσιανό, τότε για κάθε σημείο $(-\infty,+\infty)$ τότε η απάντηση είναι πάνω σε αυτή την ευθεία.
Η 2η ευθεία επειδή έχει μικρότερο α αναγκαστικά θα έχει ένα σημείο τομής με την άλλη ευθεία (x1,y1). Έτσι για x < x1 επιλέγουμε την πρώτη ευθεία ενώ για x > x1 επιλέγουμε την 2η.
Τοποθετώ λοιπόν όλες τις ευθείες που το σημείο τομής με την τελευταία ευθεία που έχει τοποθετηθεί είναι μεγαλύτερο από το προηγούμενο xi, φτιάχνοντας έτσι ένα κυρτό σχήμα.

Αν μας έρθει μια ευθεία που το σημείο τομής με την τελευταία ευθεία είναι μικρότερο πάει να πει οτι μπορούμε να αφαιρέσουμε καποια/κάποιες ευθείες. Όσον αφορά η τελευταία ευθεία που βάλαμε πλέον δεν χρειάζεται καθώς η ευθεία που εξετάζουμε είναι σίγουρα καλύτερη για αυτά τα χ. Συνεχίζοντας, και κόβωντας όσες ευθείες δεν μας χρειάζονται πλέον, φτάνουμε στο νέο κυρτό σχήμα που είναι η απάντηση μας μέχρι αυτή την ευθεία. Είναι φανερό πως επειδή κάθε ευθεία θα κοπεί μόνο 1 φορά και έτσι θα διαπεραστεί μόνο 1 φορά, τελικά η πολυπλοκότητα του αλγορίθμου θα είναι Θ(N).

Με $a=-2x_{k+1}$ και $b=x_{k+1}^2+dp[k]$ μπορούμε λοιπόν να λύσουμε το (α) πρόβλημα σε amortized O(N).

Δηλαδή κάνοντας τον παραπάνω μετασχηματισμό μετατρέπουμε το πρόβλημα μας του (α) σε ισοδύναμο πρόβλημα τύπου (β). Το παραπάνω μπορεί να παρομοιασθεί με το ότι επιλέγουμε την νέα ευθεία για τα μεγάλα χ όταν χρειάζεται να φτιάξουμε ένα στέγαστρο για εκείνες τις περιπτώσεις.

Άσκηση 4

i) Έστω ότι dp[i][k] σημαίνει ο βέλτιστος τρόπος που μπορώ να βάλω τους i-πρώτους φοιτητές σε k λεωφορεία.

Έστω ότι βάλω τους m φοιτητές σε k-1 λεωφορεία και βάλω τους υπόλοιπους i-m φοιτητές στο τελευταίο λεωφορείο(από τον m+1 – i) τότε η σχέση γίνεται dp[m][k-1] + X(m, i), όπου

$X(i,j)=\sum_{n=i}^{j}\sum_{m=i}^{j}A_{nm}$.

Συνεπώς, είναι $dp[i][k]=min(dp[m][k-1]+X(m,i))\,for\,each\,m$

Έτσι όπως είναι η πολυπλοκότητα είναι $\Theta(N^4K)$

Ωστόσο, με έξυπνη χρήση ενός prefix πίνακα μπορούμε να το μειώσουμε σε $\Theta(N^2K)$

Εν συντομία, μπορούμε να κατασκευάσουμε έναν πίνακα prefix[i][j] ο οποίος θα μας δίνει ένα άθροισμα άν βάλουμε διαδοχικά τους i, … , j. Έτσι, το X(i,j) = prefix[i][j].

Για να υλοποιηθεί ο prefix χρειάζεται ένας βοηθητικός πίνακας που help[i][j] σημαίνει ξεκινώντας από το το i στοιχείο και χρησιμοποιώντας μόνο Aik το άθροισμα των αντίστοιχων αριθμών μέχρι το j και για j < i τότε είναι 0.

Έτσι ο prefix σχηματίζεται σε Θ(N^2)

ii)
Μπορούμε να κάνουμε binary search ως προς την ανώτατη τιμή ευαισθησίας που μπορεί να μπει σε κάποιο λεωφορείο – έστω M. Αν μπορέσουμε να βρούμε λύση που να λύνει το πρόβλημα για M/2 τότε ψάχνουμε λύση στο [0, M/2], αλλιώς αν δεν βρεθεί κάποια τότε ψάχνουμε στο [M/2,M] διάστημα. Και κρατάμε το threshold της τελευταίας απάντησης που βρήκαμε. Τώρα πως αποφασίζουμε για το αν μια κατάσταση υλοποιείται. Ουσιαστικά θέλουμε να προσθέσουμε όσο των δυνατών περισσότερα αντικείμενα στο 1ο λεωφορείο και να κρατήσουμε την ευαισθησία μικρή. Ουσιαστικά δηλαδή τρέχουμε ένα knapsack για την πρώτη διάσταση και πέρνουμε τους αντίστοιχους φοιτητές που μπορούμε να τοποθετήσουμε. Τώρα συμπληρώνουμε στο 2ο λεωφόρειο για τους εναπομείναντα φοιτητές κοκ. Αν για το τελευταίο λεωφορείο δεν είναι η πλήρης εφαρμογή όλων των φοιτητών τότε η λύση απέτυχε και πρέπει να ψάξουμε σε μεγαλύτερο διάστημα.

Άσκηση 5

α) Έστω T1, T2 συνεκτικά δέντρα και ε ακμή που ανήκει στο T1 και όχι στο T2 αν αφαιρέσω την ε δημιουργώ 2 τμήματα από κορυφές όπου κάθε τμήμα είναι συνεκτικό με την έννοια ότι σε κάθε τμήμα υπάρχει μονοπάτι από κάθε κορυφή στις υπόλοιπες. Διαλέγω τώρα ακμή που ανήκει στο T2 και όχι στο T1 η ακμή αυτή δεν μπορεί να έχει άκρα στο ίδιο τμήμα αφού θα δημιουργούταν κύκλος όποτε ενώνει τα τμήματα προφανώς τώρα υπάρχει μονοπάτι από κάθε ακμή προς κάθε άλλη.

Η εύρεση της ακμής αυτής μπορεί να γίνει με dfs ξεκινώντας από κορυφή του T1 έχοντας πρώτα τοποθετήσει σε μία δομή όλες τις κορυφές του πρώτου spanning tree. Στο dfs θα ελέγχω αν οι κορυφές που βρίσκω υπάρχουν στην δομή αν όχι τότε η τελευταία ακμή που χρησιμοποίησα είναι η ζητούμενη εκτός και αν είναι η ακμή ε που αφαίρεσα οπότε και συνεχίζω την αναζήτηση. Γίνεται σε O(V)

β) Έστω T το συνεκτικό δέντρο που προκύπτει από τα T1,T2 με την διαδικασία του πρώτου ερωτήματος, τότε τα T, T1 απέχουν κατά μία ακμή άρα συνδέονται με ακμή στο H. Έστω τώρα T1,T2 τυχαία spanning trees με απόσταση κ θα δείξω ότι υπάρχει μονοπάτι από το T1 στο T2 στο H και μάλιστα η απόσταση τους είναι ακριβώς κ.

Εκτελώ επαγωγή στην απόσταση κ

για κ = 1 έχει δειχθεί στο πρώτο ερώτημα

Έστω ότι ισχύει για κάθε κ < ν θα δείξω ότι ισχύει και για κ = ν
Πράγματι αν T1,T2 απέχουν κατά κ τότε από α ερώτημα υπάρχει T3 που απέχει 1 από το T1 εξ ορισμού η ακμές που διαφέρει το T2 με το T3 είναι ν-1 άρα από επαγωγή υπάρχει μονοπάτι μήκος ν-1 που να τα συνδέει όμως το T1,T3 συνδέονται με ακμή άρα προκύπτει το ζητούμενο.

Για να βρεθεί η απόσταση αλγοριθμικά μπορώ να εφαρμόσω διαδοχικά τον αλγόριθμο του πρώτου ερωτήματος δηλαδή σε κάθε βήμα θα αφαιρώ μια ακμή που υπάρχει στο T1 και όχι στο T2 και θα τοποθετώ μια που ανήκει στο T2 και όχι στο T1. Με διαδοχικές επαναλήψεις θα φτάσω στο T2 αφού η απόσταση μειώνεται κατά 1 ύστερα από κ βήματα έχω την απάντηση. Κάθε dfs χρειάζεται o(v) στην χειρότερη περίπτωση (καθώς έχω spanning tree) άρα συνολική πολυπλοκότητα o(κ * v)

γ) Αρχικά μπορούμε να υπολογίσουμε ένα πιθανό mst σε O(ElogE) χρησιμοποιώντας Kruskal (ή Prim) το οποίο θα έχει ένα άθροισμα Σ. Τώρα, για κάθε ακμή η οποία εν τέλη ανήκει στο mst τότε η απάντηση είναι Σ. Αν μια ακμή δεν ανήκει στο mst τότε, την τοποθετούμε. Μόλις την τοποθετήσουμε θα δημιουργηθεί ένας κύκλος στον γράφο. Με DFS και κρατώντας τα parent nodes μπορούμε με το backtracking της DFS να βρούμε ακριβώς το μέγιστο κόμβο που χρησιμοποιήθηκε, τον οποίο μπορούμε και να αφαιρέσουμε χωρίς να χαθεί η συνεκτικότητα του γράφου. Έτσι, σε O(V+E) = O(V) να βρούμε κάθε φορά το αντίστοιχο άθροισμα. Τελικά αφού το τρέχουμε για E ακμές έχουμε Θ(EV).