3η Σειρά Γραπτών Ασκήσεων Αλγορίθμων

Παπαρρηγόπουλος Θοδωρής
el18040

Άσκηση 1

α)

Αν στον κόμβο 1 είναι k = 2. Τότε υπάρχουν 2 πιθανα MST με τον greedy αλγοριθμο.
i) 1, 2, 100 → 103
ii) 1, 3, 30 → 34
Όμως το MST είναι 1,2,3.

β) Έστω Α το σύνολο που περιλαμβάνει το συνολικό βάρος των συνεκτικών δέντρων όπου η κορυφή s έχει βαθμό κ.
Αν αυξάναμε την τιμή κάθε ακμής που προσπίπτει στο s κατά K το ελάχιστο αυτού του συνόλου δεν θα άλλαζε. Οπότε, λύνω το πρόβλημα όταν έχω αυξήσει τις ακμές αυτές κατα K. Αν το K είναι πολύ μεγάλη τιμή, τότε καμία ακμή δεν θα χρησιμοποιούνταν, ενώ όταν είναι πολύ μικρή θα χρησιμοποιούνταν όλες. Επομένως θα πρέπει να βρεθεί η κατάλληλη τιμή με binary search.
Για να μπει μια ακμή {s,v} στο MST, θα πρέπει να ισχύει w(s,v) + K <= της μεγαλύτερης ακμής στο υπάρχει path.
Έτσι, αλλαγή είανι πιθανόν να συμβεί μόνο όταν προσθέσω K = e – w όπου e η ακμή του γράφου και w ακμή που προσπίπτει στο s.
Έτσι, βρίσκουμε όλες τις ακμές e-w (Γίνεται σε O(E * deg(s))). Sortaro αυτό το σύνολο και κάνω binary search.
Αν βρίσκομαι σε υποψήφια τιμή K, προσθέτω K σε όλες τις ακμές που προσπίπτουν στο s και βρίσκω το mst αν έχει η κορυφή αυτή βαθμό κ.
Αν έχει ίδιο, τότε τέλος.
Αν έχει μεγαλύτερο δοκιμάζω με μεγαλύτερο K.
Ειδαλλιώς δοκιμάζω με μικρότερο K.
O(log E * (E log E + E ^ deg(s)))

Άσκηση 2

i) Ουσιαστικά εφαρμόζω τον αλγόριθμο του Bellman Ford. Για κάθε κόμβο θα ελέγχω μόνο όσα παιδιά μπορούν να φτάσουν με θετική ζωή. Αρχικοποιώ πίνακα reachability d με n θέσεις και d[u] = inf for each u in V.
Σε κάθε δωμάτιο u→v ελέγχω αν d[v] > d[u] + p[u] > 0, ελέγχω δηλαδή αν ο παίχτης μπορεί να φτάσει στο u αν μπορει να περάσει στο v.
Αν μπορεί να περάσει τότε κρατάμε την ενέργεια στην θέση v: d[v] = d[u] + p[u] > 0.
Επαναλαμβάνουμε μέχρι να ελέξουμε για κάθε βήμα.
O(VE).

ii) Σε αυτή την περίπτωσει ελέγχω αν κάποια κορυφή τέτοιων κύκλων είναι προσβάσιμη και αν η κορυφή tήταν προσβάσιμη από κάποια κορυφή αυτού του κύκλου. Αν ναι, τότε η διαδρομή είανι ασφαλής. Ελέγχω αν d[t] > 0, αν όχι χαλαρώνουμε τις ακμές άλλη μια φορά και αν αλλάξει κάποια τιμή τότε έχουμε θετικό κύκλο ο οποίος είναι προσβάσιμος από την s. O(VE).

iii) Δυαδική αναζήτηση στις τιμές και τρέχουμε τον αλγόριθμο που ελέγχει αν ο λαβύρινθος είναι ασφαλής.

Άσκηση 3

Αρχικά θα θέλαμε να αγοράζουμε όσο περισσότερη βενζίνη μοπρούμε από την πόλη με την φθηνότερη τιμή κάθε φορά. Έτσι, μπορώ είτε να αγοράσω βενζίνη αρκετή για να φτάσω σε σημείο με μικρότερη τιμή βενζίνης (αν με φτάνει), αλλιώς θα πρέπει να γεμίζω σε αυτό το σημείο και πάω στο επόμενο και συνεχίζω τον αλγόριθμο έχοντας ήδη στο ντεπόζιτο αυτό που είχα γεμίσει από την προηγούμενη.
Έτσι είναι τετραγωνικός ο χρόνος, ωστόσο μπορώ να το λύσω με τον αλγόριθμο της πρώτης σειράς άσκηση 3. Θα έχω δηλαδή μια στοίβα για να μπορώ να υπολογίσω σε O(N) το επόμενο καλό σημείο.

Ξεκινώντας από την αρχή εισάγω τα στοιχεία σε ένα stack αν το στοιχείο που κοιτάω είναι μικρότερο από την κορυφή του stack τότε για το στοιχείο που βρίσκεται στην κορυφή του stack έχω βρει την απάντηση

4 6 3 7 2 8 1
i = 0 stack = [] a[i] = 4
i = 1 stack = [4] a[i] = 6 stack.front < a[i] άρα εισάγω το 6 στο stack
i = 2 stack = [6 4] a[i] = 3 stack.front > a[i] (θα γίνει 2 φορές) άρα βρίσκω index[0] = 2 index[1] = 2
i = 3 stack = [3] stack.front < a[i] άρα εισάγω το 7 στο stack
i = 4 stack = [3 4] stack.front < a[i] (πάλι 2 φορές) άρα index[2] = 4 index[3] = 4
i = 5 stack = [2] stack.front < a[i] άρα εισάγω το 8
i = 6 stack = [2 8] γίνονται πάλι 2 pop και έχω ότι index[4] = 6 index[5] = 6

β) Το λύνω με bellman ford.
Έστω dp[u][b] η βέλτιστη τιμή που μπορώ να φτάσω στον κόμβο u με βενζίνη b, τότε επειδή δεν έχω αρνητικούς κύκλους

dp[u][b] = min (dp[i][x] + C(u) * (b-x + b(u,v))) for each x < b where e(u,v) ακμή

Τελικά η απάντηση είναι το dp[s][0]

Άσκηση 4

Αρχικά θα πρέπει να υπολογίσουμε ότι όταν ο στρατός θέλει να επιτεθεί σε κάποια βάση εξτρεμιστών ή σε κάποια δύναμη εξτρεμιστών (με τις τοποθεσίες ei και bj) θα επιλεχθεί η αντίστοιχα κοντινότερη απόσταση ak.

Δημιουργώ επίσης μια λίστα για κάθε εξτρεμιστή με όλες τις βάσεις που απέχουν απόσταση <= δ από αυτόν. Έτσι, μπορώ να σχηματίσω έναν κατευθυνόμενο γράφο που ξεκινάει από έναν κόμβο s που θεωρώ τον αρχικό μου κόμβο και πηγαίνει με διαφορετική ακμή σε όλους τους εξτρεμιστές. Από τον κάθε εξτρεμιστή θα πηγαίνει στις βάσεις με απόσταση <= δ. Και τέλος από όλες τις βάσεις θα υπάρχει μια ακμή προς τον τερματικό κόμβο t. Όσον αφορά τα κόστοι:
i) s → εξτρεμιστές: θα είναι το κόστος να σκοτώσω τον κάθε εξτρεμιστή
ii) εξτρεμιστή → βάση: άπειρο
iii) βάση → t: ο υπολογισμός του να καταστρέψεις την συγκεκριμένη βάση εξτρεμιστών.

Έχοντας τα παραπάνω, το αρχικό πρόβλημα είναι ισοδύναμο με το να βρούμε το max-flow min cut στον παραπάνο γράφο.

Απόδειξη:
Αν το πρόβλημα έχει λύση με τιμή <= k, τότε κρατώντας αυες τις ακμές και κορυφές που δημιούργησαν με το cut, μπορώ να κάνω επίθεση κόστους <= k, κάνοντας επίθεση στους κόμβους βάσης της που κράτησα και στους κόμβους εξτρεμιστών. Αν έχω τις θέσεις των εξτρεμιστών και των βάσεων που πρέπει να βαρέσω για να έχω κόστος <= k τότε στον γράφο θα υπάρχει min-cut <= k.

Έτσι, το πρόβλημα λύνεται αν εφαρμόσουμε τον αλγόριθμο Ford-fulkerson στον γράφο που σχηματίστηκε.

Πολυπλοκότητα: Θα χρειαστεί Θ(mlogm + nlogn + klogk) για να ταξιμονήμουσε και ύστερα για κάθε δύναμη και βάση αν βρούμε τον κοντινότερο στρατό και m^2 U για τον αλγόριθμο Ford-fulkerson.

Άσκηση 5

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων

Θα κάνω αναγωγή στο 2-Partition.
Έστω τώρα ότι μπορώ να λύσω το δωθέν πρόβλημα για κάθε input.
Άρχικά θεωρώ ένα παραλληλόγραμμο B (x1 + … +xn) με 2 διαστάσεις.
Υποθέτω input χ_ι > 2 ώστε να μην μπορούμε να επιλεχθούν κάθετα, παρα μόνο οριζόντια. Υποθέτουμε ότι η απάντηση στο πρόβλημα για αυτό το input είναι ναι. Τότε απευθείας βλέπουμε ότι το παραλληλόγραμμο B θα έχει 2 γραμμές με άθροισμα (x1+...+xn)/2 και άρα έχουμε partition.

Αν η απάντηση στο 2-Partition είναι ναι τότε θα είναι και στο πρόβλημα επειδή μπορώ να επιλέξω για κάθε γραμμή το αντίστοιχο partitioned set.

2-Partition <= Π
Συνεπώς το πρόβλημα ανήκει στο NP-Complete.

Μέγιστη Τομή με Βάρη στις Κορυφές

Θα κάνω αναγωγή στο 2-Partition.
Έστω τώρα ότι μπορώ να λύσω το δωθέν πρόβλημα για κάθε input.
Αρχικά θεωρώ ένα πλήρες γράφημα G που το βάρος κάθε κορυφής προκύπτει από τα βάρη των αντικειμένων και B = (x1+..xk)^2/4 όπου x_i είναι οι κορυφές που ανήκουν στο S. Στο V\S θεωρούμε ότι οι κορυφές είναι y1,..,ym. Έστω ότι για δωθέν input το πρόβλημα απαντάει με “Ναι”.

Το βάρος της τομής ισούντε με x1(y1 + .. + ym) + x2(y1 +.. +ym) + ... + xk(y1 + ... + ym) = (x1 + ... + xk) * (y1 + … +ym)

Το άθροισμα όλων των κορυφών του γράφο θα είναι:
(Sum for each u in G {w(u) }) = x1 + .. + xk + y1 + … + ym, έστω = C.
Αν χ1+...+χκ = X
Το βάρος της τομής είναι = X * (C – X)
Έστω F(X) = X * (C – X) <= B
F’(X) = C – 2X
F’’(X) = -2 < 0, συνεπώς επειδή η παράγωγος είναι φθίνουσα θα έχουμε μέγιστο στο σημείο X = C/2.
Άρα θεωρώ X = C/2 και F(C/2) = C^2/4.
Άρα, προκύπτει ότι Y = C/2 και συνεπώς έκανα Partition.

Αν η απάντηση στο 2-Partition είναι ναι τότε θα είναι και στο πρόβλημα επειδή μπορώ να επιλέξω για τομές τα partitioned σύνολα.

2-Partition <= Π
Συνεπώς το πρόβλημα ανήκει στο NP-Complete.

Αραιό Γράφημα (Sparse Subgraph)

θα κάνω αναγωγή στο independent set.

Έστω τυχαίος γράφος G που είναι είσοδος του independent set το input του sparse είναι το εξής μεταφέρω αυτούσιο τον γράφο και σε κάθε κορυφή προσθέτω ένα sparse subgraph με χ κόμβους αν 2 κόμβοι στο G συνδέονται τότε συνδέω και στο νέο γράφημα όλους τους κόμβους που προστέθηκαν σε αυτούς. Προφανώς αν έχω independent set μεγέθους κ έχω και ένα sparse μεγέθους κ * χ αν τώρα έχω ένα sparse μεγέθους κ * χ μπορώ να επιλέξω το χ μεγάλο έτσι ώστε αν στο γράφημα που θα προκύψει έχω 2 κορυφές που να συνδέονται λόγω του ότι θα συνδέονται και όλοι οι νέοι κόμβοι που προστέθηκαν σε αυτούς θα έχω είδη χ^2 > κχ κόμβους άρα όλοι οι αρχικοί κόμβοι του γραφήματος που θα προστεθούν πρέπει να είναι ανεξάρτητοι

Συντομότερο Μονοπάτι με Περιορισμούς

Θα κάνω αναγωγή στο 2-Partition.
Έστω τώρα ότι μπορώ να λύσω το δωθέν πρόβλημα για κάθε input.

Σχεδιάζω κατευθυνόμενο γράφημα με V κορυφές και E = 2*V ακμές.
Από κάθε κορυφή θα φεύγουν 2 ακμές προς την επόμενη κορυφή.

Η πάνω ακμή θα έχει κόστος (xi, 0).
Η κάτω ακμκή θα έχει κόστος (0, xi).

Έστω επίσης ότι s η πρώτη κορυφή και t η τελευταία κορυφή του γράφου (πιο δεξιά) και έστω W = (χ1 + … χν)/2 με C = W.

Έστω τώρα επίσης διαδρομή που έχει συνολικό βάρος <= W και συνολικό μήκος <= C.

Τέλος έστω ότι η απάντηση σε αυτή την ερώτηση είναι Ναι.
Αν οι πάνω ακμές που ανήκουν στο path έχουν πλήθος κ και έχουν βάρος xa1 + xa2 + … + xak και μήκος 0.
Αν οι κάτω ακμές που ανήκουν στο path έχουν πλήθος n-k θα έχουν βάρος 0 και μήκος xb1 + xb2 + … + xb(n-k)

Το άθροισμα των 2 αυτών θα είναι ίσο με χ1+...+χn.
Έτσι δεν μπορούν να είναι και οι 2 μικρότεροι από το μισό επειδή τοτε το άθροισμα θα γινόταν είτε μεγαλύτερο το W είτε το C.
Έτσι θα πρέπει να είναι ίση με το μισό και συνεπώς θα έχουμε partition.

Αν η απάντηση στο 2-Partition είναι ναι τότε θα είναι και στο πρόβλημα επειδή μπορώ να επιλέξω για κορυφές το partitioned sequence.

2-Partition <= Π
Συνεπώς το πρόβλημα ανήκει στο NP-Complete.