Αλγόριθμοι και Πολυπλοκότητα
3η Εργασία

Χατζηθεοδώρου Ιάσων
03117089

Άσκηση 1

Μπορούμε να ξεκινήσουμε είτε περιττές είτε ζυγές στιγμές. Χρησιμοποιώντας τον αλγόριθμο BFS και αποθηκεύοντας για κάθε κόμβο στο $queue$ το αν μπορούμε να μεταβούμε σε αυτόν περιττή ή ζυγή στιγμή με βάση την ακμή του, μπορούμε να βρούμε την ελάχιστη καθυστέρηση για να φτάσει στον προορισμό. Άρα αρκούν δύο BFS, μία ξεκινώντας περιττή στιγμή και μία ξεκινώντας ζυγή στιγμή, ώστε να βρεθούν οι δύο πιθανές ελάχιστες καθυστερήσεις dt_1, dt_2.

Προκύπτουν οι εξής περιπτώσεις:

- Αν $T-dt_1$ είναι περιττός τότε αυτή είναι η μέγιστη περιττή στιγμή που μπορούμε να ξεκινήσουμε, αλλιώς η στιγμή $T - dt_1 - 1$.
- Αν $T - dt_2$ είναι ζυγός τότε αυτή είναι η μέγιστη ζυγή στιγμή που μπορούμε να να ξεκινήσουμε, αλλιώς η στιγμή $T - dt_2 - 1$.

Άρα μπορούμε έτσι να επιλέξουμε τη μέγιστη στιγμή να ξεκινήσουμε, η οποία θα προκύψει ως μέγιστο των δύο παραπάνω αποτελεσμάτων.

Η πολυπλοκότητα του αλγορίθμου είναι $O(n + m)$ αφού γίνονται δύο BFS.

Η ορθότητα του αλγορίθμου προκύπτει από το γεγονός ότι η BFS βρίσκει ελάχιστες αποστάσεις σε γράφο χωρίς βάρη.

Άσκηση 2

Χρησιμοποιώντας τον αλγόριθμο BFS μπορούμε να βρούμε το δέντρο ελάχιστων αποστάσεων για κάθε ένα από τα σωματίδια. Το σημείο G που ψάχνουμε είναι αυτό που έχει την ελάχιστη μέγιστη απόσταση από τα σωματίδια. Είναι δηλαδή το σημείο στο οποίο θα φτάσουν το συντομότερο δυνατό. Άρα

- Αρκούν k BFS, μία για κάθε σωματίδιο, με τις οποίες θα βρούμε k δέντρα ελάχιστων αποστάσεων
- Σε κάθε BFS μπορούμε να κάνουμε $update$ τη μέγιστη απόσταση του κάθε κόμβου από τα σωματίδια
- Για να βρούμε το G αρκεί ένα πέρασμα σε όλους του κόμβους ώστε να βρεθεί αυτός με ελάχιστη μέγιστη απόσταση από όλα τα σωματίδια
- Αφού βρεθεί το σημείο, τα σωματίδια θα πρέπει να κινηθούν πάνω στο δέντρο ελάχιστων αποστάσεων που τους αντιστοιχεί για να φτάσουν σε αυτό

Η πολυπλοκότητα θα είναι $O(k(n + m))$.

Άσκηση 3

Αν υπάρχουν κύκλοι στον γράφο, τότε οι περίπατοι που ξεκινούν από τις κορυφές τους θα έχουν αναγκαστικά ίδιο ελάχιστο κόστος, εφόσον υπάρχει περίπατος από όλες τις κορυφές του κύκλου προς τις υπόλοιπες. Άρα αρκεί να συμπτήξουμε τους κόμβους τους και να λύσουμε το πρόβλημα στο DAG που θα προκύψει.

Σε DAG μπορούμε για κάθε κορυφή να υπολογίσουμε τα δυνατά κόστη περιπάτων που ξεκινάνε από κάθε κορυφή. Για να υλοποιηθούν τα παραπάνω χρειάζονται τα εξής

- Μία DFS η οποία όταν κληθεί για μια κορυφή και βρει $back - edge$ ξεκινάει να συμπτήσσει τις κορυφές που βρίσκονται στο $stack$ μέχρι να φτάσει στην ίδια κορυφή. Μετά την σύμπτηξη φτιάχνεται ένας κόμβος ο οποίος έχει κόστος ίσο με το gcd των κορυφών του κύκλου και όλες τις ακμές τους.
- Για κάθε κορυφή φτιάχνουμε ένα πίνακα με μέγεθος όσο το μέγιστο κόστος κορυφής στο DAG. Ξεκινώντας από την τελευταία κορυφή και πηγαίνοντας προς τα πίσω, συμπληρώνουμε τον πίνακα κάθε κορυφής u σημειώνοντας τα $gcd(c(u), gcd_v)$ για κάθε $v \in adj(u)$, όπου gcd_v είναι όλα gcd που μπορεί να φτάσει η κορυφή v.

Η ορθότητα του αλγορίθμου για την περίπτωση ακυκλικού γράφου προκύπτει με επαγωγή ως προς το πλήθος των κορυφών. Για κάθε κορυφή γνωρίζουμε ότι δεν ανήκει σε κύκλο, οπότε οι περίπατοι που ξεκινούν από αυτήν περνούν μόνο από τις υπόλοιπες. Περνώντας από τις κορυφές μετά το τέλος του αλγορίθμου μπορούμε να βρούμε το ελάχιστο κόστος.

Ως προς την ορθότητα του συνολικού αλγορίθμου, γνωρίζουμε οτι ο περίπατος ελάχιστου κόστους οποιασδήποτε κορυφής που ανήκει σε κύκλο έχει ίσο κόστος με των υπολοίπων και είναι το πολύ ίσο με το gcd όλων των κορυφών του κύκλου. Επιπλέον σχύει ότι

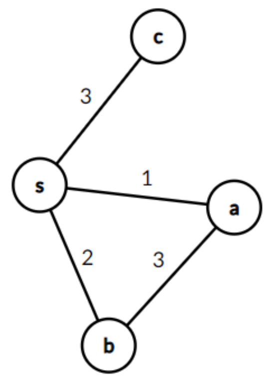
$$gcd(c(v_1), c(v_1), \ldots, c(v_k)) = gcd(gcd(c(v_1), \ldots, c(v_{k-1})), c(v_k))$$

Ως προς την πολυπλοκότητα η DFS είναι γραμμική, ενώ η σύμπτηξη δύο κορυφών γίνεται σε σταθερό χρόνο και στη χειρότερη περίπτωση θα συμπτηχθούν όλες οι κορυφές. Έπειτα, περνάμε από κάθε κορυφή και ακμή μία φορά και για κάθε ακμή κάνουμε ένα πέρασμα στον πίνακα. Άρα $O((n + m) \cdot max_c)$.

Άσκηση 4

(α)

Έστω ότι θέλουμε να βρούμε το $T^*(s,2)$ και χρησιμοποιούμε το *greedy* κριτήριο επιλογής των 2 ελαφρύτερων ακμών του s στον ακόλουθο γράφο.



Στο παραπάνω γράφο θα επιλέγαμε τις ακμές $(s,a),(s,b)$ για τον κόμβο s. Με τις συγκεκριμένες ακμές δεν θα έχουμε καν δέντρο, ενώ θα μπορούσαμε να επιλέξουμε τις $(s,a),(s,c),(a,b)$ οπότε το κριτήριο είναι λανθασμένο.

(β)

Για να υπολογίσουμε το $T^*(s,k)$ μπορούμε να προσθέτουμε ή να αφαιρούμε ακμές με άκρο το s και ανταλλάσσοντάς τες κατάλληλα με άλλες να φτάσουμε σε δέντρο με $\deg(s)=k$.

Για την ορθότητα της παραπάνω διαδικασίας αρκεί να αποδειχθεί ότι το $T^*(s,k)$ προκύπτει από το $T^*(s,k+1)$ με μία ανταλλαγή ακμών και αντίστροφα.

- Έστω οτι το βάρος ενός $T^*(s,i)$ είναι W_i.
- Υπάρχει σίγουρα ακμή $e=(s,u)$ έτσι ώστε $e \in T^*(s,k+1) \setminus T^*(s,k)$.
- Αφαιρώντας την από το $T^*(s,k+1)$ προκύπτουν δύο συνεκτικές συνιστώσες που ορίζουν μια τομή.
- Αν προσθέσω την e στο $T^*(s,k)$ τότε θα δημιουργηθεί κύκλος. Αν για κάθε πιθανή e ο κύκλος του $T^*(s,k)$ αποτελείται μόνο από ακμές με άκρο s, τότε θα είναι κύκλος της μορφής $(s,u),(u,s)$, άρα δεν μπορεί να υπάρξει δέντρο με $deg(s)=k+1$ κάτι που είναι άτοπο. Άρα υπάρχει ακμή η οποία δεν έχει άκρο s και ανήκει στον κύκλο άρα καλύπτει την τομή. Προφανώς μία από αυτές, έστω e' δεν ανήκει στο $T^*(s,k+1)$ αλλιώς θα είχε κύκλο.
- Άρα μπορούμε να αφαιρέσουμε την e από το $T^*(s,k+1)$ και να του προσθέσουμε την e' και να καταλήξουμε σε συνδετικό δέντρο, με $\deg(s)=k$. Για το βάρος αυτού του δέντρου θα ισχύει

$$W_{k+1} - w(e) + w(e') \geq W_k$$

- Ομοίως μπορούμε να προσθέσουμε την e στο $T^*(s,k)$ και να του αφαιρέσουμε την e' και να καταλήξουμε σε συνδετικό δέντρο, με $\deg(s)=k+1$. Για το βάρος αυτού του δέντρου θα ισχύει

$$W_k + w(e) - w(e') \geq W_{k+1}$$

- Συνδυάζοντας τις παραπάνω σχέσεις θα ισχύει

$$W_k + w(e) - w(e') = W_{k+1}$$

Άρα για να πάμε από το $T^*(s,k+1)$ στο $T^*(s,k)$ ή ανάποδα αρκεί μία ανταλλαγή, την οποία πρέπει να βρούμε.

Έστω ότι ένα MST του γράφου που έχει $\deg(s)=d$

- Αν $k=d$ τότε βρέθηκε το δέντρο, αφού το MST είναι $T^*(s,d)$
- Αν $k>d$ τότε πρέπει να προστεθούν $k-d$ ακμές με άκρο s. Για κάθε προσθήκη θεωρούνται υποψήφιες όλες οι ακμές του s που δεν έχουν προστεθεί ακόμα. Για κάθε μία από αυτές, δοκιμάζουμε να την προσθέσουμε και να βρούμε τον κύκλο που προέκυψε, με DFS. Από τις ακμές του κύκλου αρκεί να αφαιρέσουμε την βαρύτερη που δεν έχει άκρο το s. Άρα στο προηγούμενο κόστος θα προστεθεί ένας όρος $w(e)-w(e')$. Δοκιμάζοντας όλες τις πιθανές ακμές βρίσκουμε το ελάχιστο επιπλέον βάρος.
- Αν $k<d$ τότε πρέπει να αφαιρεθούν $d-k$ ακμές με άκρο s. Για κάθε αφαίρεση θεωρούνται υποψήφιες όλες οι ακμές του s στο δέντρο. Για κάθε μία από αυτές δοκιμάζουμε να την αφαιρέσουμε και να βρούμε τις δύο συνεκτικές συνιστώσες που δημιουργούνται, με DFS. Με ένα πέρασμα στις ακμές μπορούμε να βρούμε την ελαφρύτερη ακμή που συνδέει τις δύο συνιστώσες. Άρα στο προηγούμενο βάρος προστίθεται ένας όρος της μορφής $w(e')-w(e)$. Ελαχιστοποιώντας τον συγκεκριμένο όρος βρίσκουμε το ελάχιστο επιπλέον βάρος.

Άρα η συνολική πολυπλοκότητα θα είναι ίση με $O(m+n\log n+kn(n+m))=O(kn(n+m))$

Άσκηση 5

(α)

Για κάθε ακμή που προστίθεται από τον αλγόριθμο *Boruvka* συμπτήσσονται δύο κορυφές, άρα σε κάθε επανάληψη του αλγορίθμου απομένουν το πολύ οι μισές κορυφές.

Το ότι ο συγκεκριμένος αλγόριθμος καταλήγει σε συνδετικό δέντρο αποδεικνύεται με επαγωγή στο πλήθος κορυφών

- Για $n = 1$ προφανώς ισχύει
- Έστω ότι για $n \leq n_0$ ο αλγόριθμος καταλήγει σε συνδετικό δέντρο
- Για $n = 2n_0$ μετά από μια επανάληψη θα απομείνουν το πολύ n_0 κορυφές, και άρα συνδετικές συνεκτικές συνιστώσες, οπότε από επαγωγική υπόθεση φτιάχνει συνδετικό δέντρο

Το συνδετικό δέντρο που φτιάχνει ο αλγόριθμος είναι ελάχιστο γιατί κάθε ακμή που επιλέγει είναι ακμή επαύξησης για την τομή $(S, V \setminus S)$ αν S το σύνολο των κορυφών που έχουν συμπτηχθεί σε μία, αφού η ακμή θα έχει το ελάχιστο βάρος από όσες διασχίζουν την τομή.

(β)

- Μέχρι να απομείνει ένας κόμβος
  - Για κάθε κορυφή βρες την ελαφρύτερη ακμή της και πρόσθεσέ την
  - Σύμπτηξε τα άκρα της ακμής που προστέθηκε
  - Συνέχισε με το νέο γράφο που προέκυψε από τις συμπτήξεις

Σε κάθε επανάληψη απομένουν το πολύ οι μισοί κόμβοι άρα γίνονται $\log n$ επαναλήψεις. Σε κάθε επανάληψη γίνεται γραμμικό πέρασμα στις ακμές. Άρα συνολικά η πολυπλοκότητα είναι $O(mlogn)$.

(γ)

Μπορούμε να κάνουμε τα εξής

- Κάνουμε $\log \log n$ επαναλήψεις του αλγορίθμου *Boruvka*, οι οποίες απαιτούν

$$O(m \log \log n)$$

- Μετά την ολοκλήρωσή τους, οι κορυφές που απομένουν είναι

$$\frac{n}{2^{\log \log n}} = \frac{n}{\log n}$$

- Αν στις κορυφές που απομένουν εφαρμοστεί ο αλγόριθμος *Prim* με *Fibonacci Heap* θα χρειαστεί

$$O(m + \frac{n}{\log n} \cdot \log \frac{n}{\log n}) = O(m + n) = O(m)$$

Άρα συνολικά η πολυπλοκότητα θα είναι $O(m \log \log n)$

(δ)

Αν ο γράφος ανήκει σε βολική κλάση, τότε μετά από κάθε επανάληψη του αλγορίθμου το πλήθος των ακμών, m, θα παραμένει $O(n)$. Αν $T(n)$ είναι ο αριθμός των επαναλήψεων τότε ισχύει η εξής αναδρομική σχέση

$$T(n) = T(\frac{n}{2}) + O(n)$$

Ισχύει $n^{\log_b a} = 0$. Άρα βρισκόμαστε στην τρίτη περίπτωση του $Master\ Theorem$ και $T(n) = O(n)$.

Άσκηση 6

(α)

- Υπάρχει τουλάχιστον μία ακμή $e \in T_1 \setminus T_2$ αλλιώς δεν θα ήταν διαφορετικά συνδετικά δέντρα
- Αφαιρώντας από το T_1 την e δημιουργούνται δύο συνεκτικές συνιστώσες που ορίζουν μία τομή (S, T)
- Αν προσθέσουμε την e στο T_2 δημιουργείται κύκλος επειδή η τομή (S, T) διασχίζεται και από άλλες ακμές. Έστω μία από αυτές, e'
- Η e' δεν ανήκει στο T_1 αλλιώς θα υπήρχει κύκλος. Αν την προσθέσουμε στο $T_1 \setminus \{e\}$ θα ενώνει τις δύο συνεκτικές συνιστώσες άρα το $T_1 \setminus \{e\} \cup \{e'\}$ θα είναι συνδετικό δέντρο

Ο αλγόριθμος είναι ο εξής

- Αφαίρεση της e από το T_1
- Εύρεση των δύο συνεκτικών συνιστωσών του T_1 που ορίζουν μια τομή με DFS
- Προσθήκη της e στο T_2
- Εύρεση κύκλου στο T_2 με DFS
- Με ένα πέρασμα στις ακμές του κύκλου μπορεί να βρεθεί αυτή που ενώνει την παραπάνω τομή. Αυτή η ακμή θα είναι η e'

Άρα η πολυπλοκότητα θα είναι $O(n + m)$.

(β)

Αρκεί να αποδείξω με επαγωγή ότι αν τα T_1, T_2 είναι συνδετικά με $|T_1 \setminus T_2| = k$, υπάρχει διαδρομή από T_1 ως T_2 μήκους k.

- Για $k = 1$ διαφέρουν κατά μία ακμή, άρα στο H συνδέονται με ακμή
- Έστω για $k \leq k_0$ υπάρχει διαδρομή μήκους k_0
- Για $k = k_0 + 1$
  - Επιλέγω μία από τις $k_0 + 1$ ακμές, έστω $e \in T_1 \setminus T_2$.
  - Όπως αποδείχτηκε υπάρχει $e' \in T_2 \setminus T_1$ ώστε το $T_1' = T_1 \setminus \{e\} \cup \{e'\}$ να είναι συνδετικό δέντρο.
  - Ισχύει $|T_1' \setminus T_1| = 1$ άρα έχουν ακμή στο H. Επίσης, $|T_1' \setminus T_2| = k_0$ άρα από επαγωγική υπόθεση υπάρχει διαδρομή μήκους k_0 ανάμεσά τους. Άρα συνολικά υπάρχει διαδρομή μήκους $k_0 + 1$ ανάμεσα στα T_1, T_2.

Εφαρμόζοντας τον προηγούμενο αλγόριθμο k φορές μπορούμε σε κάθε επανάληψη να μειώνουμε κατά μία τη διαφορά σε ακμές των δύο δέντρων. Άρα συνολικά η πολυπλοκότητα θα είναι $O(k(n + m))$.

(γ)

Για κάθε ακμή e

- Πρόσθεσε την e στο συνδετικό δέντρο
- Χρησιμοποίησε τον αλγόριθμο $Prim$ ξεκινώντας από τη συνεκτική συνιστώσα που δημιουργήθηκε προσθέτοντας την ακμή

Με αυτόν τον τρόπο θα βρεθούν τα ελάχιστα συνδετικά δέντρα που αντιστοιχούν στην κάθε ακμή. Η πολυπλοκότητα είναι $O(m^2 + mn\log n)$.