3η Σειρά Γραπτών Ασκήσεων

Μάνος Αραπίδης

ΑΜ: 03116071

1η Άσκηση

Αρχικά θέλουμε να μετατρέψουμε τις πληροφορίες σχετικά με τα κουτιά και τα κλειδιά σε μια μορφή που να είναι εύκολο να τις επεξεργαστούμε. Μια τέτοια μορφή είναι ένας κατευθυνόμενος γράφος $G(V,E)$, όπου κάθε κορυφή αντιπροσωπεύει ένα κουτί , οπότε έχουμε συνολικά $|V|=n$ κορυφές και κάθε ακμή αντιστοιχεί σε ένα κλειδί, άρα έχουμε $|E|=m$ ακμές. Ένα κλειδί το οποίο είναι μέσα στο κουτί i και ανοίγει το κουτί j, στον γράφο G αναπαριστάται με την ακμή του κόμβου i προς τον j: (i,j) .

Το πρόβλημα είναι ότι θέλουμε να σπάσουμε τον ελάχιστο αριθμό κουτιών για να αποκτήσουμε όλα τα m κλειδιά, το οποίο μεταφράζεται στο να βρούμε τις ελάχιστες κορυφές που είναι αρκετές για να μπορούμε,ξεκινώντας από αυτές να κινηθούμε σε όλο τον γράφο G.

Για να λύσουμε το πρόβλημα,πρέπει πρώτα να βρούμε τις ισχυρά συνεκτικές συνιστώσες του γράφου. Ο λόγος που χρειαζόμαστε τις ΙΣΣ είναι ότι σύμφωνα και με τον ορισμό τους είναι το σύνολο κόμβων όπου για κάθε κορυφή i και j υπάρχει μονοπάτι (i,j) και (j,i) , άρα μία κορυφή που ανήκει στην ΙΣΣ είναι αρκετή ώστε ξεκινώντας από αυτήν να μπορούμε να επισκεφτούμε όλες τις κορυφές της ΙΣΣ.

Ο αλγόριθμος για να βρούμε τις ΙΣΣ είναι γνωστός και στηρίζεται στην αναζήτηση dfs. Η πολυπλοκότητα του αλγορίθμου είναι $O(|V|+|E|)$,καθώς το dfs γίνεται με λίστα γειτνίασης (το σημείωμά) και μέτα την ολοκλήρωση του έχουμε όλες τις ΙΣΣ του γράφου. Αν έχουμε μία μόνο ΙΣΣ τότε ο γράφος G είναι ισχυρά συνεκτικός, το οποίο σημαίνει ότι ξεκινώντας από οποιαδήποτε κορυφή μπορούμε να καλύψουμε τον γράφο. Άρα, σε αυτήν την περίπτωση χρειαζόμαστε να σπάσουμε ένα μόνο τυχαίο κουτί.

Σε διαφορετική περίπτωση, που έχουμε παραπάνω από μία ΙΣΣ: $C_1,C_2,...,C_k$, $1<k\leqslant n$ συμπτύσσουμε την κάθε μία, ώστε να αναπαριστάται με ένα κόμβο και παράγουμε ένα γράφημα DAG $G'(V',E')$ με $|V'|=k\leqslant n$ και $|E'|\leqslant m$. Προκείμενου να βρούμε τον τρόπο με τον οποίο οι κορυφές (ή αλλιως ΙΣΣ) στο νέο γράφημα G′ γίνονται προσβάσιμες, ώστε να διαλέξουμε μόνο εκείνες που μας είναι απαραίτητες, θα εκτελέσουμε τον παρακάτω γνωστό αλγόριθμο τοπολογικής ταξινόμησης.

Αρχικά φτιάχνουμε ένα vector στο οποίο θα αποθηκεύσουμε την τοπολογική διάταξη των κορυφών.

Τοπολογική ταξινόμηση:

1. καλούμε dfs(G′) για να βρούμε τον χρόνο περάτωσης για κάθε κόμβου $u\in V'$.
2. μετά την περάτωση κάθε κόμβου τον τοποθετούμε στην αρχή του vector.
3. μετά την ολοκλήρωση του dfs επιστρέφουμε το vector.

Ο παραπάνω αλγόριθμος παράγει ένα δάσος $\Delta(V')$ (ή δεντρό αν ο γράφος είχε μια μόνο συνιστώσα) και τοποθετεί τους κόμβους του G΄ στο vector σε φθίνουσα σειρά βάσει τον χρόνο που εκείνες έγιναν εξερευνημένες. Το οποίο σημαίνει ότι για ένα κόμβο που βρίσκεται σε μία τυχαία θέση στο vector , η κίνηση προς κόμβους μπροστά από αυτόν στο vector (προς την αρχή του vector) είναι αδύνατη, το οποίο είναι αυταπόδεικτο καθώς σε διαφορετική περίπτωση θα είχε χρόνο εξερεύνησης μεγαλύτερο, σύμφωνα με τους κανόνες του dfs. Άρα, θέλουμε να βρούμε τις ελάχιστες κορυφές του vector που καλύπτουν την κίνηση προς όλες τις άλλες. Για να το κάνουμε αυτό ακολουθούμε τον παρακάτω αλγόριθμο.

1. Ξεκινάμε από την αρχή του vector.
2. Διαγράφουμε από το vector όσες κορυφές είναι προσβάσιμες από την κορυφή που εξετάζουμε, μπορεί να γίνει με dfs.
3. Συνεχίζουμε στο επόμενο στοιχείο του vector και επαναλαμβάνουμε το βήμα 2.
4. Όταν δεν έχουμε άλλα στοιχεία να εξετάσουμε, επιστρέφουμε το vector.

Εν τελεί, στο vector έχουν μείνει οι κορυφές εκείνες ώστε ξεκινώντας από αυτές να μπορείς να επισκεφτείς όλες κορυφές του γράφου G΄. Όπως είπαμε προηγουμένως, οι κορυφές του G΄ αντιστοιχούν σε ΙΣΣ του γράφου G, οπότε διαλέγοντας μια οποιαδήποτε κορυφή από κάθε ΙΣΣ όπου επέστρεψε ο παραπάνω αλγόριθμος αποτελεί την λύση στο πρόβλημα μας.

Η πολυπλοκότητα του αλγορίθμου είναι $O(|V|+|E|)=O(n+m)$, η οποία υπολογίζεται ως εξής:

- Εύρεση ΙΣΣ : $O(|V|+|E|)=O(n+m)$.
- Τοπολογική ταξινόμηση: $O(|V'|+|E'|)=O(|V|+|E|)=O(n+m)$, $|V'|\leqslant n$ και $|E'|\leqslant m$.
- Επιλογή ΙΣΣ: $O(|V'|)=O(|V|)=O(n)$, $|V'|\leqslant n$.

Η ορθότητα της λύσης συμπεραίνεται από την παραπάνω επεξήγηση του αλγορίθμου.

2 Άσκηση

Έχουμε ένα δέντρο $T(V,E)$ και κάθε κορυφή u που ανήκει στο T έχει βάρος $w(u)$.

Το πρόβλημα μας είναι να βρούμε το μονοπάτι p με το μέγιστο συνολικό βάρος $w(p)$.

Αρχικά πρέπει να ορίσουμε την θέση που μπορεί να έχει μία κορυφή u στο μονοπάτι p και είναι η εξής:

- Η κορυφή μπορεί να είναι στις άκρες του μονοπατιού , δηλαδή είτε στην αρχή είτε στο τέλος.
- Η κορυφή μπορεί ενδιάμεσα του μονοπατιού, οπότε σε αυτήν την κορυφή μπορούμε να πούμε ότι το μονοπάτι κάνει ‘τόξο’.
- Η κορυφή δεν ανήκει στο μονοπάτι.

Οπότε,για κάθε κορυφή u χρειάζεται να αποθηκεύουμε δύο τιμές, το βάρος $w_{άκρης}(u)$ στην περίπτωση που βρίσκεται στην άκρη του μονοπατιού και το βάρος $w_{τόξου}(u)$ αν βρίσκεται ενδιάμεσα. Για να υπολογίσουμε αυτά τα βάρη χρειάζεται να γνωρίζουμε για τα παιδιά της u : u_1, u_2, ..., u_m το βάρος $w_{άκρης}(u_ι)$, $1\leqslant i\leqslant m$.

Για να υπολογίσουμε το βάρος $w_{άκρης}(u)$, χρειάζεται να βρούμε το μέγιστο βάρος $w_{άκρης}(u_i)\geq 0$ των παιδίων της u, αν έχει παιδια και να προσθέσουμε σε αυτό το βάρος w(u). Οπότε, διαλέγοντας το μέγιστο $w_{άκρης}(u_i)$ εξασφαλίζουμε ότι και το βάρος $w_{άκρης}(u)$ θα έχει την μέγιστη τιμή. Επίσης, χρειάζεται να αποθηκεύουμε την επιλογή παιδιού που κάναμε για να μπορούμε βρούμε το μονοπάτι που επιλέξαμε.

$w_{άκρης}(u) = w(u) + \max\{w_{άκρης}(u_i)\}, \quad 1 \leqslant i \leqslant m \quad . \quad O(m)$

Στην περίπτωση που η μέγιστη τιμή $w_{άκρης}(u_i)$ είναι αρνητική,δηλαδή όλα τα μονοπάτια των παιδιών έχουν αρνητικά βάρη, τότε αποφασίζουμε να μην διαλέξουμε κανένα από αυτά καθώς δεν μας προσφέρουν κανένα κέρδος. Άρα, σε αυτήν την περίπτωση το βάρος είναι $w_{άκρης}(u) = w(u)$. Ομοίως, αν η κορυφή είναι φύλλο , δηλαδή δεν έχει παιδιά, τοτε το βάρος άκρης είναι $w_{άκρης}(u) = w(u)$.

Για να υπολογίσουμε το βάρος $w_{τόξου}(u)$, χρειάζεται να βρούμε τις δύο κορυφές παιδιά i και j με το μέγιστο βάρος $w_{άκρης}(u_i)\geq 0$ και $w_{άκρης}(u_j)\geq 0$ και προσθέτουμε τα βάρη με το w(u). Με αυτόν τον τρόπο εξασφαλίζουμε ότι το μονοπάτι 'τόξου' με ενδιάμεση κορυφή u θα έχει την μέγιστη τιμή. Ακόμα , χρειάζεται να αποθηκεύουμε τις επιλογές παιδιών που κάναμε, για να μπορούμε μετά να βρούμε το μονοπάτι 'τόξου' που επιλέξαμε.

$w_{τόξου}(u) = w(u) + \max\{w_{άκρης}(u_i)\} + \max\{w_{άκρης}(u_j)\}, \quad 1 \leqslant i \leqslant m \quad , \quad 1 \leqslant j \leqslant m \quad \text{και} \quad i \neq j \quad . \quad O(m)$

Στην περίπτωση που η κορυφή u έχει μόνο ένα παιδί ή είναι φύλλο δεν έχει νόημα να αναφερόμαστε σε μονοπάτι 'τόξου' , για αυτό θα εξισώσουμε το βάρος $w_{τόξου}(u)$ με το βάρος $w_{άκρης}(u)$ που έχουμε υπολογίσει πριν: $w_{τόξου}(u)=w_{άκρης}(u)$. Επίσης, αν δεν υπάρχουν τουλάχιστον δύο κόμβοι i και j, με μονοπάτια $w_{άκρης}(u_i)\geq 0$ και $w_{άκρης}(u_j)\geq 0$ τότε το βάρος $w_{τόξου}(u)\leq w_{άκρης}(u)$, οπότε πάλι εξισώνουμε τις δύο τιμές :$w_{τόξου}(u)=w_{άκρης}(u)$.

Για να βρούμε τα βάρη $w_{τόξου}(u)$ και $w_{άκρης}(u)$ για κάθε κόμβο κάνουμε dfs. Σύμφωνα με τους κανόνες του dfs μια κορυφή u γίνεται εξερευνημένη όταν όλα τα παιδιά της : $u_1, u_2, ..., u_m$ γίνουν εξερευνημένα, οπότε για κάθε κορυφή u την στιγμή της εξερεύνησης της υπολογίζουμε τα βάρη $w_{τόξου}(u)$ και $w_{άκρης}(u)$ όπως περιγράψαμε παραπάνω. Άρα μετά την ολοκλήρωση του dfs για κάθε κορυφή u έχουμε υπολογιστεί τα βάρη $w_{τόξου}(u)$ και $w_{άκρης}(u)$.
Τέλος, με ένα ακόμα dfs διατρέχουμε το δέντρο για να βρούμε την κορυφή u με το μέγιστο βάρος, το οποίο μπορεί να είναι $w_{τόξου}(u)$ ή $w_{άκρης}(u)$. Οπότε, τυπώνουμε το μονοπάτι(για κάθε κορυφή είχαμε αποθηκεύσει στο προηγούμενο dfs) που αντιστοιχεί στην λύση μας.

Η πολυπλοκότητα του αλγορίθμου είναι $O(|V|)$ και υπολογίζεται ως εξής:

- Αφού τα dsf γίνονται με λίστα γειτνίασης η πολυπλοκότητα του είναι $O(|V|)$.
- Για να υπολογίσουμε για μία κορυφή u τα βάρη $w_{τόξου}(u)$ και $w_{άκρης}(u)$ χρειαζόμαστε $O(m)$ συγκρίσεις, όπου m ο αριθμός των παιδιών της. Αυτό συμβαίνει επειδή οι υπολογισμοί γίνονται με την σειρά του dfs. Άρα συνολικά θα χρειαστούν $O(|V|)$ συγκρίσεις.

<u>3η Άσκηση</u>

Έχουμε ένα συνεκτικό μη κατευθυνόμενο γράφο $G(V,E)$,με n κορυφές και m ακμές. Στο παιχνίδι παίζουν οι παίχτες Κώστας και Ανδρές, στον ρόλο του κλέφτη και αστυνομικού ο καθένας αντίστοιχα και τους συμβολίζουμε ως Κ και Α. Οι παίχτες Κ,Α αρχίζουν από τις θέσεις $t, p \in V$ και το

καταφύγιο βρίσκεται στη θέση $d \in V$. Σε κάθε γύρο παίζει ένας παίκτης και έχει την υποχρέωση να κινηθεί σε μια γειτονική θέση από όπου βρίσκεται. Το παιχνίδι τερματίζει με τους έξης τρόπους :

- Αν ο Κ φτάσει στο καταφύγιο πριν τον "πιάσει" ο Α, οπότε κερδίζει ο Κ.
- Αν ο Α βρεθεί στην ίδια θέση με τον Κ, οπότε κερδίζει ο Α.
- Αν επαναληφθεί κάποια προηγούμενη θέση κάποιου παίχτης, δηλαδή ο παίχτης έκανε κύκλο, οπότε έχουμε ισοπαλία.

Θέλουμε να βρούμε ένα αλγόριθμο ο οποίος, επειδή οι παίχτες κάνουν πάντα τις βέλτιστες κινήσεις, θα καθορίζει το αποτέλεσμα του παιχνιδιού βάση τις αρχικές θέσεις του Κ και Α.

Για να λύσουμε το πρόβλημα θα χρειαστεί να φτιάξουμε ένα νέο γράφημα $G'(V', E')$ μ ε $|V'| = O(|V|^2) = O(n^2) \kappa\alpha\iota |E'| = O(|V| * |E|) = O(n * m)$. Κάθε κόμβος αναπαριστά της εξής κατάσταση $(\theta\acute{\epsilon}\sigma\eta\, K, \theta\acute{\epsilon}\sigma\eta\, A, \epsilon\pi\acute{o}\mu\epsilon\nu\sigma\varsigma\, \pi\alpha\acute{\iota}\chi\tau\eta\varsigma)$. Για κάθε θέση του Κ: $u \in V$, πρέπει να εξετάσουμε τις n διαφορετικές θέσεις: $v \in V$ που μπορεί να βρίσκεται ο Α, ισχύει η ίδια λογική συμμετρικά για τον Α. Οπότε προκύπτουν $O(|V|^2)$ καταστάσεις , άρα ο G θα έχει $O(|V|^2) = O(n^2)$ κόμβους.

Στο γράφημα G΄ μια ακμή μπορεί να είναι της μορφής : {(u, v, K),(u΄,v,A)} ή {(u,v,A),(u,v΄,K)}. Στο αρχικό γράφημα G από την θέση u μπορείς να μετακινηθείς σε edges(u) διαφορετικές θέσεις. Στο γράφημα υπάρχουν n διαφορετικές καταστάσεις για την θέση u του παίκτη Κ: (u,v,K), $v \in V$ και κάθε μία μπορεί να μεταβεί σε edges(u) διαφορετικές καταστάσεις. Άρα για κάθε $u \in V$ υπάρχουν n καταστάσεις με edges(u) ακμές η καθεμία, οπότε συνολικά έχουμε $O(n * edges(u))$ ακμές. Άρα προκύπτει ότι ο G΄ έχει $O(|V| * |E|) = O(n * m)$.

Σύμφωνα με τα παραπάνω ο Κ κερδίζει αν βρεθεί στο καταφύγιο, οπότε σε όλες τις καταστάσεις (d,v,_) κερδίζει ο Κ. Δεν έχει σημασία ποίος παίζει καθώς ο Κ είναι "ασφαλής". Αντίστοιχα ο Α κερδίζει όταν βρεθεί στην ίδια θέση με τον Κ, άρα στις καταστάσεις (v,v,-). Πάλι δεν έχει σημασία ο επόμενος παίχτης ο Κ "πιάστηκε". Τις καταστάσεις από τις οποίες ξεκινώντας κερδίζει ο Κώστας τις συμβολίζουμε με Κ, ενώ εκείνες για τον Αδρέα με Α. Τέλος, εκείνες που οδηγούμαστε σε ισοπαλία τις συμβολίζουμε Ι.

Αν για μία κατάσταση $m \in V'$ γνωρίζουμε τον συμβολισμό που έχουν όλοι οι γείτονες της Κ,Α ή Ι, τότε μπορούμε να συμπεράνουμε τον συμβολισμό της m. Έστω ότι στην m παίζει ο Κ,δηλαση είναι της μορφής (u,v,K), τότε μπορούμε να καταλήξουμε στον συμβολισμό του m σύμφωνα με τα παρακάτω.

i. Αν υπάρχει έστω και μία γειτονική κατάσταση με συμβολισμό Κ, τότε και ο m αποκτά Κ καθώς αν ακολουθήσουμε την διαδρομή αυτού του γείτονα φτάνουμε σε κατάσταση που κερδίζει ο Κ. Στην συγκεκριμένη περίπτωση δεν χρειάζεται να έχουν καταλήξει όλοι οι γείτονες σε συμβολισμό, αρκεί που υπάρχει έστω και ένας με Κ.
ii. Αν όλες οι γειτονικές καταστάσεις έχουν συμβολισμό Α, τότε ο m αποκτά Α καθώς όλες του οι επιλογές οδηγούν σε νίκη του Α.
iii. Αν οι γειτονικές καταστάσεις έχουν συμβολισμούς Α και Ι, τότε ο m θα επιλέξει τον γείτονα που θα τον οδηγήσει σε ισοπαλία, αποκτώντας τον συμβολισμό Ι.

Θεωρούμε φυσικά ότι οι γείτονες έχουν κάνει την βέλτιστη επιλογή και καταλήγουν σε Α,Κ ή Ι. Εφαρμόζουμε την ίδια λογική συμμετρικά για τις καταστάσεις που παίζει ο Α.

Αρχικά , βάζουμε σε όλες τις καταστάσεις του γραφήματος G΄ τον συμβολισμό Ι και για κάθε κατάσταση $m \in V'$ διατηρούμε μία μεταβλητή num_m που μας ενημερώνει για το πόσες γειτονικές καταστάσεις έχουν συμβολισμό Ι, την οποία αρχικοποιούμε με τον αριθμό τον γειτόνων. Βρίσκουμε τις

καταστάσεις που γνωρίζουμε που γνωρίζουμε ότι ο Κ και ο Α, όπως περιγράψαμε παραπάνω και τους αποδίδουμε συμβολισμό Κ και Α αντίστοιχα. Τις καταστάσεις αυτές τις ονομάζουμε αρχικές και μπορούμε να τις βρούμε με ένα dfs στο γράφο G΄.

Προκειμένου να αποδώσουμε συμβολισμούς και στις υπόλοιπες καταστάσεις θα εφαρμόσουμε ένα τροποποιημένο αλγόριθμο για παράλληλα bfs από τις αρχικές καταστάσεις, σύμφωνα με τα παρακάτω:

1) Φτιάχνουμε μία άδεια ουρά και τοποθετούμε σε αυτήν τις αρχικές καταστάσεις.
2) Επαναλαμβάνουμε μέχρι η ουρά να αδειάσει.
   a) Αφαιρούμε το πρώτο στοιχείο m της ουράς.
   b) Ελέγχουμε αν υπάρχουν γειτονικές ακμές με συμβολισμό I, εκτελούμε τα παρακάτω
      I. Προσπαθούμε αν γίνεται να αποδώσουμε συμβολισμό σύμφωνα με όσα περιγράψαμε στο i) . Για όσες γίνεται, τις συμβολίζουμε με Κ ή Α ανάλογα .
      II. Για εκείνες p ,που δεν μπορούμε να κάνουμε το παραπάνω μειώνουμε την μεταβλητή num_p κατα 1 , καθώς στην κατάσταση m από την οποία εξετάζουμε την p έχει αποδοθεί συμβολισμός. Αν η μεταβλητή num_p μηδενιστεί, αποδίδουμε συμβολισμό όπως περιγράψαμε στα ii) και iii).
3) Για κάθε κατάσταση που αποδίδουμε Κ ή Α , την εισάγουμε στο τέλος της ουράς.

Μετά την ολοκλήρωση του παραπάνω έχουμε εξασφαλίσει ότι με το άδειασμα της ουράς έχουμε αποδώσει συμβολισμό σε όλες τις καταστάσεις. Άρα, ελέγχοντας τις αρχικές συνθήκες μπορούμε να συμπεράνουμε για το αποτέλεσμα του παιχνιδιού.

Ο αλγόριθμος στηρίζεται σε υλοποίηση bfs στο γράφο G΄ , άρα προκύπτει πολυπλοκότητα $O(n^2+n*m)$.

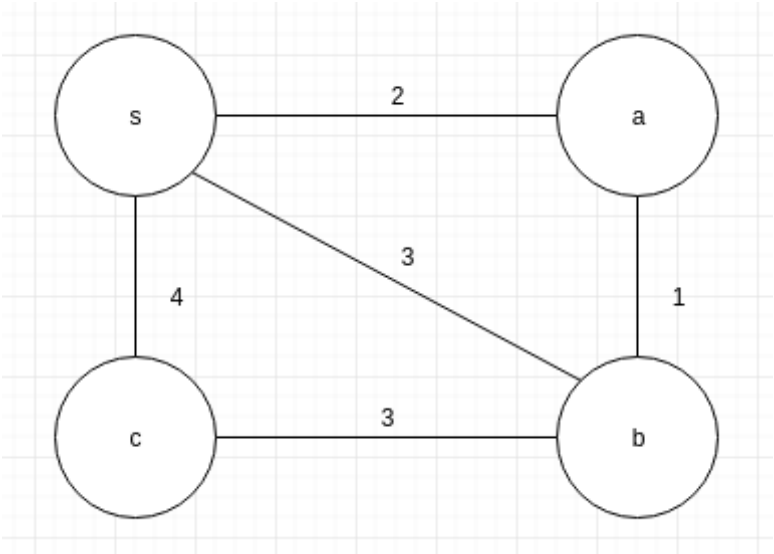
<u>4η Άσκηση</u>

Έχουμε ένα συνεκτικό μη κατευθυνόμενο γράφημα $G(V,E,w)$, με n κορυφές , m ακμές και θετικά βάρη w σε αυτές. Θέλουμε να βρούμε ένα ελάχιστο συνεκτικό δέντρο $T*(s,k)$, το οποίο θα έχει την ιδιότητα ότι η κορυφή s θα έχει βαθμό ίσο με k.

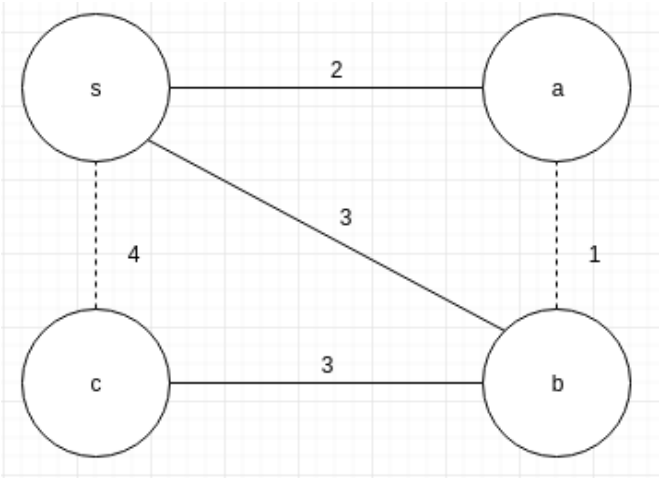
a)

Η άπληστη στρατηγική η οποία είναι να συμπεριλαμβάνουμε στο συνδετικό δέντρο τις k κορυφές του s με το ελάχιστο βάρος είναι λάθος. Αυτό φαίνεται από το επόμενο αντιπαράδειγμα.

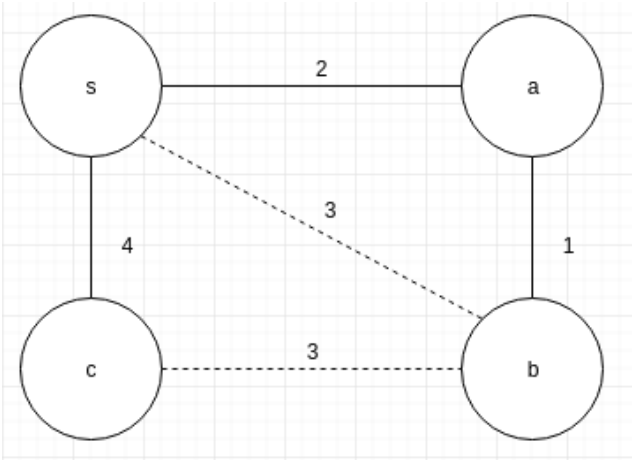
Έστω ότι έχουμε τον εξής γράφημα:



Έστω ότι θέλουμε το ΕΣΔ $T'(s,2)$. Ακολουθώντας την παραπάνω άπληστη στρατηγική προκύπτει το εξής δέντρο:



Ενώ το σωστό $T'(s,2)$ είναι:



Οπότε η άπληστη στρατηγική είναι λανθασμένη.

b)

Για να βρούμε το ΕΣΔ $T'(s,k)$ θα ακολουθήσουμε τα βήματα του παρακάτω αλγόριθμου.

1. Αρχικά , αφαιρούμε προσωρινά την κορυφή s από το γράφο G. Αυτό έχει ως αποτέλεσμα, επειδή ο γράφος είναι συνεκτικός ,να προκύψουν m συνεκτικές συνιστώσες : $C_1,C_2,...,C_m$.
2. Για κάθε μία από τις $C_1,C_2,...,C_m$ βρίσκουμε το ΕΣΔ :T_i, $1 \leqslant i \leqslant m$, οπότε παράγεται ένα δάσος $\Delta(V)$. Για να βρούμε τα ΕΣΔ χρησιμοποιούμε τον αλγόριθμο του Kruskal , $O(m\log(n))$
3. Προσθέτουμε ξανά την κορυφή s στο δάσος Δ και για κάθε ΕΣΔ T_i βρίσκουμε την ελάχιστη ακμή που το συνδέει στην κορυφή s. Για να το κάνουμε αυτό πρέπει να ελέγξουμε όλες τις ακμές της s και να κρατάμε μία ακμή προς κάθε ΕΣΔ, αυτή με το ελάχιστο βάρος. Η s μπορεί να έχει το πολύ n ακμές , άρα $O(n)$.
4. Μέτα την ολοκλήρωση θα έχει παραχθεί ένα ΕΣΔ T_s του γράφου G, με το μέγιστο βαθμό της κορυφής s.Αν ο βαθμός του s είναι k, τότε έχουμε βρει την λύση μας και είμαστε σίγουροι για την εγκυρότητα της ,καθώς το T_s παράχθηκε ενώνοντας την κορυφή s με ελάχιστες ακμές προς ΕΣΔ.
5. Όσο ο βαθμός της s είναι <k επαναλαμβάνουμε τα παρακάτω βήματα.
6. Για κάθε ακμή της κορυφής s: {s,u}, $u \in V-s$ που δεν ανήκει στο δέντρο T_s βρίσκουμε στο μονοπάτι $s \rightarrow u$ την ακμή μέγιστου βάρους και την ονομάζουμε u_{max}. Για κάθε ζευγάρι ακμών {s,u} και u_{max} υπολογίζουμε την διαφορά κόστους που θα είχαμε αν αντικαθιστούσαμε την ακμή u_{max} με {s,u} : $dif(s,u)=w(u_{max})-w(s,u)$. Το παραπάνω μπορεί να γίνει ξεκινώντας ένα dfs από την κορυφή s και να κρατάμε για κάθε μονοπάτι την ακμή με το μέγιστο βάρος, $O(n)$. Είναι προφανές ότι δεν συμπεριλαμβάνουμε στην εύρεση της μέγιστης ακμής των μονοπατιών ,τις ακμές της κορυφής s καθώς αυτό μπορεί να μείωνε τον βαθμό της κορυφής s.
7. Από τις διαφορές κόστους dif(s,u) που υπολογίσαμε, κρατάμε την μεγαλύτερη και αντικαθιστούμε στο δέντρο T_s την ακμή u_{max} με την {s,u}.

Η πολυπλοκότητα του αλγορίθμου είναι $O(n^2)$ και προκύπτει ως εξής: επειδή η κορυφή s μπορεί να έχει μέχρι και n ακμές και ο απαιτούμενος βαθμός να είναι $k=n$, τα βήματα 5 έως 7 μπορεί να χρειαστεί να επαναληφθούν n φορές, άρα θα κάνουμε n dfs => $O(n^2)$.
$O(n^2+O(m\log(n))+n)=O(n^2)$.

Η ορθότητα του αλγορίθμου συμπεραίνατε από το γεγονός ότι όσο ο βαθμός της κορυφής s είναι <k , αυξάνουμε τον βαθμό του κατά ένα κάθε φόρα, ανταλλάσοντας αχρησιμοποίητες ακμές της s με ακμές του δέντρου T_s, και διατηρούν το δέντρο ελάχιστο για κάθε βαθμό από 1 εώς k. Οπότε σε κάθε επανάληψη του αλγορίθμου το δέντρο T_s παραμένει ελάχιστο.

<u>5η Άσκηση</u>

a)

Θεωρούμε ένα συνεκτικό μη κατευθυνόμενο γράφο $G(V,E,w)$, με n κορυφές , m ακμές και θετικό βάρος $w(e), e \in E$.

Ο αλγόριθμος του Boruvka είναι ο παρακάτω:

1) Αρχικά αρχικοποιούμε ένα δάσος Δ με n δέντρα ενός στοιχείου , κάθε ένα από τα οποία είναι έχει μία κορυφή από τον γράφο G. Δηλαδή φτιάχνουμε n διαφορετικές συνιστώσες.
2) Επαναλαμβάνουμε τα παρακάτω βήματα όσο το δάσος Δ έχει παραπάνω από μια συνιστώσα.
   a) Για κάθε συνιστώσα C του G κάνουμε τα εξής:
      i. Φτιάχνουμε με ένα άδειο σύνολο ακμών: S.

ii. Για κάθε κορυφή u που ανήκει στο C βρίσκουμε την φτηνότερη ακμή προς μια κορυφή v που δεν ανήκει στο C. Την ακμή που βρήκαμε την προσθέτουμε στο S.
iii. Βρίσκουμε την ακμή με το μικρότερο βάρος στο σύνολο S και την προσθέτουμε στο C.

b) Συμπτύσσουμε τις συνιστώσες που συνδέονται για να φτιάξουμε μεγαλύτερες.

3) Το Δ είναι πλέον το ΕΣΔ του γράφου G.

Αρχικά θέλουμε να δείξουμε ότι το γράφημα που επιστρέφει είναι συνεκτικό δέντρο. Για να το κάνουμε αυτό πρέπει να δείξουμε ότι είναι συνεκτικό και χωρίς κύκλους. Αρχίζουμε τον αλγόριθμο μας με n συνεκτικές συνιστώσες και σε κάθε επανάληψη ψάχνουμε να βρούμε ακμές επαύξησης, όπου εξορισμού είναι η ακμή που συνδέει δυο συνιστώσες. Στο τέλος της επανάληψης συμπτύσσουμε εκείνες που καταφέραμε να ενώσουμε, παράγοντας νέες μεγαλύτερες συνεκτικές συνιστώσες και είμαστε σίγουροι ότι είναι συνεκτικές καθώς για να τις φτιάξουμε συνδέσαμε δυο άλλες συνεκτικές συνιστώσες. Όποτε σε κάθε επανάληψη συνδέουμε συνεκτικές συνιστώσες , μέχρι να υπάρχει μόνο μία. Άρα το γράφημα που προκύπτει είναι συνεκτικό.

Για να εξασφαλίσουμε ότι το γράφημα δεν θα έχει κύκλους εφαρμόζουμε ολική διάταξη χωρίς ισοπαλίες. Στην περίπτωση που δύο άκυκλες συνεκτικές συνιστώσες έχουν παραπάνω από μια ακμή επαύξησης ίδιου χαμηλότερου κόστους, για να αποφύγουμε το ενδεχόμενο να επιλέξουν η κάθε μία διαφορετική ακμή και να σχηματιστεί κύκλος ,εφαρμόζουμε ολική διάταξη στις ακμές και καθιστούμε την ακμή επαύξησης χαμηλότερου κόστους μοναδική. Οπότε δυο συνιστώσες διαλέγουν την ίδια ακμή επαύξησης και αποφεύγουμε τους κύκλους. Έτσι λοιπόν, αφού ξεκινάμε με n άκυκλες συνεκτικές συνιστώσες και τις συνδέουμε με ακμές επαύξησης που δεν προκαλούν προκαλούν κύκλο , παράγουμε νέες άκυκλες συνεκτικές συνιστώσες. Άρα το γράφημα στο τέλος του αλγορίθμου είναι άκυκλο.

Οπότε, αποδείξαμε ότι παράγεται συνδετικό δέντρο.

Όπως περιγράψαμε και παραπάνω διαλέγουμε πάντα την ακμή επαύξησης με το μικρότερο βάρος. Όποτε, εφαρμόζοντας την κίνησης αυτή αναδρομικά ότι το δέντρο μας θα έχει το ελάχιστο βάρος.

Άρα, αποδείξαμε ότι ο αλγόριθμος του Boruvka παράγει πάντα ΕΣΔ.

b)

Ο αλγόριθμος του Boruvka έχει πολυπλοκότητα $O(m\,logn)$, κάθε επανάληψη χρειάζεται $O(m)$ χρόνο και συνολικά κάνουμε $O(logn)$ επαναλήψεις.

Κάθε επανάληψη χωρίζεται σε δύο φάσεις:

- 1η φάση: Βρίσκουμε την ελάχιστη ακμή επαύξησης για κάθε συνεκτική συνιστώσα C_i , $1 \le i \le n$. Για να το κάνουμε αυτό πρέπει για κάθε κορυφή $u \in C_i$,να ελέγξουμε αν υπάρχει ακμή {u,v} : $v \notin C_i$, να την προσθέσουμε σε ένα σύνολο ακμών S_i και έπειτα να βρούμε την ελάχιστη ακμή για κάθε S_i. Προκείμενου να συμπληρώσουμε τα σύνολα S_i, θα χρειαστεί να εξετάσουμε όλες τις ακμές του γράφου ,οπότε θέλουμε χρόνο $O(m)$ και για να βρούμε την ελάχιστη ακμή σε κάθε S_i , στην χειρότερη περίπτωση πάλι χρειαζόμαστε $O(m)$ χρόνο. Οπότε η πρώτη φάση έχει $O(m)$ χρόνο εκτέλεσης,
- 2η φάση: Βρίσκουμε τις νέες συνεκτικές συνιστώσες στο δάσος Δ που προέκυψαν με την προσθήκη των ακμών επαύξησης. Για αυτό τον σκοπό χρησιμοποιηούμε γνωστό αλγόριθμο που στηρίζεται στο dfs και επειδή η υλοποίηση του αφορά δάσος με n κορυφές, η πολυπλοκότητα του είναι $O(n)$.

Επειδή $m \geqslant n$, η πολυπλοκότητα προκύπτει $O(m)$.

Σε κάθε επανάληψη συνδέουμε συνεκτικές συνιστώσες, οπότε ο αριθμός αυτών μειώνεται στο μισό κάθε φορά. Αρχικά έχουμε n συνεκτικές συνιστώσες , άρα θα γίνουν το πολύ $O(logn)$ επαναλήψεις.

c)

Σε κάθε επανάληψη του Boruvka συνδέουμε συνεκτικές συνιστώσες, κάνοντας το γράφημα πίο αραιό. Την ιδιότητα αυτή θα εκμεταλλευτούμε χρησιμοποιώντας τον αλγόριθμο Prim σε συνδυασμό με σωρό Fibonacci, για να βρούμε το ΕΣΔ σε χρόνο $O(mlog(logn))$.
Αρχικά θα τρέξουμε τον Boruvka k φορές, αυτό απαιτεί χρόνο $O(m*k)$. Μετά από k επαναλήψεις ο αριθμός των συνεκτικών συνιστωσών θα είναι $n/(2^k)$, τις συμπτύσσουμε ώστε καθεμία να αντιπροσωπεύεται και δημιουργούμε ένα νέο γράφο $G'(V',E), |V'|=n/(2^k)$. Ο αλγόριθμος Prim αξιοποιώντας σωρό Fibonacci για βρει ένα ΕΣΔ, χρειάζεται $O(E+VlogV)$ χρόνο. Όποτε για το νέο γράφημα έχουμε $O(m+n/(2^k)*\log(n/(2^k)))$.
Άρα η πολυπλοκότητα προκύπτει $O(m+n/(2^k)*\log(n/(2^k))+m*k)=O(n/(2^k)*\log(n/(2^k))+m*k)$.
Αν επιλέξουμε $k=\log(\log(n))$, τότε έχουμε πολυπλοκότητα $O(mlog(logn))$.

<u>6η Άσκηση</u>

Έχουμε ένα συνεκτικό μη κατευθυνόμενο γράφο $G(V,E,w)$, με n κορυφές , m ακμές και θετικό βάρος $l(e), e \in E$. Για κάθε μονοπάτι p: $u \to v$ θεωρούμε ως κόστος c(p) το μέγιστο βάρος ακμής e που ανήκει στο p, $c(p): max_{eEp}\{l(e)\}$.

a)

Θέλουμε να δείξουμε ότι για ένα ΕΣΔ T* του G, για κάθε ζεύγος κορυφών $u, v \in V$ το μοναδικό μονοπάτι $u \to v$ στο T* είναι μονοπάτι ελάχιστου κόστους.

Έστω ένα μονοπάτι p΄: $u \to v$ το οποίο έχει μικρότερο κόστος από το μονοπάτι p: $u \to v$ του ΕΣΔ T*. Έστω μια ακμή e με το μέγιστο βάρος l(e) στο μονοπάτι p. Δεδομένου ότι $c(p') < c(p)$ όλες οι ακμές του μονοπατιού p΄ έχουν βάρος < l(e). Αν αφαιρέσουμε την ακμή e από το δέντρο T*, τότε θα έχουμε δύο συνιστώσες T_1 και T_2 , όπου κάθε μία θα περιέχει την κορυφή u και v αντίστοιχα. Αν συνδέσουμε τις δύο συνιστώσες με την ακμή e΄ του μονοπατιού p΄ που διασχίζει την τομή $\{T_1,T_2\}$, τότε προκύπτει ένα νέο ΕΣΔ T με μικρότερο βάρος από το T*, καθώς $l(e') < l(e)$. Άτοπο.

b)

Για κάθε ζευγάρι κορυφών $u, v \in V$,θεωρούμε μονοπάτι ελάχιστου κόστους: p^*_{uv}. Θέλουμε να βρούμε το συνολικό κόστος για όλα τα μονοπάτια ελάχιστου κόστους για κάθε ζεύγος κορυφών στο G, δηλαδή το άθροισμα $\sum_{u,v \in V, u \neq v} c(p'_{uv})$. Για να λύσουμε το πρόβλημα θα στηριχτούμε στο ερώτημα α) , δηλαδή ότι σε ένα ΕΣΔ κάθε μονοπάτι είναι ελάχιστου κόστους.

Ο αλγόριθμος για την λύση είναι ο παρακάτω:

1. Αρχικά βρίσκουμε ένα ΕΣΔ T για τον γράφο $G(V,E,w)$. Εφαρμόζοντας τον αλγόριθμο του Kruskal βρίσκουμε το T σε χρόνο $O(mlogn)$.
2. Αρχικοποιούμε μια μεταβλητή max=0 , στην οποία θα αποθηκεύουμε τον άθροισμα των ελαχίστων μονοπατιών.
3. Επαναλαμβάνουμε μέχρι να έχουμε n συνιστώσες .

i. Για κάθε συνεκτική συνιστώσα $C_i(V_i, E_i)$:
$|V_i| = n_i \kappa\alpha\iota |E_i| = m_i, 1 \leqslant n_i \leqslant n \kappa\alpha\iota 1 \leqslant m_i \leqslant m$ που προκύπτει από την διάσπαση του δέντρου T, βρίσκουμε την ακμή μέγιστου βάρος e_i . $O(n_i)$

ii. Αφαιρούμε από την C_i την ακμή e_i και προκύπτουν δυο νέες συνεκτικές συνίστωσες $C_{(i_1)}(V_{(i_1)}, E_{(i_1)})$, $C_{(i_2)}(V_{(i_2)}, E_{(i_2)})$: $|V_{(i_1)}| = n_{(i_1)} \kappa\alpha\iota |V_{(i_2)}| = n_{(i_2)} = n_i - n_{(i_1)}$
$|E_{(i_1)}| = m_{(i_1)} \kappa\alpha\iota |E_{(i_2)}| = m_{(i_2)} = m_i - m_{(i_1)} - 1$. Προσθέτουμε στην max :
$max = max + n_{(i_1)} * n_{(i_2)} * w(e_i)$

4. Τυπώνουμε την τιμή της max , ως την λύση στο πρόβλημα μας.

Όπως γνωρίζουμε το μονοπάτι ελάχιστου κόστους καθορίζεται από την ακμή μέγιστου βάρους, οπότε αν διασπάμε κάθε συνεκτική συνιστώσα C_i (αρχική συνιστώσα θεωρούμε το δέντρο T) με βάση την ακμή e_i , γνωρίζουμε ότι κάθε μονοπάτι p_{uv} : $u \in V_{(i_1)} \kappa\alpha\iota v \in V_{(i_2)}$ μεταξύ των παραγόμενων συνεκτικών συνιστωσών $C_{(i_1)} \kappa\alpha\iota C_{(i_2)}$ έχει κόστος $w(e_i)$. Άρα προκύπτει $p^*_{uv} = p_{uv}$. Υπάρχουν $n_{(i_1)} * n_{(i_2)}$ μονοπάτια από την $C_{(i_1)} \rightarrow C_{(i_2)}$ και αντίστροφα, όλα έχουν κόστος $w(e_i)$ καθώς διέρχονται από την ακμή μέγιστη ακμή e_i. Οπότε προκύπτει συνολικό κόστος διαδρομών $C_{(i_1)} \rightarrow C_{(i_2)}$:
$n_{(i_1)} * n_{(i_2)} * w(e_i)$.
Το παραπάνω το κάνουμε μέχρι να μην υπάρχουν άλλα μονοπάτια να εξετάσουμε, δηλαδή να έχουμε n συνεκτικές συνιστώσες οι οποίες θα είναι όλες μίας κορυφής.

Ο αλγόριθμος έχει πολυπλοκότητα στην χειρότερη περίπτωση $O(n^2)$ και στην μέση περίπτωση $O(m * logn)$.

Η χειρότερη περίπτωση είναι εκείνη, όπου η συνεκτική συνιστώσα $C_{(i_1)}$ που προκύπτει από την διάσπαση της C_i έχει πλήθος κορυφών κατά ένα μικρότερο : $n_{(i_1)} = n_i - 1$. Οπότε η αναδρομή είναι της μορφής : $T(n) = T(n-1) + \Theta(n) \Rightarrow O(n^2)$. Η πολυπλοκότητα του αλγορίθμου είναι $O(m * logn + n^2) = O(n^2)$.

Στην καλύτερη περίπτωση η συνιστώσα $C_{(i_1)}$ διασπάτε στο μισό στις $C_{(i_1)} \kappa\alpha\iota C_{(i_2)}$. Οπότε η αναδρομή είναι της μορφής : $T(n) = 2T(n/2) + \Theta(n)$, όπου σύμφωνα με το Master Theorem προκύπτει πολυπλοκότητα $O(n * logn)$. Άρα η πολυπλοκότητα του αλγορίθμου είναι $O(m * logn + n * logn) = O(m * logn)$.

Έστω μία μέση περίπτωση, την οποία θεωρούμε “καλή”, όπου η διάσπαση της $C_{(i_1)}$ παράγει $C_{(i_1)} \kappa\alpha\iota C_{(i_2)}$ με $\frac{n}{4} \leqslant n_{(i_1)} \leqslant \frac{3*n}{4} \kappa\alpha\iota n_{(i_2)} = n - n_{(i_1)}$, με πιθανότητα ≥50%. Αποδεικνύεται , με τον ίδιο τρόπο όπως τον γνωστό αλγόριθμο Quicksort, ότι η πολυπλοκότητα της μέσης περίπτωσης είναι $O(n * logn)$. Άρα η πολυπλοκότητα του αλγορίθμου είναι $O(m * logn + n * logn) = O(m * logn)$.