ΤΕΧΝΙΚΉ ΑΝΑΦΟΡΆ

Δομές Δεδομένων 2021-2022 – 1η Εργασία

*n: Το σύνολο των κόμβων της λίστας (εκτός των head και tail)

public ArcList() (Constructor):

Υλοποίηση:

-Γίνεται ορισμός και αρχικοποίηση των κόμβων Κεφαλής και Ουράς.

-Γίνεται διπλή σύνδεσή τους , με τη χρήση των μεθόδων της Node (linktoNext και linktoPrev), που συνδέουν την Κεφαλή με τον επόμενό της Κόμβο (Ουρά) και την Ουρά με τον προηγούμενό της Κόμβο (Κεφαλή).

Πολυπλοκότητα: **O(1)** (σταθερού χρόνου εντολές)

public int size():

Υλοποίηση:

-Γίνεται ορισμός μεταβλητή μετρητής και κόμβου δείκτη current.

-Η current δείχνει στον επόμενο της κεφαλής κόμβο.

- Όσο η current δεν έχει φτάσει στο τέλος της λίστας.

    - Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο μετά από κάθε επανάληψη και

    - Γίνεται αύξηση του μετρητή κατά 1 κάθε φορά που περνά από τον κάθε κόμβο της λίστας.

- Γίνεται επιστροφή του μετρητή - αριθμητικού μεγέθους λίστας

Πολυπλοκότητα: **n*O(1)=O(n)**

(Λόγω βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας)

public int insert(Arc arc):

Υλοποίηση:

-(Κάνει εισαγωγή μόνο όταν δίνεται έγκυρη ακμή για εισαγωγή)

-Ορισμός current που δείχνει στην tail.

-Ορισμός tmp.

- Εισαγωγή κόμβου tmp πριν τον current/ουρά.

(Σύνδεση προς τα πίσω της tmp , με τον προηγούμενο της current – Σύνδεση προς τα μπροστά της tmp, με την current – Σύνδεση προς τα μπροστά του προηγούμενου της current [εκείνον στον οποίο η current δείχνει προς τα πίσω], με την tmp - Σύνδεση προς τα πίσω του επόμενου της tmp [εκείνον στον οποίο η tmp δείχνει προς τα μπροστά] με την tmp .)

- Επιστροφή νέου μεγέθους λίστας.

Πολυπλοκότητα: **O(1)** (σταθερού χρόνου εντολές)

public Arc removeFirst():

Υλοποίηση:

-Ορισμός node που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Αφαίρεση του πρώτου κόμβου της λίστας.

(Σύνδεση προς τα μπροστά του προηγούμενου της node [εκείνον στον οποίο η node δείχνει προς τα πίσω] με τον επόμενο της node [εκείνον στον οποίο η node δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της node [εκείνον στον οποίο η node δείχνει προς τα μπροστά] με τον προηγούμενο της node [εκείνον στον οποίο η node δείχνει προς τα πίσω] - Κατάργηση των δεικτών prev και next της node).

-Επιστροφή της ακμής που περιέχεται στον κόμβο που δείχνει η node.

Πολυπλοκότητα: **O(1)** (σταθερού χρόνου εντολές)

public Arc remove(int from, int to):

Υλοποίηση:

-Ορισμός current που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Αναζήτηση κόμβου με τα δεδομένα to και from (δηλαδή επανάληψη όσο ο πρώτος κόμβος με τα στοιχεία αυτά δεν βρίσκεται).

-Επιστρέφει null αν δεν υπάρχει καθόλου τέτοιος κόμβος στην λίστα (Δηλαδή, ο επόμενος του κόμβου δείκτη είναι η ουρά).

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο σε κάθε επανάληψη.

-Αφαίρεση του κόμβου στον οποίο δείχνει η current από τη λίστα.

(Σύνδεση προς τα μπροστά του προηγούμενου της current [εκείνον στον οποίο η current δείχνει προς τα πίσω] με τον επόμενο της current [εκείνον στον οποίο η current δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της current [εκείνον στον οποίο η current δείχνει προς τα μπροστά] με τον προηγούμενο της current [εκείνον στον οποίο η current δείχνει προς τα πίσω] - Κατάργηση των δεικτών prev και next της current).

- Επιστροφή της ακμής που περιέχεται εν τελεί στον κόμβο που δείχνει η current.

Πολυπλοκότητα: **n*O(1)=O(n)**

(Λόγω βρόγχου while που κάνει προσπέλαση μέχρι και όλων των κόμβων της λίστας [Αν ο ζητούμενος κόμβος δεν βρίσκεται στη λίστα.] Βέβαια, αν ο ζητούμενος κόμβος είναι ο τελευταίος, δεν θα εκτελεστεί η τελευταία επανάληψη λόγω της συνθήκης της και θα γίνουν, δηλαδή, n-1 επαναλήψεις.)

public Arc arc(int from, int to):

Υλοποίηση:

-Ορισμός current που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Αναζήτηση κόμβου με τα δεδομένα to και from (δηλαδή επανάληψη όσο ο πρώτος κόμβος με τα στοιχεία αυτά δεν βρίσκεται).

-Επιστρέφει null αν δεν υπάρχει καθόλου τέτοιος κόμβος στην λίστα (Δηλαδή, ο επόμενος του κόμβου δείκτη είναι η ουρά).

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο σε κάθε επανάληψη.

- Επιστροφή αντιγράφου της ακμής που περιέχεται εν τελεί στον κόμβο που δείχνει η current.

Πολυπλοκότητα: **n*O(1)=O(n)**

(Λόγω βρόγχου while που κάνει προσπέλαση μέχρι και όλων των κόμβων της λίστας [Αν ο ζητούμενος κόμβος δεν βρίσκεται στη λίστα.] Βέβαια, αν ο ζητούμενος κόμβος είναι ο τελευταίος, δεν θα εκτελεστεί η τελευταία επανάληψη λόγω της συνθήκης της και θα γίνουν, δηλαδή, n-1 επαναλήψεις.)

public ArcList arcWeightsIn(int lb, int ub):

Υλοποίηση:

-Ορισμός current που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Κατασκευάζεται καινούργια βοηθητική λίστα.

-Αναζήτηση κόμβου με weight στο δεδομένο εύρος. Δηλαδή:

-Επανάληψη όσο ο κόμβος δείκτης δεν έχει φτάσει στο τέλος της λίστας.

-Έλεγχος και εισαγωγή αντιγράφων των ακμών με βάρος στα δεδομένα όρια, στην βοηθητική λίστα.

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο μετά από κάθε έλεγχο.

- Επιστροφή της βοηθητικής λίστας με όσες ακμές από την αρχική λίστα, είχαν βάρος στα δεδομένα όρια.

Πολυπλοκότητα:**n*O(1) = O(n)**

(Λόγω βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας.)

public ArcList heaviestArcs(int k):

Υλοποίηση:

-Ορισμός μεταβλητής μετρητή βημάτων(i), μεταβλητής που κρατά το μέγεθος της λίστας(s) και μεταβλητής που κρατά το μεγαλύτερο βάρος από όλες τις ακμές για τις οποίες έχει γίνει σύγκριση(maxw).

-Ορισμός current που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Αν δίνεται αρνητικό ή μηδενικό k, τότε δεν εκτελείται η διαδικασία, ενώ αν το k είναι μεγαλύτερο του μεγέθους της λίστας, τότε εξισώνεται με το μέγεθος της λίστας.

-Κατασκευάζονται καινούργιες βοηθητικές λίστες(arc1, arc2).

-Εισαγωγή αντιγράφων όλων των ακμών στην πρώτη βοηθητική λίστα.

-Ορισμός δείκτη max, που δείχνει στη βαρύτερη ακμή από όλες τις ακμές για τις οποίες έχει γίνει σύγκριση.

- Η παρακάτω διαδικασία εκτελείται k φορές:

-Μετατόπιση του δείκτη current στην αρχή της βοηθητικής λίστας.

- Αρχικοποίηση του πρώτου κόμβου και του αντίστοιχου βάρους της ακμής του ως το μεγαλύτερο (στις max και maxw αντίστοιχα), για να μπορεί να γίνει η πρώτη σύγκριση κάθε φορά.

- Προσπελαύνεται η λίστα. Αν η ακμή ενός κόμβου είναι βαρύτερη από την ακμή της max, τότε ο δείκτης max δείχνει σε αυτή και η maxw παίρνει για τιμή το βάρος της.

-Βρίσκεται η μεγαλύτερη ακμή για αυτή την επανάληψη. Μετά αντίγραφο της ακμής του κόμβου στον οποία δείχνει η max εισάγεται στη δεύτερη βοηθητική λίστα.

-Ύστερα διαγράφεται από την πρώτη βοηθητική λίστα, έτσι ώστε να μην ξαναληφθεί υπόψιν κατά την επόμενη εύρεση του max.

(-Σύνδεση προς τα μπροστά του προηγούμενου της max [εκείνον στον οποίο η max δείχνει προς τα πίσω] με τον επόμενο της max [εκείνον στον οποίο η max δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της max [εκείνον στον οποίο η max δείχνει προς τα μπροστά] με τον προηγούμενο της max [εκείνον στον οποίο η max δείχνει προς τα πίσω])

- Κατάργηση των δεικτών prev και next της max).

- Επιστροφή λίστας με τις k βαρύτερες ακμές.

*(Λόγω του ότι, οι ακμές βρίσκονται με τη σειρά από τη μεγαλύτερη σε βάρος προς την μικρότερη και κάθε φορά εισάγονται στο τέλος της καινούργιας λίστας, θα επιστραφεί μια καινούργια λίστα με τις k μεγαλύτερες ακμές, ταξινομημένες από τη μεγαλύτερη προς την μικρότερη σε βάρος)

Πολυπλοκότητα: **O(n)+n*O(1)+n*(n*O(1)+O(1))=O(n)+O(n^2)=O(n^2)**

(Λόγω εκτέλεσης size(), βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας για να γίνει η εισαγωγή όλων των κόμβων στην βοηθητική(σταθερού χρόνου εντολές). Ύστερα ο βρόγχος for μπορεί να γίνει έως και n φορές [στην περίπτωση που όλες οι ακμές έχουν διαφορετικά βάρη και ζητούνται οι k=n βαρύτερες ακμές]. Μέσα στην for, υπάρχει εμφωλευμένος βρόγχος while, που κάνει προσπέλαση όλων των κόμβων της λίστας (την πρώτη φορά n επαναλήψεις) για να γίνει η εύρεση του max και μετά γίνεται η διαγραφή του max από τη βοηθητική λίστα (σταθερού χρόνου εντολές).)

private int HeaviestInfluence():

Υλοποίηση:

-Ορισμός μεταβλητής με το to, με το βαρύτερο υπογράφημα επιρροής (maxpoint) - μεταβλητής με το βάρος του βαρύτερου υπογραφήματος επιρρόης (max) - βοηθητικής μεταβλητής μέτρησης του συνολικού αθροίσματος του κάθε to , για να βρεθεί το βάρος του κάθε υπογραφήματος επιρροής(sumtmp), βοηθητική μεταβλητή που κρατά το to για κάθε προσπέλαση, για την εύρεση του βάρους του υπογραφήματος επιρροής του(s1).

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Κατασκευάζεται καινούργια βοηθητική λίστα.

-Εισαγωγή αντιγράφων όλων των ακμών στην πρώτη βοηθητική λίστα.

-Αρχικοποίηση κόμβων-δεικτών στην αρχή της βοηθητικής λίστας(current1 – current2).

-Αρχικοποίηση sumtmp.

-Αρχικοποίηση του πρώτου to στο s1.

-Γίνεται εύρεση του αθροίσματος του υπογραφήματος επιρροής του πρώτου to στην λίστα και αρχικοποιείται κατευθείαν το to αυτό ως maxpoint και το άθροισμά του ως max. Αυτό γίνεται για να έχουν οι μεταβλητές αυτές μια πρώτη τιμή και να γίνουν μετά οι συγκρίσεις από το επόμενο to και μετά.

-Η εύρεση του αθροίσματος του υπογραφήματος επιρροής του κάθε to γίνεται ως εξής:

-Γίνεται προσπέλαση των κόμβων της βοηθητικής λίστας. Αν κάποιος από αυτούς έχει ακμή με to= s1, τότε το βάρος του προστείθεται στην sumtmp και ύστερα αφαιρείται από την λίστα αυτή (για να αφαιρεθεί εν τέλει όλο το υπογράφημα επιρροής και να μην ξαναληφθεί υπόψην και γίνουν επιπλέον έλεγχοι και εντολές χωρίς λόγο).

-Η αφαίρεση του κάθε κόμβου τώρα γίνεται ως εξής:

-Ο tmp κρατά τη θέση του προηγουμένου του κόμβου που θα αφαιρεθεί.

-Ύστερα διαγράφεται από την λίστα:

(-Σύνδεση προς τα μπροστά του προηγούμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω] με τον επόμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] με τον προηγούμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω])

-Μετά επιστρέφει τη θέση στην current2 και έτσι όταν πάει στο επόμενο, να καταλήγει στον επόμενο κόμβο από αυτόν που αφαιρέθηκε.

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο μετά από κάθε έλεγχο.

- Αρχικοποιείται κατευθείαν το s1 ως maxpoint και το άθροισμά του ως max.

-Η διαδικασία ουσιαστικά μετά, ξεκινά με τον επόμενο του πρώτου to

που αρχικοποιήθηκε, αφού οι ακμές του έχουν αφαιρεθεί.

-Μέσω προσπέλασης της λίστας μέσω εξωτερικής while και με άλλον βοηθητικό δείκτη current1, γίνεται εύρεση του αθροίσματος του υπογραφήματος επιρροής του κάθε to.

-Αρχικοποίηση του to για το οποίο θα βρεθεί το άθροισμα σε αυτήν την επανάληψη στην s1.

- Μηδενισμός του sumtmp.

-Γίνεται εύρεση του αθροίσματος του υπογραφήματος επιρροής του επιλεγμένου to ως εξής:

-Γίνεται προσπέλαση των κόμβων της βοηθητικής λίστας. Αν κάποιος από αυτούς έχει ακμή με to= s1, τότε το βάρος του προστείθεται στην sumtmp και ύστερα αφαιρείται από την λίστα αυτή (για να αφαιρεθεί εν τέλει όλο το υπογράφημα επιρροής και να μην ξαναληφθεί υπόψην και γίνουν επιπλέον έλεγχοι και εντολές χωρίς λόγο).

-Η αφαίρεση του κάθε κόμβου τώρα γίνεται ως εξής:

-Ο tmp κρατά τη θέση του προηγουμένου του κόμβου που θα αφαιρεθεί.

-Ύστερα διαγράφεται από την λίστα:

(-Σύνδεση προς τα μπροστά του προηγούμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω] με τον επόμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] με τον προηγούμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω])

-Μετά επιστρέφει τη θέση στην current2 και έτσι όταν πάει στο επόμενο, να καταλήγει στον επόμενο κόμβο από αυτόν που αφαιρέθηκε.

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο μετά από κάθε έλεγχο.

-Αν το άθροισμα του υπογραφήματος επιρροής είναι μεγαλύτερο του max, τότε γίνεται αυτό το max και το to του γίνεται το maxpoint.

- Γίνεται αρχικοποίηση στον κόμβο δείκτη, του επόμενου κόμβου μετά την κεφαλή της βοηθητικής, έτσι ώστε να ξαναξεκινήσει μετά από την αρχή η εύρεση του καινούργιου αθροίσματος στην πια ανανεωμένη λίστα .

-Μετά το πέρας όλων, γίνεται επιστροφή του to, που έχει το βαρύτερο υπογράφημα επιρροής.

Πολυπλοκότητα: **n*O(1)+ n*O(1)+n*n*O(1)=O(n^2)**

(Λόγω βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας για να γίνει η εισαγωγή όλων των κόμβων στην βοηθητική (σταθερού χρόνου εντολές). Ύστερα μέσω while πάλι βρίσκεται το άθροισμα του πρώτου to (σταθερού χρόνου εντολές) και προσπελαύνεται όλη η βοηθητική λίστα. Μετά βρόγχος while εκτελείται για κάθε διαφορετικό to που υπάρχει στη λίστα (άρα n φορές αν κάθε to είναι διαφορετικό) και μέσα σε αυτόν είναι εμφωλευμένη μια άλλη while με την οποία βρίσκεται το άθροισμα του επιλεγμένου to (σταθερού χρόνου εντολές) και προσπελαύνεται όλη η βοηθητική λίστα κάθε φορά.)

public ArcList leastInfluence(int sum):

Υλοποίηση:

- Ορισμός βοηθητικής μεταβλητής μέτρησης του συνολικού αθροίσματος του κάθε to , για να βρεθεί το βάρος του κάθε υπογραφήματος επιρροής(sumtmp) - βοηθητικής μεταβλητή που κρατά το to για κάθε προσπέλαση για την εύρεση του βάρους του υπογραφήματος επιρροής του (s1).

-Ορισμός δείκτη current1.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Ορισμός δεικτών current2 και tmp.

-Κατασκευάζονται καινούργιες βοηθητικές λίστες(arc1, arc2).

-Εισαγωγή αντιγράφων όλων των ακμών στην πρώτη βοηθητική λίστα.

-Αρχικοποίηση κόμβων-δεικτών στην αρχή της βοηθητικής λίστας

-Μέσω προσπέλασης της λίστας μέσω εξωτερικής while και με βοηθητικό δείκτη current1, γίνεται εύρεση του αθροίσματος του υπογραφήματος επιρροής του κάθε to.

-Μηδενισμός του sumtmp κάθε φορά για να ανανεώνεται και να βρίσκεται το άθροισμα του κάθε υπογραφήματος επιρροής σωστά.

-Αρχικοποίηση του to στην s1 για το οποίο θα βρούμε το συνολικό βάρος του υπογραφήματος επιρροής του, κάθε φορά.

-Γίνεται εύρεση του αθροίσματος του υπογραφήματος επιρροής του επιλεγμένου to ως εξής:

-Γίνεται προσπέλαση των κόμβων της βοηθητικής λίστας. Αν κάποιος από αυτούς έχει ακμή με to= s1, τότε το βάρος του προστείθεται στην sumtmp και ύστερα αφαιρείται από την λίστα αυτή (για να αφαιρεθεί εν τέλει όλο το υπογράφημα επιρροής και να μην ξαναληφθεί υπόψην και γίνουν επιπλέον έλεγχοι και εντολές χωρίς λόγο).

-Η αφαίρεση του κάθε κόμβου τώρα γίνεται ως εξής:

-Ο tmp κρατά τη θέση του προηγουμένου του κόμβου που θα αφαιρεθεί.

-Ύστερα διαγράφεται από την λίστα:

(-Σύνδεση προς τα μπροστά του προηγούμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω] με τον επόμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της current2 [εκείνον στον οποίο η current2 δείχνει προς τα μπροστά] με τον προηγούμενο της current2 [εκείνον στον οποίο η current2 δείχνει προς τα πίσω])

-Μετά επιστρέφει τη θέση στην current2 και έτσι όταν πάει στο επόμενο, να καταλήγει στον επόμενο κόμβο από αυτόν που αφαιρέθηκε.

- Γίνεται μετατόπιση του κόμβου δείκτη στον επόμενο κόμβο μετά από κάθε έλεγχο.

-Αν το άθροισμα του υπογραφήματος επιρροής είναι μεγαλύτερο του sum, τότε γίνεται εισαγωγή όσων ακμών έχουν to, ίδιο με το επιλεγμένο(s1) (από την κανονική λίστα όχι την πρώτη βοηθητική, γιατί από την πρώτη βοηθητική έχουν αφαιρεθεί), στην δεύτερη βοηθητική λίστα

-Επιστροφή στην αρχή της λίστας, πριν την επόμενη επανάληψη

-Μετά το πέρας όλης της διαδικασίας, γίνεται επιστροφή λίστας με όλες τις ακμές των to, με υπογραφήματα επιρροής βαρύτερα του sum.

Πολυπλοκότητα: **n*O(1)+n*(n*O(1)+n*O(1))=O(n)+O(n^2)+O(n^2)=O(n^2)**

(Λόγω βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας για να γίνει η εισαγωγή όλων των κόμβων στην βοηθητική (σταθερού χρόνου εντολές). Μετά βρόγχος while εκτελείται για κάθε διαφορετικό to που υπάρχει στη λίστα (άρα n φορές αν κάθε to είναι διαφορετικό) και μέσα σε αυτόν είναι εμφωλευμένη μια άλλη while με την οποία βρίσκεται το άθροισμα του επιλεγμένου to (σταθερού χρόνου εντολές) και προσπελαύνεται όλη η βοηθητική λίστα κάθε φορά, καθώς και ύστερα (αν sumtmp>=sum) γίνεται προσπέλαση όλης της κανονικής λίστας και εισαγωγή όλων των κόμβων με to=s1 στην δεύτερη βοηθητική λίστα (σταθερού χρόνου εντολές).)

public ArcList topInfluencers(int k):

Υλοποίηση:

-Ορισμός μεταβλητής με το to, με το βαρύτερο υπογράφημα επιρροής (maxpoint) - μεταβλητής μετρητή βημάτων(i) - μεταβλητής που κρατά το μέγεθος της λίστας(s).

-Ορισμός current1 που δείχνει στον επόμενο της head.

- Δεν εκτελείται η διαδικασία, αν η λίστα είναι κενή (δηλαδή ο επόμενος της head είναι η tail).

-Αν δίνεται αρνητικό ή μηδενικό k, τότε δεν εκτελείται η διαδικασία, ενώ αν το k είναι μεγαλύτερο του μεγέθους της λίστας, τότε εξισώνεται με το μέγεθος της λίστας.

-Κατασκευάζονται καινούργιες βοηθητικές λίστες(arc4, arc5).

-Εισαγωγή αντιγράφων όλων των ακμών στην πρώτη βοηθητική λίστα.

-Αρχικοποίηση κόμβου-δείκτη tmp1.

- Η παρακάτω διαδικασία εκτελείται k φορές:

-Εκτέλεση της HeaviestInfluence για τη βοηθητική λίστα arc4,

και εισχώρηση του to με το βαρύτερο υπογράφημα επιρροής στην maxpoint.

-Μετατόπιση του δείκτη current στην αρχή της βοηθητικής λίστας.

-Προσπελαύνεται η λίστα arc4 μέσω μιας while και αν το to ενός arc ταυτίζεται με το maxpoint, τότε εισάγεται αντίγραφο του arc αυτού στην δεύτερη βοηθητική λίστα(arc5) και ο κόμβος που το έχει, αφαιρείται από την πρώτη βοηθητική λίστα (για να αφαιρεθεί εν τέλει όλο το υπογράφημα επιρροής και να μην εισαχθούν παραπάνω από μία φορά η ακμές στην arc5).

-Η αφαίρεση του κάθε κόμβου τώρα γίνεται ως εξής:

-Ο tmp1 κρατά τη θέση του προηγουμένου του κόμβου που θα αφαιρεθεί.

-Ύστερα διαγράφεται από την λίστα:

(-Σύνδεση προς τα μπροστά του προηγούμενου της current1 [εκείνον στον οποίο η current1 δείχνει προς τα πίσω] με τον επόμενο της current1 [εκείνον στον οποίο η current1 δείχνει προς τα μπροστά] - Σύνδεση προς τα πίσω του επόμενου της current1 [εκείνον στον οποίο η current1 δείχνει προς τα μπροστά] με τον προηγούμενο της current1 [εκείνον στον οποίο η current1 δείχνει προς τα πίσω])

-Μετά επιστρέφει τη θέση στην current2 και έτσι όταν πάει στο επόμενο, να καταλήγει στον επόμενο κόμβο από αυτόν που αφαιρέθηκε.

-Υπάρχει μία συνθήκη, όπου εξασφαλίζει ότι, δεν θα συνεχιστεί η διαδικασία (break;) αν η πρώτη βοηθητική λίστα έχει ήδη αδειάσει (λόγω του ότι μετά από κάθε επανάληψη της for, αφαιρείται ένα ολόκληρο υπογράφημα επιρροής από αυτή).

-Μετά το πέρας της όλης διαδικασίας γίνεται επιστροφή λίστας με αντίγραφα των ακμών των k βαρυτέρων υπογραφημάτων επιρροής.

Πολυπλοκότητα: **O(n)+n*O(1)+ n*(O(n^2)+n*O(1)+O(n))=O(n)+O(n)+O(n^3)=O(n^3)**

(Λόγω εκτέλεσης size() και βρόγχου while που κάνει προσπέλαση όλων των κόμβων της λίστας για να γίνει η εισαγωγή όλων των κόμβων στην βοηθητική(σταθερού χρόνου εντολές). Ύστερα ο βρόγχος for μπορεί να γίνει έως και n φορές [στην περίπτωση που όλες οι ακμές έχουν διαφορετικά βάρη και ζητούνται οι k=n βαρύτερες ακμές]. Μέσα στην for, εκτελείται μία φορά η HeaviestInfluence και υπάρχει εμφωλευμένος βρόγχος while, που κάνει προσπέλαση όλων των κόμβων της λίστας, εισάγονται στην δεύτερη βοηθητική λίστα όλες οι ακμές με to=maxpoint και αφαιρούνται μετά η

καθεμία(σταθερού χρόνου εντολές). Τέλος εκτελείται μέσα στη for και άλλη μία φορά η size()).

Σύνθετες Ερωτήσεις:

(α):

Γίνεται εκτέλεση των δύο παρακάτω μεθόδων:

-Εκτέλεση της heaviestArcs για δεδομένο k **[O(n^2)]**

-Εκτέλεση της topInfluencers για k=1 **[O(n^3)]**

Εφόσον θέλουμε το βαρύτερο υπογράφημα επιρροής, τότε εκτελούμε

την topInfluencers για k=1.

Πολυπλοκότητα: **O(n^2)+O(n^3)=O(n^3)**

(β):

Γίνεται εκτέλεση των δύο παρακάτω μεθόδων:

-Εκτέλεση της arcWeightsIn για δεδομένα lb και ub **[O(n)]**

-Εκτέλεση της topInfluencers για k=1 **[O(n^3)]**

Εφόσον θέλουμε το βαρύτερο υπογράφημα επιρροής, τότε εκτελούμε

την topInfluencers για k=1.

Πολυπλοκότητα: **O(n)+O(n^3)=O(n^3)**

(γ):

Γίνεται εκτέλεση των δύο παρακάτω μεθόδων:

-Εκτέλεση της topInfluencers για δεδομένο k **[O(n^3)]**

-Εκτέλεση της arcWeightsIn για δεδομένα lb και ub **[O(n)]**

Πολυπλοκότητα: **O(n^3)+O(n)=O(n^3)**

(δ):

Γίνεται εκτέλεση των δύο παρακάτω μεθόδων:

-Εκτέλεση της leastInfluence για δεδομένο sum **[O(n^2)]**

-Εκτέλεση της heaviestArcs για δεδομένο k **[O(n^2)]**

Πολυπλοκότητα: **O(n^2)+O(n^2)=O(n^2)**

***ΣΗΜΕΙΩΣΗ:** **Η Άσκηση αυτή έχει περάσει από 3 εκδόσεις-φάσεις. Αρχικά γινόταν εισαγωγή κόμβων στην λίστα, με ταυτόχρονη σειριακή ταξινόμηση ως προς τα βάρη των ακμών (weight). Αυτό είχε ως αποτέλεσμα η insert να έχει πολυπλοκότητα O(n), αλλά έτσι ευνοούταν στην πολυπλοκότητα η heaviestArcs. Ύστερα, αποφάσισα να κάνω ταξινόμηση ως προς τον αριθμό του to, της κάθε ακμής (πάλι με insert πολυπλοκότητας O(n)) . Αυτό είχε ως αποτέλεσμα να είναι όλα τα ίδια to μαζεμένα, κι έτσι να γλυτώνω μια while (λόγω του ότι γινόταν ουσιαστικά μία προσπέλαση της λίστας, αντί μία για κάθε to), σε κάποιες μεθόδους (π.χ. HeaviestInfluence) και εν τέλει η πολυπλοκότητα σε αυτές να μειώνεται. Ωστόσο, στο τελικό της στάδιο, η εισαγωγή κόμβων στην εργασία γίνεται χωρίς ταξινόμηση, με απλή εισαγωγή απλά στο τέλος της λίστας. Την πήρα αυτήν την απόφαση για δύο λόγους. Πρώτον, διότι έτσι η insert γινόταν O(1) πολυπλοκότητας και όπως αναφέρεται στην εκφώνηση, τουλάχιστον οι βασικές μέθοδοι (όπως είναι η insert) πρέπει να είναι ορθά υλοποιημένες (άρα και με τη μικρότερη δυνατή πολυπλοκότητα). Δεύτερον, με αυτόν τον τρόπο (παρά την αύξηση σειρών κώδικα και πολυπλοκότητας σε κάποιες μεθόδους), η πολυπλοκότητα σε άλλες μεθόδους (π.χ. topInfluencers), οι οποίες έκαναν χρήση της insert, μίκρυνε και θεώρησα ότι αυτός ο τρόπος επέφερε την μικρότερη δυνατή πολυπλοκότητα συνολικά, σε σχέση με τους δύο προηγούμενους τρόπους.**

ΣΩΤΗΡΙΟΣ ΔΗΜΗΤΡΑΚΟΥΛΑΚΟΣ – Ε20040