Algorithms Spring 2024
Project Report

Δημήτρης Παπαδημητρίου-Βασίλης Μαδέσης
AEM: 03750-03748

Εύρεση CPL:

Για να βρούμε το CPL ενός γράφου χρησημοποιήσαμε Breadth First Search ώστε να υπολογίσουμε όλα τα μήκοι των σύντομων μονοπάτιων μεταξύ όλων των κόμβων.Συγκεκριμένα υλοποιούμε μια συνάρτηση η οποία λαμβάνει ως input το source node και έχει ένα πίνακα που κρατάει τις αποστάσεις όλων των κόμβων από τον κόμβο source μαζί με ένα πίνακα με τους γονείς κάθε κόμβου.Ο πίνακας γονιών θα χρησιποιηθεί αργότερα για το B ερώτημα όπου θα εξηγήσουμε την χρήση του.Τελός η συνάρτηση αυτή επιστρέφει τους πίνακες αύτους ενώ αν της δωθεί συγκεκριμένος destination κομβός τότε θα τρέχει μέχρι να βρεί το κόμβο αυτό επιστρέφοντας επιτυχία με 1 και αποτυχία με 0.

Για την υλοποίηση της Breadth First Search χρήσιμοποιήουμε μια queue (FIFO ούρα) ώστε να τοποθετούμε μέσα της τους κόμβους που πρέπει να επισκεφτούμε μετά από μια επίσκεψη κόμβου σε κάθε επανάληψη.

Η συνάρτηση λειτουργεί ως εξής:

```
pathSearch()
distance[size of graph] -> 0

Insert in queue node S
while(Queue is not empty)
   parent node <- remove from queue the first element

   for all child nodes of parent node
      if(distance[child node] is 0 && child node != S)
         Put child node into queue
         distance[child node] = distance[parent node] + 1
         parent[child node] = parent node
      else if(distance[child node] == distance[parent node] + 1)
         parent[child node] -> add parent to the list of parents for
              child node
      if (distance[child node] == distance[parent node] + 1 && child
          node == destination node)
         return 1
   end
return 0
end
```

Για να βρούμε το CPL εκτελούμε κατα επανάληψη την προηγούμενη συνάρτηση για όλους τους κόμβους και μηδενίζουμε τις επαναλαμβανόμενες αποστάσεις που προκύπτουν στον πίνακα αποστάσεων που μας επιστρέφει η συνάρτηση και έπειτα προσθέτουμε στο συνολικό άθροισμα, το άθροισμα των CPL του πίνακα αυτού.

```
Sum of all cpls = 0;

for all nodes inside the graph
   distances array = execute pathSearch
   distances array -> distance array with zeroed the recurring cpls
   Sum of all cpls = distances array + Sum of all cpls
end

cpl = Sum of all cpls / binomial coefficient(number of nodes,2)
```

Για να υπολογίσουμε τον συνολικό αριθμό των συντομοτέρων μονοπατιών (SPs) μεταξύ κάθε δυνατού ζεύγους κόμβων j και k στα οποία μεσολαβεί κάθε ακμή e του γραφήματος.Θα χρησιμοποιήσουμε τον πίνακα γονέων που μας επιστρέφει η συνάρτηση pathSearch().

Ο πίνακας γονέων είναι ένας πίνακας που σε κάθε θέση i έχει τον γονέα του κόμβου $i+1$.Γονέας ενός κόμβου είναι ο γειτονικός κόμβος από τον οποίο διαβάστηκε και μπήκε στον πίνακα αποστάσεων ο κόμβος παιδί(στην περιπτωσή μας ο κόμβος $i+1$).Σε περίπτωση ύπαρξης δύο ή περισσότερων σύντομων μονοπατιών μεταξύ δύο κόμβων ο κόμβος destination θα έχει δύο γονείς καθώς και οι δύο είναι μέρος σύντομου μονοπατιού.

Οι συμλήρωση του πίνακα γονέων φαίνεται στο παρακάτω code snippet της συνάρτησης pathSearch:

```
if(distance[child node] is 0 && child node != S)
        Put child node into queue
        distance[child node] = distance[parent node] + 1
        parent[child node] = parent node
else if(distance[child node] == distance[parent node] + 1)
        parent[child node] -> add parent to the list of parents for
             child node
```

Αν ο κόμβος έχει παραπάνω από έναν γονέα τότε δημιουργείται μια λίστα με του γονείς του κόμβου αυτού στην αντίστοιχη θέση του πίνακα.

Προσθέτοντας στον προήγουμενο κώδικα, για κάθε επανάληψη της 'λούπας' παίρνουμε τον πίνακα γονέων και για κάθε ζευγάρι κόμβων source-destination διατρέχουμε το πίνακα ξεκίνοντας από το destination και ακολουθώντας τον γονέα του και από εκεί ακολουθώντας τον γονέα του γονέα του ,φτάνουμε στο source καταγράφοντας τα edges που συμμετείχαν στο δρόμο μέχρι το source.

Το path για κάθε ζεύγος καθώς και τα edges που συμμετέχουν σε αυτό βρίσκεται από τον ψευδοκώδικα:

```
countEdgesPath()
   current node = destination node
   while(parent of current node != Source)
     array of edges [parent node][current node] ++
     array of edges [parent node][current node] ->add info about the edge
   end

   if (there are more than one parents for destination node)
      repeat the process for the other parent of destination node
end
```

Ο συνολικός ψευδοκώδικας προκύπτει:

```
Sum of all cpls = 0;

for all nodes inside the graph
   distances array = execute pathSearch
   distances array -> distance array with zeroed the recurring cpls
   Sum of all cpls = distances array + Sum of all cpls

   for all combinations of sources and destination nodes inside a graph
      countEdgesPath() fill the edge array
   end
end

cpl = Sum of all cpls / binomial coefficient(number of nodes,2)
```

Τέλος έξω απο την ¨λούπα' κάνουμε merge sort το πίνακα edge array και επιστρέφουμαι το πρώτο edge του πίνακα το οποίο και θα αφαιρέσουμε.Το δύο προηγούμενα ερωτήματα τα εκτελεί η συνάρτηση cpl sp.

```
cpl_sp()
Sum of all cpls = 0;

for all nodes inside the graph
   distances array = execute pathSearch
   distances array -> distance array with zeroed the recurring cpls
   Sum of all cpls = distances array + Sum of all cpls

   for all combinations th source and destination nodes inside the graph
      countEdgesPath() fill the edge array
   end
end

cpl = Sum of all cpls / binomial coefficient(number of nodes,2)
mergesort(edge array)

return top elemnent in edge array
end
```

Συνεπώς το complexity της συνάρτησης προκύπτει ως εξής:

Το complexity της αναζήτησης είναι $O(V + E)$ όπου V κόμβοι και E ακμές.

Για την συνάρτηση μηδενισμού έχουμε complexity $O(V)$ όπως το ίδιο και για την συνάστηση αθροίσματος.

Η συνάστηση επαναληπτική εκτέλεση της συνάστησηςcountEdgesPath() έχει comlexity $O(V*BiggestPathLength)$ καθώς εκτελείται περίπου $O(V)$ φορές για ένα δεδομένο πίνακα γονέων και η διάτρεξη του πίνακα γονέων έχει στην μεγαλύτερη περίπτωση μεταξύ όλων των $O(BiggestPathLength)$ πράξεις.

Άρα το συνολικό complexity εφόσον γίνονται V επαναλήψης είναι
$O(V)*[O(V+E)+O(V)+O(V)+O(V*BiggestPathLength)] \ = \ O(V^2+VE)$

Εφόσον έχουμε εκτελέσει την συνάρτηση cpl sp τότε κάνουμε αφαίρεση της ακμής μεταξύ των δύο κόμβων αφαιρόντας τις μεταξύ ακμές από τις αντίστοιχες λίστες γειτνίασης τους.

Έπειτα για να ελένξουμε για συνοχή του γράφου εκτελούμε την pathSearch με στόχο να βρούμε το κόμβο με τον οποίο κόπηκε η απευθείας σύνδεση. Εάν τον βρούμε θα επιστραφεί επιτυχία από την συνάρτηση και θα σημαίνει οτι ο γράφος παραμένει συνεκτικός αλλίως σε περίπτωση αποτυχίας θα σημαίνει οτι ο γράφος χωρίστηκε.