

**Algorithms Spring 2024**

**Project Report**

*Δημήτρης Παπαδημητρίου-Βασίλης Μαδέσης*

*AEM: 03750-03748*

Εύρεση CPL:

Για να βρούμε το CPL ενός γράφου χρησιμοποιήσαμε 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 χρησιμοποιηθεί αργότερα για το B ερώτημα όπου θα εξηγήσουμε την χρήση του. Τελός η συνάρτηση αυτή επιστρέφει τους πίνακες αυτούς ενώ αν της δοθεί 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 χρησιμοποιήουμε 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 συμπλήρωση 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 προηγούμενο κώδικα, για κάθε επανάληψη της 'λούπας' παίρνουμε τον πίνακα γονέων και για κάθε ζευγάρι κόμβων source-destination διατρέχουμε το πίνακα ξεκινώντας 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
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 από την "λούπα" κάνουμε merge sort το πίνακα edge array και επιστρέφουμε 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 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)
mergesort(edge array)

return top element in edge array
end
```

---