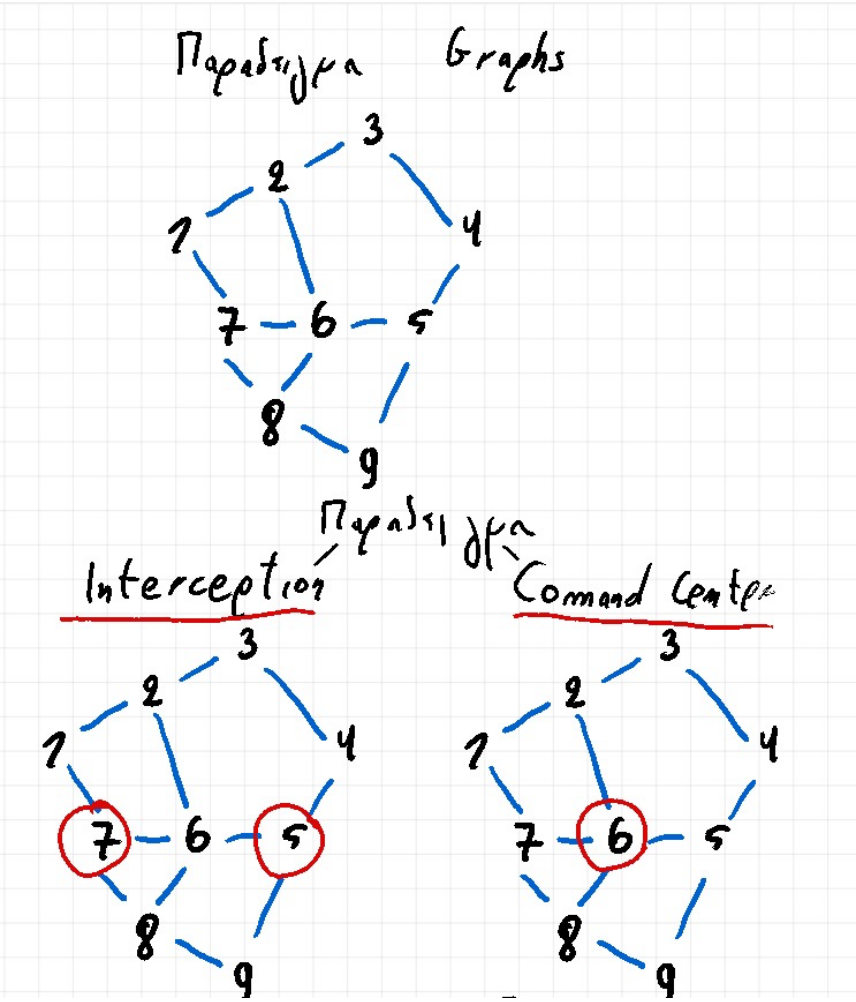
# Project

Orestis Panopoulos, Giannis Giannoulas, Panagiotis Nanousis

03269, 03534, 03599

## Introduction

Στην εργασία αυτή μας ζητείται να βρούμε την συσχέτιση μεταξύ των κέντρων κρυφακοής και κέντρο διοίκησης.
Για να μπορέσουμε να απαντήσουμε στο ερώτημα αυτο αναπτύξετε ένα πρόγραμμα σε C που παίρνει τα αρχεία με τους κομβους, φτιάχνει τα γραφήματα και χρησιμοποιώντας τους αλγόριθμους που θα αναλύσουμε παρακάτω, βρίσκει το betweenness και το closeness rankings των κόμβων. Έπειτα υπολογίζει τον Kendall Coefficient ο οποίο μας δίνει μια ιδέα για την ομοιότητα των δύο rankings.
Το source code του προγράμματος μπορεί να βρεθεί στο github link και σε συνημμένο στο email.
https://github.com/Nanousis/Algorithms-Project

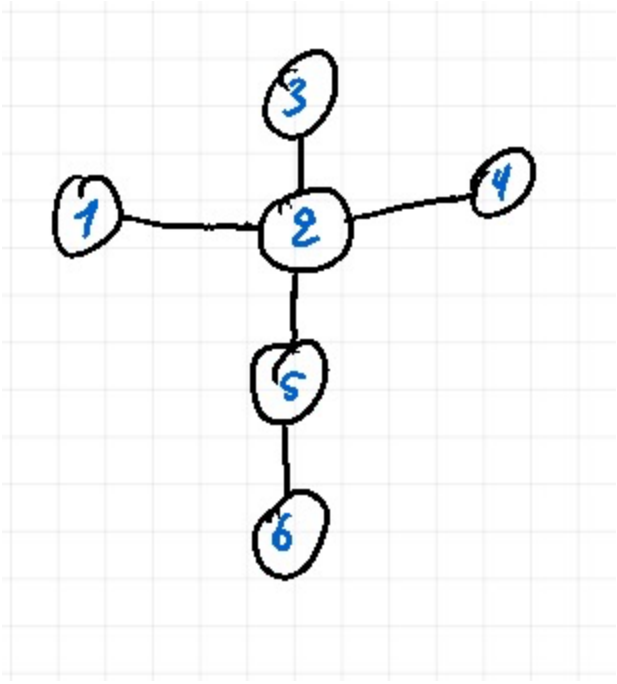



## Πρόβλημα Κέντρων Διοίκησης (Closeness)

Για να μπορέσουμε να αποφασίσουμε που πρέπει να βάλουμε τα κέντρα διοίκησης πρέπει να βρούμε τους κόμβους που έχουν την μικρότερη μέση απόσταση από τους άλλους κόμβους, να τους ταξινομήσουμε και να επιλέξουμε τον κόμβο που βρίσκεται πιό κοντά (closeness) στους άλλους. Έτσι δημιουργήσαμε τον αλγόριθμο 1 για να βρίσκουμε το ranking αυτό.

## Πρόβλημα Κρυφακοής (betweenness)

Παρομοίως για να μπορέσουμε να αποφασίσουμε που πρέπει να βάλουμε τις συσκευές κρυφακοής πρέπει να βρούμε τους κόμβους οι οποίοι μεσολαβούν στα shortest paths κατα την επικοινωνία, να τα ταξινομήσουμε και να επιλέξουμε τον κόμβο που μεσολαβεί (betweenness) στα περισσότερα shortest paths. Έτσι δημιουργήσαμε τον αλγόριθμο 2 για να βρίσκουμε το ranking αυτό.




Για να μπορέσουμε να κάνουμε παραδείγματα με τους αλγορίθμους θα χρησιμοποιήσουμε το απλό αυτό γράφημα.

## Αλγόριθμος 1

Για αυτόν τον αλγόριθμο θα υπολογίσουμε το μικρότερο μονοπάτι και ταυτόχρονα θα αποθηκεύσουμε αυτό το μονοπάτι σε έναν πίνακα για να μπορέσουμε αργότερα να υπολογίσουμε σε πόσα μικρότερα μονοπάτια ανήκει ο κάθε κόμβος. Σημαντικό πως στον αλγόριθμο αυτόν δεν προβλέπουμε τι θα γίνει αν μεταξύ δύο κόμβων έχουμε πολλαπλά μονοπάτια τα οποία έχουν το μήκος που εμείς θεωρούμε μικρότερο.

Για τους σκοπούς του υπολογισμού των συντομότερων μονοπατιών από κάθε κόμβο προς άλλον κόμβο θα χρησιμοποιήσουμε τον αλγόριθμο Floyd Warshall, ο οποίος με πηγή την Βικιπαίδεια https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
είναι ο εξής:

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each edge (u, v) do
    dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
for each vertex v do
    dist[v][v] ← 0
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if
```

Ο οποίος γνωρίζουμε πάλι από την βικιπαίδεια πως έχει πολυπλοκότητα $O(|V|^3)$.

Μπορούμε βέβαια να το αποδείξουμε και εμείς.
Περιπλοκότητα κάθε γραμμής:
1→ O(1) για δέσμευση μνήμης
2 και 3→ $O(|V|^2)$ για μεταφορά δεδομένων
4 και 5 → O(|V|) για μεταφορά δεδομένων
6, 7, 8, 9, 10, 11 → $O(2^* (|V|^3)) = O(|V|^3)$ αφού έχουμε τρεις βρόγχους επανάληψης με ένα if και ανάθεση τιμής μέσα

Άρα $O(1) + O(|V|^2) + O(|V|^3) + O(|V|) = O(|V|^3)$

Έτσι μετά την εφαρμογή του Floyd-Warshall προκύπτουν οι παρακάτω πίνακες.

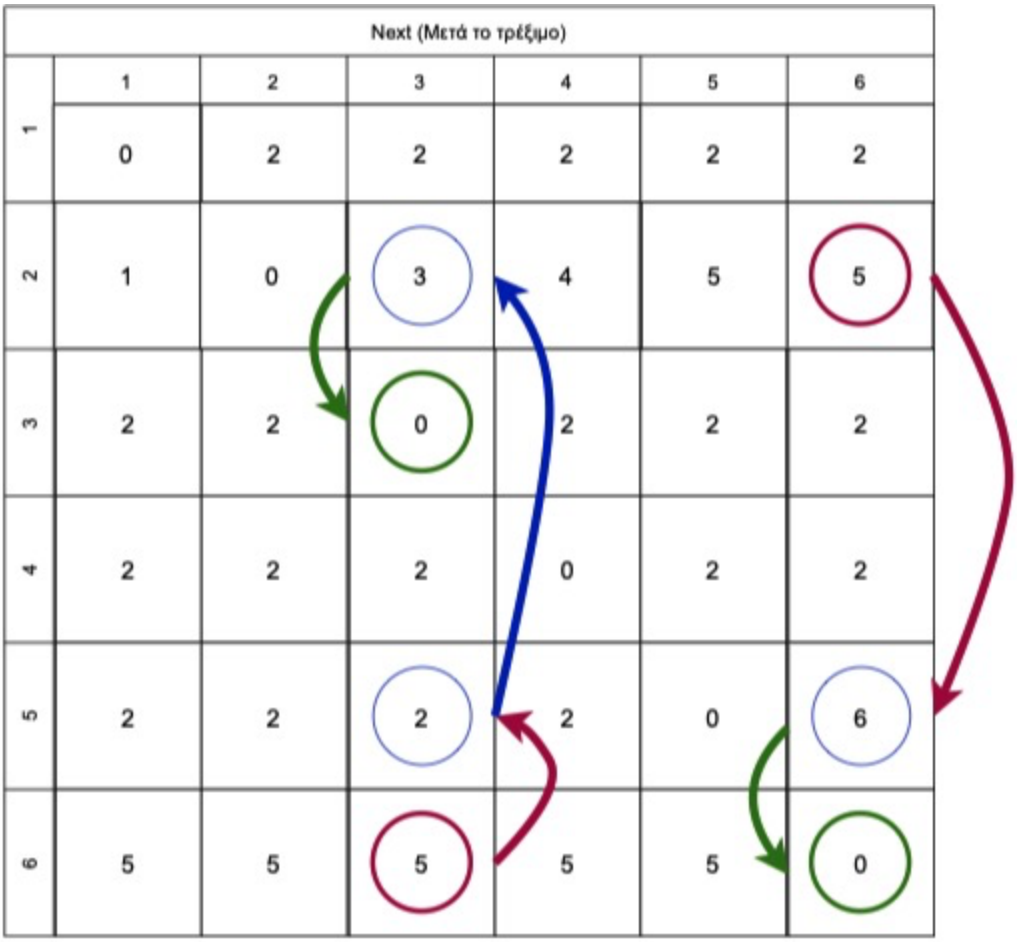
| Floyd-Warshall | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 1 | 2 | 2 | 2 | 2 |
| 2 | 1 | 0 | 1 | 1 | 1 | 2 |
| 3 | 2 | 1 | 0 | 2 | 2 | 3 |
| 4 | 2 | 1 | 2 | 0 | 2 | 3 |
| 5 | 2 | 1 | 2 | 2 | 0 | 1 |
| 6 | 3 | 2 | 3 | 3 | 1 | 0 |

Στον παραπάνω πίνακα Floyd-Warshall το κάθε κελί συμβολίζει την απόσταση μεταξύ των κόμβων (x, y)

Για την ταξινόμηση τον δεδομένων έχουμε ένα struct που κρατάει τον αριθμό του κόμβου (index) και το βάρος που έχει για το κάθε ranking.

```
typedef struct NodeCount{
    int index;
    int count;
} NodeCount;
```

Για να αποθηκεύσουμε τους κόμβους του συντομότερου μονοπατιού, αρκεί να προσθέσουμε έναν πίνακα $|V|^2$, έστω Next το όνομα του, στον οποίο η κάθε γραμμή θα αναπαριστά το συντομότερο μονοπάτι.




## Αλγόριθμος 2

Αυτός ο αλγόριθμος είναι πιο απλός από τον προηγούμενο και έχει δύο μέρη. Το πρώτο είναι αυτό που υπολογίζει σε πόσα shortest paths μεσολαβεί κάποιος κόμβος και το δεύτερο η ταξινόμηση σε φθίνουσα σειρά.

Τον υπολογισμό τον κάνουμε χρησιμοποιώντας τον πίνακα Next ο οποίος δημιουργήθηκε όπως εξηγήσαμε παραπάνω.

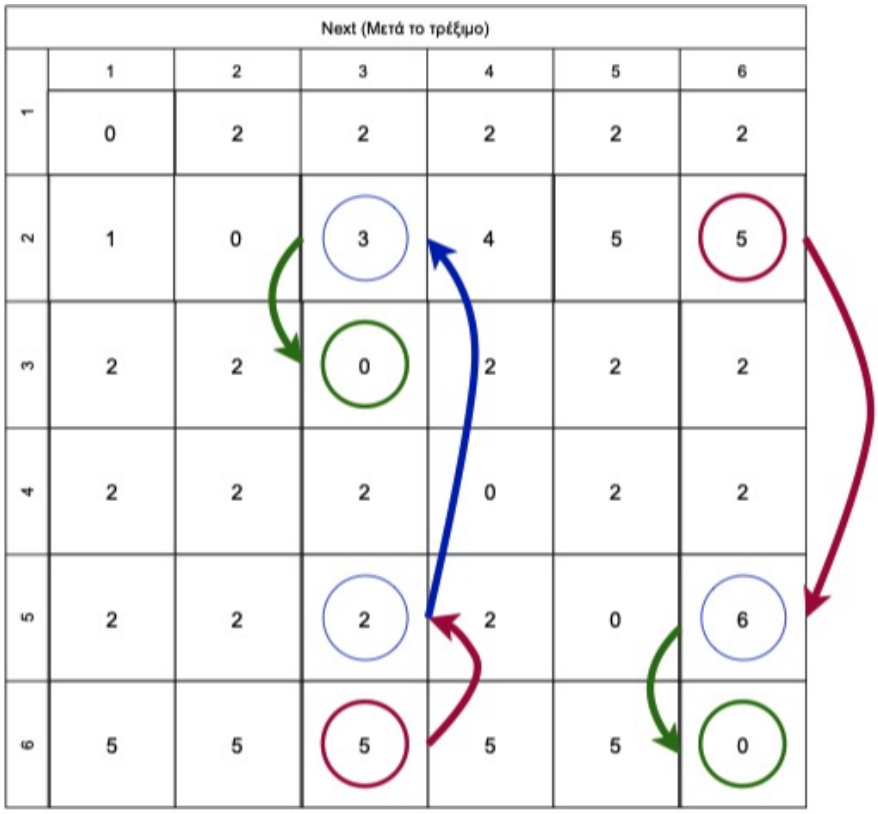
Δομές Δεδομένων:

- Ένας μονοδιάστατος πίνακας ακεραίων όπου θα αποθηκεύεται ο αριθμός εμφάνισης του κάθε κόμβο
- Ο πίνακας Next, βλέπε παραπάνω
- Node Count struct

**Υπολογισμός:**

Διαβάζουμε κάθε γραμμή του πίνακα, και μετά κοιτάμε στήλη στήλη για να βρούμε ποιοι κόμβοι μεσολαβούν στο μονοπάτι γραμμής στήλης, δηλαδή όταν κοιτάμε την i γραμμή και την j στήλη, θα βρούμε ποιοι κόμβοι μεσολαβούν στο μονοπάτι ανάμεσα στον κόμβο i και στον κόμβο j. Εννοείται με την χρήση ενός βοηθητικού πίνακα ακεραίων, με θέσεις όσοι είναι κόμβοι, αυξάνουμε τον μετρητή της εμφάνισης του κάθε κόμβου κάθε φορά που τον συναντάμε.

Παρακάτω φαίνεται στον πίνακα σαν παράδειγμα, πως θα διαβάζαμε ποιοι κόμβοι παρεμβάλλονται, για τις διαδρομές 6 προς 3 και 2 προς 6. Σε αυτό το παράδειγμα θα αυξήσουμε τον αριθμό εμφάνισης των κόμβων που είναι σε κύκλο. Άρα για να βρούμε το συντομότερο μονοπάτι απο το 6 στο 3 ακολουθούμε τα βελάκια τα οποία μας δείχνουν τους κόμβους που θα μεσολαβούν. Οπου το **κόκκινο** ειναι η αρχη, τα **μπλε** οι ενδιάμεσοι κόμβοι και το **πράσινο** ο τελικός κόμβος.




| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 2 | 2 | 2 | 2 |
| 2 | 1 | 0 | 3 | 4 | 5 | 5 |
| 3 | 2 | 2 | 0 | 2 | 2 | 2 |
| 4 | 2 | 2 | 2 | 0 | 2 | 2 |
| 5 | 2 | 2 | 2 | 2 | 0 | 6 |
| 6 | 5 | 5 | 5 | 5 | 5 | 0 |

Έτσι προκύπτει το παρακάτω Ranking:

```
The betweenness array is (bigger is better):
  1.)Node   2 is in  18
  2.)Node   5 is in   8
  3.)Node   1 is in   0
  3.)Node   3 is in   0
  3.)Node   4 is in   0
```

**Ψευδοκώδικας**:

```
1. ΑΡΧΙΚΟΠΟΙΗΣΕ ΠΙΝΑΚΑ ΑΚΕΡΑΙΩΝ εμφάνιση(αριθμός κόμβων) σε
ΜΗΔΕΝ
2. ΟΣΟ ΥΠΑΡΧΟΥΝ i ΓΡΑΜΜΕΣ στον Next
3.     ΟΣΟ ΥΠΑΡΧΟΥΝ j ΣΤΗΛΕΣ στον Next
4.           k ΔΕΙΚΤΗΣ σε ΚΕΛΙ Next(i,j):
5.           ΟΣΟ |k| != 0:
6.                 εμφάνιση(|k|)++
7.                 k ΔΕΙΚΤΗΣ σε κελί Next(|k|, j)
```

Η πολυπλοκότητα αυτού του αλγορίθμου είναι στις γραμμές:
1) O(|V|)
2->7) O((|V|^2)*(1 + |V|*2) = O(|V|^3)

Το |V|^2 προφανές, αλλά στην γραμμή 5, έχουμε επίσης V, γιατί στην χειρότερη περίπτωση θα σκανάρουμε όλην την στήλη. Στην πραγματικότητα θα είναι λιγότερο, γιατί είναι αδύνατο όλοι οι κόμβοι να απέχουν τόσο μεταξύ τους, αλλά δεν επηρεάζει το |V|^3.

Πιο αναλυτικά μπορούμε να πούμε ότι η μέση πολυπλοκότητα του αλγόριθμου αυτού δίνεται απο τον τύπο:

$$Complexity = n\frac{\sum_{i=0}^{n}(n-i)}{AvgConnections} = \frac{1}{2}n^2\frac{(n-1)}{AvgConnections}$$

Ύστερα φτιάχνουμε έναν πίνακα από node Count struct ταξινομημένα από το μεγαλύτερο πρός το μικρότερο με αλγόριθμο quicksort και πολυπλοκότητα $O(|V| * log(|V|))$.

Στον κώδικα που ακολουθεί δεν έχει γίνει βελτιστοποίηση για cache επειδή ο αλγόριθμος αυτός, παρόλο που έχει θεωρητικά την ίδια πολυπλοκότητα με τον Floyd Warshall, στην πράξη δεν συμβαίνει η χειρότερη περίπτωση, και είναι η πολυπλοκότητα πολύ μικρότερη οπότε δεν προκύπτει κάποιο θέμα.

## Αλγόριθμος 3

Κώδικας για τον υπολογισμό του μέσου μήκους των συντομότερων μονοπατιών από κάθε κόμβο προς όλους τους άλλους.

Αυτό το κάνουμε διαπερνώντας τον πίνακα που προέκυψε από τον Floyd Warshall, γραμμή γραμμή, προσθέτωντας το μέγεθος των μονοπατιών και ύστερα διαιρώντας τον αριθμό αυτών με το |V|.
Μετά τα ταξινομούμε αυτά σε μια αύξουσα σειρά.
Στον πραγματικό κώδικα καθώς δουλεύουμε με integers δεν διαιρούμε με τον αριθμό |V| γιατί θέλουμε την απλή αναλογία και δεν επηρεάζει το αποτέλεσμα.

Δομές Δεδομένων:

- Μονοδιάστατος πίνακας ακεραίων μεγέθους |V| για τους μέσους όρους
- Πίνακας που προκύπτει από τον Floyd Warshall
- Node Count struct

Ψευδοκώδικας:

```
1. ΑΡΧΙΚΟΠΟΙΗΣΕ ΠΙΝΑΚΑ αποστάσεις(|V|) σε ΜΗΔΕΝ
2. ΌΣΟ ΥΠΑΡΧΟΥΝ i ΓΡΑΜΜΕΣ floyd_warshall:
3.     ΟΣΟ ΥΠΑΡΧΟΥΝ j ΓΡΑΜΜΕΣ floyd_warshall:
4.         αποστάσεις(i) = floyd_warshall(i, j)
5.     αποστάσεις(i) = αποστάσεις(i)/|V|
```

Πολυπλοκότητα $O(|V|^2)$

Floyd-Warshall

| | 1 | 2 | 3 | 4 | 5 | 6 | Avg Distance |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 2 | 2 | 3 | =10/5 = 2.0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 2 | = 6/5 = 1.2 |
| 3 | 2 | 1 | 0 | 2 | 2 | 3 | =10/5 = 2.0 |
| 4 | 2 | 1 | 2 | 0 | 2 | 3 | =10/5 = 2.0 |
| 5 | 2 | 1 | 2 | 2 | 0 | 1 | = 8/5 = 1.6 |
| 6 | 3 | 2 | 3 | 3 | 1 | 0 | =12/5 = 2.4 |

Ύστερα φτιάχνουμε και έναν πίνακα από node Count struct και με τον quicksort, ο οποίος έχει πολυπλοκότητα O(|V|log(|V|)), τον ταξινομούμε σε αύξουσα σειρά.

## Αλγόριθμος 4

Ο αλγόριθμος αυτός χωρίζεται  σε δύο στάδια, το ένα είναι η δημιουργία δύο πινάκων, οι οποίοι θα έχουν τα στοιχεία στην μορφή που χρειάζεται για τον Kendall και το δεύτερο θα είναι η εκτέλεση του Kendall.

Δομές Δεδομένων:

- Μονοδιάστατοι πίνακες
- Node Count struct

**Πρώτο Στάδιο:**

Βάζουμε σε δύο μονοδιάστατους πίνακες, έναν που θα χρησιμεύει για τα ranks των κόμβων που εμφανίζονται μέσα στα συντομότερα μονοπάτια και έναν στα ranks των μέσων μεγεθών του κάθε μονοπατιού. Αυτο το κάνουμε για να μην εχουμε θέματα ισοτιμίας κόμβων.
Έτσι μπορούμε να έχουμε πρόσβαση στην θέση ταξινόμησης ενός κόμβου Χ καλόντας
`in_rankingsPtr->rankA[X]`
`in_rankingsPtr->rankB[X]`
Για τους ταξινομημένου πίνακες closeness και betweenness αντίστοιχα.
Θεωρούμε πως η κάθε θέση των πινάκων είναι ο αριθμός του κόμβου, και η κάθε τιμή της θέσης είναι το ranking του κόμβου. Αυτήν την αντιστοίχηση την κάνουμε με έναν αλγόριθμο που σκανάρει του πίνακες που δημιουργούνται στους αλγόριθμους 2 και 3.

**Ψευδοκώδικας:**

```
1.      ΔΕΣΜΕΥΣΗ ΜΝΗΜΗΣ πίνακα1(|V|)
2.      ΔΕΣΜΕΥΣΗ ΜΝΗΜΗΣ πίνακα2(|V|)
3.      curr_rankA = 1
4.      curr_rankB = 1
5.      πίνακας1(πίνακας_αλγο2(0).index) = 1
6.      πίνακας2(πίνακας_αλγο3(0).index) = 1
7.       i = 1
8.      ΟΣΟ i < |V|:
9.              ΑΝ πίνακας_αλγο2(i).count != πίνακας_αλγο2(i-1).count:
10.             curr_rankA ++
11.             ΑΝ πινακας_αλγο3(i).count != πίνακας_αλγο3(i-1).count:
12.             curr_rankB ++
13.             πίνακας1(πίνακας_αλγο2(i).index) = curr_rankA
14.             πίνακας2(πίνακας_αλγο3(i).index) = curr_rankB
```

Πολυπλοκότητα O(κ|V|) = O(|V|)

**Δεύτερο Στάδιο:**

Τώρα παίρνουμε τους δύο πίνακες που φτιάξαμε στο πρώτο στάδιο και τους χρησιμοποιούμε για τον Kendall βρίσκοντας το tauβ προσπαθώντας να λάβουμε υπόψη μας τις ισοβαθμίες κόμβων.

**Ψευδοκώδικας:**

(πίνακας1 και πίνακας2 είναι αυτοί που βρήκαμε από το πρώτο στάδιο)

```
1.      i = 1, j =1, numerator = 0, numpairs = 0, numties_x = 0, numties_y = 0
2.      ΟΣΟ i < |V|
3.              ΟΣΟ j < |V|
4.                      x_diff = πίνακας1(j) - πίνακας1(i)
5.                      y_diff = πίνακας2(j) - πίνακας2(i)
6.                      numerator = numerator + πρόσημο(x_diff)*πρόσημο(y_diff)
7.                      numpairs++
8.                      ΑΝ x_diff == 0
9.                              numties_x++
10.                     ΑΝ y_diff == 0
11.                             numties_y++
12.     kendal = numerator / sqrt((numpairs - numties_x)*(numpairs - numties_y))
```

Ο rankA, rankB περιέχει τα ranks xi, yi αντίστοιχα. Το πρώτο στοιχείο βρίσκεται στη θέση 1.
Ο αλγόριθμος τρέχει n-1 + n-2 + … + 2 + 1 φορές = (n-1)*(n-2)/2 φορές. Άρα έχουμε πολυπλοκότητα O(n^2).
Ξεκινάει από το ζευγάρι (i.i+1) έως το (i,size-1) οπότε θα έχουμε σαρώσει όλα τα πιθανά ζευγάρια τα οποία προσμετρούνται.
Παίρνει τη διαφορά ενός ζευγαριού (i1,j1), (i2,j2) και βρίσκει το πρόσημο για καθένα μέλος του ζευγαριού x,y.
Αν είναι θετικό το πρόσημο του πολλαπλασιασμού σημαίνει πως έχουμε concurrent pair οπότε προσθέτει 1 στον αριθμητή, αλλιώς αφαιρεί 1. Στην περίπτωση που κάποιο μέλος είναι 0, δεν προσμετράται στον αριθμητή, άλλα προστίθεται στον αριθμό των ίσων μελών, είτε από το μέλος x, είτε από το μέλος y, είτε και τα δύο, ανάλογα την περίπτωση.
Στο τέλος υπολογίζει τον Kendall με βάση τον τύπο nc-nd/squareroot( (n-n1)*(n-n2)) όπου n ο αριθμός όλων των πιθανών ζευγαριών, n1 και n2 ο αριθμός των ίσων ranks στους πίνακες rankA και rankB αντίστοιχα και nc, nd ο αριθμός των concurrent και discocurrent pairs. (Σε εμάς η μεταβλητή numerator.)

Γραμμές και απόδοση:
1 -> O(6)
2 μέχρι 11 -> $O(|V|^2*7) = O(|V|^2)$

| Όνομα Τεστ | Αριθμός Κόμβων | Kendall's Coefficient ( τ ) |
|---|---|---|
| Τεστ 1 | 35 | 0.573 |
| Τεστ 2 | 4941 | 0.202 |
| Τεστ 3 | 6927 | 0.399 |

Παρατηρούμε ότι ο κένταλ δεν εξαρτάται από τον αριθμό των κόμβων αλλα απο την τοπολογία τους. Άμα είναι οι δυό λίστες παρόμοια τότε θα πρέπει το Kendall να είναι κοντά στο 1. Άμα είναι αντιστρόφως ανάλογα θα είναι με συντελεστή -1. Αν είναι κοντά στο 0 σημαίνει ότι είναι τελείως διαφορετικές οι λίστες χωρίς κάποια σχέση μεταξύ των δύο λύσεων.

```
The betweenness array is (bigger is better):
 1.)Node   1 is in 528
 2.)Node   3 is in 334
 3.)Node  33 is in 292
 4.)Node  34 is in 126
 5.)Node  32 is in  72
 6.)Node   6 is in  60
 7.)Node  28 is in  48
 8.)Node   9 is in  44
 9.)Node  24 is in  30
10.)Node   2 is in  30
The closeness array is (total distance)(smaller is better):
 1.)Node   1 is in  58
 2.)Node   3 is in  59
 3.)Node  34 is in  61
 4.)Node  32 is in  61
 5.)Node  33 is in  64
 6.)Node   9 is in  64
 7.)Node  14 is in  65
 8.)Node  20 is in  67
 9.)Node   2 is in  68
10.)Node   4 is in  71
Total number of vertices: 34
Ranking of vector 1 in S.P. 1, of Distance 1
Ranking of vector 2 in S.P. 9, of Distance 7
Ranking of vector 3 in S.P. 2, of Distance 2
Ranking of vector 4 in S.P. 14, of Distance 8
Ranking of vector 5 in S.P. 14, of Distance 16
Ranking of vector 6 in S.P. 6, of Distance 15
Ranking of vector 7 in S.P. 13, of Distance 15
Ranking of vector 8 in S.P. 14, of Distance 12
Ranking of vector 9 in S.P. 8, of Distance 4

Kendall coefficient is: 0.573

Time taken is 0.00 minutes / 0.000711 seconds
```

**Output of test 1**

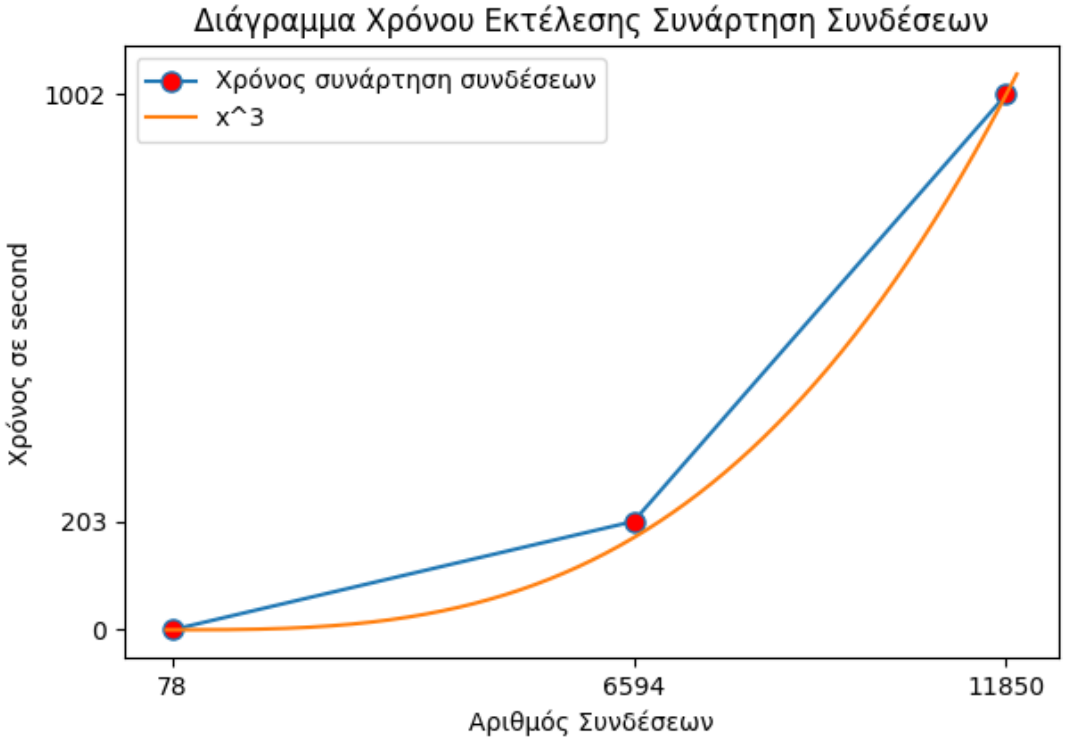
```
The betweenness array is (bigger is better):
 1.)Node 651 is in 6920928
 2.)Node 559 is in 6803696
 3.)Node 1365 is in 6786260
 4.)Node 2824 is in 6717278
 5.)Node 1324 is in 6642510
 6.)Node 2685 is in 6497662
 7.)Node 433 is in 5113348
 8.)Node 1377 is in 5059912
 9.)Node 1213 is in 5002920
10.)Node 1380 is in 4897320
The closeness array is (total distance)(smaller is better):
 1.)Node 1378 is in 60374
 2.)Node 1678 is in 61030
 3.)Node 2944 is in 61475
 4.)Node 1377 is in 61849
 5.)Node 2781 is in 62040
 6.)Node 1365 is in 62274
 7.)Node 1368 is in 62385
 8.)Node 1380 is in 62441
 9.)Node 2685 is in 62712
10.)Node 2795 is in 63308
Total number of vertices: 4941
Ranking of vector 1 in S.P. 817, of Distance 214
Ranking of vector 2 in S.P. 1172, of Distance 390
Ranking of vector 3 in S.P. 104, of Distance 75
Ranking of vector 4 in S.P. 1249, of Distance 571
Ranking of vector 5 in S.P. 576, of Distance 3585
Ranking of vector 6 in S.P. 907, of Distance 3834
Ranking of vector 7 in S.P. 522, of Distance 3222
Ranking of vector 8 in S.P. 2464, of Distance 3222
Ranking of vector 9 in S.P. 1239, of Distance 3833

Kendall coefficient is: 0.202

Time taken is 3.38 minutes / 203.096087 seconds
```

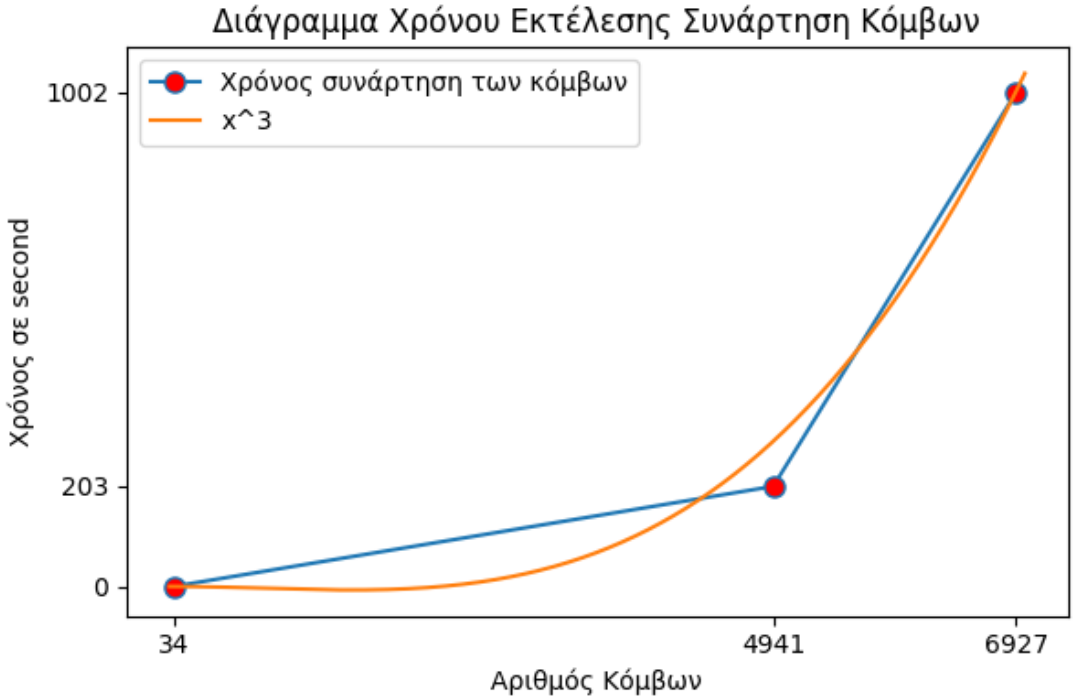
**Output of test 3**

```
The betweenness array is (bigger is better):
  1.)Node 6927 is in 32649134
  2.)Node  186 is in 3157434
  3.)Node   10 is in 2982098
  4.)Node   86 is in 1487364
  5.)Node  417 is in 1401312
  6.)Node  164 is in 1392230
  7.)Node   44 is in 1278582
  8.)Node   78 is in 1168702
  9.)Node  333 is in 1051952
 10.)Node  249 is in 1011862
The closeness array is (total distance)(smaller is better):
  1.)Node 6927 is in 13345
  2.)Node 164 is in 18547
  3.)Node  10 is in 18570
  4.)Node 186 is in 18789
  5.)Node 374 is in 18840
  6.)Node 436 is in 18940
  7.)Node  78 is in 18940
  8.)Node  44 is in 18948
  9.)Node 475 is in 19012
 10.)Node 147 is in 19035
Total number of vertices: 6927
Ranking of vector 1 in S.P. 207, of Distance 184
Ranking of vector 2 in S.P. 12, of Distance 146
Ranking of vector 3 in S.P. 379, of Distance 252
Ranking of vector 4 in S.P. 77, of Distance 65
Ranking of vector 5 in S.P. 181, of Distance 117
Ranking of vector 6 in S.P. 104, of Distance 62
Ranking of vector 7 in S.P. 408, of Distance 298
Ranking of vector 8 in S.P. 144, of Distance 87
Ranking of vector 9 in S.P. 344, of Distance 225

Kendall coefficient is: 0.399

Time taken is 16.70 minutes / 1002.218258 seconds
```

**Output of Test 3.**

**Διαγράμματα Χρόνου συνάρτηση συνδέσεων και κόμβων**


Διάγραμμα Χρόνου Εκτέλεσης Συνάρτηση Συνδέσεων
Χρόνος συνάρτηση συνδέσεων
x^3
1002
203
0
Χρόνος σε second
78
6594
11850
Αριθμός Συνδέσεων


Διάγραμμα Χρόνου Εκτέλεσης Συνάρτηση Κόμβων
Χρόνος συνάρτηση των κόμβων
x^3
1002
203
0
Χρόνος σε second
34
4941
6927
Αριθμός Κόμβων

Ο χρόνος των παραπάνω διαγραμμάτων φαίνεται ότι ακολουθεί την θεωρία μας οτι η πολυπλοκότητα του αλγορίθμου μας είναι $O(n^3)$.

## Απάντηση στο πρακτικό ΕΡΏΤΗΜΑ

Για να δοθεί απάντηση σε αυτό το πρόβλημα πρέπει πρώτα να αναφερθεί με ποιον τρόπο οι αλγόριθμοι 2 και 3 συνεισφέρουν στην λύση.

### Αλγόριθμοι 2 και 3

Όσον αφορά τον αλγόριθμο 2, ο οποίος χρησιμεύει στην εύρεση των φορών εμφάνισης ενός κόμβους στα συντομότερα μονοπάτια σύνδεσης άλλων αλγορίθμων, μας είναι χρήσιμος στο να βρούμε ποιοι είναι ο καλύτεροι κόμβοι για τοποθέτηση κέντρων υποκλοπών. Αυτό συμβαίνει καθώς σε όσο πιο πολλά μονοπάτια παρεμβαίνει ένας κόμβος, τόσο περισσότερες υποκλοπές θα έχουμε.

Όσον αφορά τον αλγόριθμο 3, ο οποίος χρησιμεύει στην εύρεση του μικρότερου μέσου όρου μήκους συντομότερων μονοπατιών σύνδεσης ενός κόμβου με όλους τους υπόλοιπους, μας είναι χρήσιμος αφού το κέντρο πρέπει να τοποθετηθεί σε κόμβο που έχει τα μικρότερα μονοπάτια προς αλλού. Αυτό συμβαίνει γιατί το κέντρο επικοινωνίας πρέπει να έχει την ιδιότητα να μπορεί να επικοινωνήσει όσο γρηγορότερα με τους υπόλοιπους κόμβους.

### Πρώτη σκέψη

Είναι λογικό να υποθέσει κανείς πως σε όσα περισσότερα μονοπάτια παρεμβάλλεται ένας κόμβος τόσο πιο πιθανό είναι να είναι κοντά σε πολλούς άλλους κόμβους. Παρόλο που φαίνεται να βγάζει νόημα, και μάλιστα μπορούμε να το φανταστούμε εύκολα σε ένα γράφημα με λίγους κόμβους, σε πολύ μεγάλα γραφήματα δεν μπορούμε διαισθητικά να κάνουμε υποθέσεις. Για αυτό μπορούμε να χρησιμοποιήσουμε το Kendall το οποίο μας δίνει μια μαθηματική λύση σε αυτό το πρόβλημα.

### Kendall

Ο Kendal μπορεί να ελέγξει πόσο όμοιες είναι δύο ακολουθίες αριθμών, δίνοντας μας 1 αν είναι ίδιες, 0 αν δεν υπάρχει τίποτα κοινό και -1 αν είναι αντίθετες. Αυτό μας φαίνεται χρήσιμο καθώς μπορούμε να το χρησιμοποιήσουμε για να συγκρίνουμε δύο ταξινομημένες ακολουθίες, η μία που να προκύπτει από τον αλγόριθμο 2 και η άλλην από τον 3, και να δούμε αν είναι όμοιες.

### Σύγκριση

Αν αυτές οι ακολουθίες είναι σε καλό βαθμό όμοιες, αυτό μας αφήνει να συμπεράνουμε πως η βέλτιστη λύση στο ένα πρόβλημα, δηλαδή οι πρώτοι κόμβοι της ταξινομημένης ακολουθίας, θα είναι μια σχετικά καλή λύση στο άλλο, αφού πάλι θα βρίσκονται στην αρχή της άλλης ταξινομημένης ακολουθίας. Οπότε χρησιμοποιούμε το Kendall και τα αποτελέσματα είναι ανεπαρκή για να βγάλουμε συμπεράσματα. Αφού όπως φαίνεται και στον πίνακα, το test 1 και 3 έχουν αποτελέσματα που μας δείχνουν πως ίσως το ερώτημα απαντάται θετικά, ενώ το test 2 μας δείχνει πως δεν υπάρχει μεγάλη σχέση.

## Παρατηρήσεις

Κοιτάζοντας τα τεστ που τρέξαμε αλλά και τα δικά μας τεστ, παρατηρούμε ότι αρκετά συχνα οι πρώτοι κόμβοι είναι παρόμοιοι μεταξύ τους. Αρα για το τεστ 1 και 3 η λύση των δύο προβλήματων είναι ίδια για έναν αριθμό σταθμών αφού έχουν τον ίδιο κόμβο στην πρώτη θέση άρα είναι και η βέλτιστη. Στο δύο ενώ οι πρώτες θέσεις είναι διαφορετικά ταξινομημένες οι περισσότεροι κόμβοι βρίσκονται και στις δύο δεκάδες άρα μπορεί να θεωρηθεί και εκεί σχετικά καλή λύση αν επιλέξουμε έναν ή λίγους από του πρώτους κόμβους να βάλουμε τα κέντρα μας.

## Συμπέρασμα

Για να μπορέσουμε να κρίνουμε άμα τα δυό προβλήματα έχουν παρόμοια λύση πρέπει να δούμε εάν μοιαζουν οι πίνακες R κρυφ  και  R διοικ. Επιπλέον πρέπει να έχουμε στο νου μας εάν τα σχετικά κοντά σε ranks κόμβοι έχουν μεγάλη διαφορά. Τέλος παίζει και ρόλο πόσους σταθμούς έχουμε για να κρυφακούμε και να διοικούμε χρειαζόμαστε.
Εάν ο αριθμός των σταθμών αυτών είναι μικρός τότε μπορούμε να εμπιστευτούμε και άλλα στοιχεία εκτός από τον Kendall, αλλά αν είναι μεγάλος δεν μπορούμε να ξέρουμε. Επομένως το αποτέλεσμα του ερωτήματος είναι ασαφές.