# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
## Τμήμα Πληροφορικής

## ΕΠΛ 232 – Προγραμματιστικές Τεχνικές και Εργαλεία

## ΑΣΚΗΣΗ 2 – Υλοποίηση Αλγορίθμου Μεταβατικής Κλειστότητας με χρήση Δυναμικής Δέσμευσης Μνήμης

**Διδάσκων: Ανδρέας Αριστείδου**
**Υπεύθυνοι Εργαστηρίων: Παύλος Αντωνίου & Πύρρος Μπράτσκας**

Ημερομηνία Ανάθεσης: Δευτέρα, 16 Οκτωβρίου 2023
Ημερομηνία Παράδοσης: Παρασκευή, 3 Νοεμβρίου 2023, Ώρα 13:00

(ο κώδικας να υποβληθεί μέσω του Moodle)

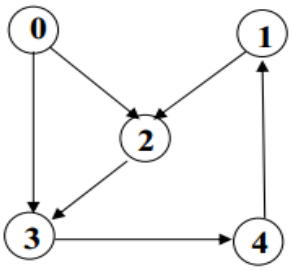
### I. Στόχος

Στην άσκηση αυτή θα ασχοληθούμε σε περισσότερο βάθος με τη γλώσσα προγραμματισμού C. Η άσκηση απαιτεί μεταξύ άλλων χρήση συμβολοσειρών, συναρτήσεων, αρχείων, δεικτών και δυναμική δέσμευση μνήμης. Επιτρέπεται η δημιουργία ενός στατικού μονοδιάστατου πίνακα μόνο ΑΝ θα κάνετε ανάγνωση δεδομένων από το αρχείο γραμμή προς γραμμή με χρήση fgets (δεν είναι αναγκαίο ούτε υποχρεωτικό). Όλοι οι υπόλοιποι πίνακες του προγράμματος πρέπει να δημιουργούνται δυναμικά και όταν δεν χρειάζονται να απελευθερώνεται η δεσμευμένη μνήμη. Συνίσταται επίσης να αποφευχθεί η χρήση global μεταβλητών/δεικτών οποιουδήποτε τύπου.

### II. Υλοποίηση Αλγόριθμου Μεταβατικής Κλειστότητας

#### A. Εισαγωγή

Το **Οδικό Δίκτυο (G, γράφος)** μιας περιοχής μπορεί να απεικονιστεί από ένα σύνολο ***κόμβων (V, τις πόλεις)*** και ένα σύνολο ***ακμών*** *(**E,** τις οδικές διασυνδέσεις* μεταξύ των πόλεων), όπως φαίνεται διαγραμματικά στο πιο κάτω παράδειγμα:




**Σχήμα 1: Το Οδικό Δίκτυο (G) μιας περιοχής απεικονίζεται από Κορυφές (V) και Ακμές (E).**

Σε ένα πρόγραμμα, ένας γράφος αναπαριστάται από ένα **πίνακα γειτνίασης A** δυο διαστάσεων, όπου το στοιχείο A[i][j] του πίνακα έχει τη τιμή 1 αν υπάρχει *ακμή (απευθείας σύνδεση)* από την πόλη i στην πόλη j ή 0 αν δεν υπάρχει απευθείας σύνδεση. Μια σύνδεση δεν είναι απαραιτήτως δυο κατευθύνσεων, δηλαδή αν `A[i][j]=1` δεν σημαίνει ότι και το `A[j][i]=1`. Για παράδειγμα, ο πίνακας γειτνίασης του οδικού δικτύου που απεικονίζεται στο σχήμα 1 φαίνεται πιο κάτω:

| | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0 | 0 | **1** | **1** | 0 |
| **1** | 0 | 0 | **1** | 0 | 0 |
| **2** | 0 | 0 | 0 | **1** | 0 |
| **3** | 0 | 0 | 0 | 0 | **1** |
| **4** | 0 | **1** | 0 | 0 | 0 |

**Σχήμα 2: Πίνακας Γειτνίασης (A) του γράφου που απεικονίζεται στο Σχήμα 1.**

Εκτός από τον πίνακα γειτνίασης, ένας γράφος μπορεί να αναπαρασταθεί σε ένα πρόγραμμα και από μια **Λίστα Γειτνίασης (R),** καταγράφοντας μόνο τις απευθείας συνδέσεις (ακμές). Η λίστα γειτνίασης ενός γράφου **Ν κορυφών** έχει μέγιστο μέγεθος **$N^2$ – Ν** θέσεις, εφόσον δεν καταγράφονται οι αυτό-αναφορικές συνδέσεις (A[i][i]). **Ο αριθμός Ν και ο πίνακας γειτνίασης θα διαβάζεται από το αρχείο εισόδου.** Παράδειγμα της Λίστας Γειτνίασης (R) του παραδείγματος μας φαίνεται στο σχήμα 3.

| **Αφετηρία** | **Προορισμός** |
|---|---|
| **0** | **2** |
| **0** | **3** |
| **1** | **2** |
| **2** | **3** |
| **3** | **4** |
| **4** | **1** |

**Σχήμα 3: Λίστα Γειτνίασης (R) του γράφου που απεικονίζεται στο Σχήμα 1.**

Το αρχείο που περιέχει τα δεδομένα του πιο πάνω γράφου έχει την πιο κάτω μορφή:

```
5
0 0 1 1 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 1 0 0 0
```

Στην πρώτη γραμμή του αρχείου υπάρχει μόνο ο αριθμός των κόμβων N (5 στο εν λόγω παράδειγμα), θα ακολουθεί αλλαγή γραμμής και στη συνέχεια ο πίνακας γειτνίασης. Κάθε γραμμή του πίνακα γειτνίασης περιέχει μόνο τα ψηφία 0 και 1 με ένα κενό αναμεταξύ τους και στο τέλος ακολουθεί αλλαγή γραμμής. Υποθέστε ότι τα αρχεία εισόδου έχουν πάντα τη σωστή μορφή και δεν θα πρέπει να κάνετε κανένα έλεγχο ορθότητας για τα περιεχόμενα των αρχείων αυτών.
Σημείωση: Η αλλαγή γραμμής στα αρχεία που δημιουργήθηκαν σε περιβάλλον UNIX είναι \n ενώ στα αρχεία που δημιουργήθηκαν σε περιβάλλον Windows είναι \r\n. Συστήνεται όπως τα προγράμματά σας λαμβάνουν υπόψη τα πιο πάνω για να δουλεύουν είτε με αρχεία UNIX ή Windows (βασικά το \r αν διαβάζεται να αγνοείται).

### Β. Το Πρόβλημα

**Ο στόχος της παρούσας άσκησης είναι, δεδομένων δύο πόλεων `i` (αφετηρία) και `j` (προορισμός) να βρείτε κατά πόσο υπάρχει σύνδεση (απευθείας ή όχι) από την πόλη i προς την πόλη `j`. Δεν μας ενδιαφέρει αν η σύνδεση που θα βρεθεί από το πρόγραμμα είναι η συντομότερη.**

Η εύρεση των απευθείας συνδέσεων είναι προφανώς ένα εύκολο πρόβλημα, η λύση του οποίου προκύπτει απευθείας από τον Πίνακα ή την Λίστα Γειτνίασης. Το πρόβλημα που παραμένει, είναι πώς να βρεθούν οι έμμεσες συνδέσεις μεταξύ 2 πόλεων. Για παράδειγμα, ενώ υπάρχει έμμεση

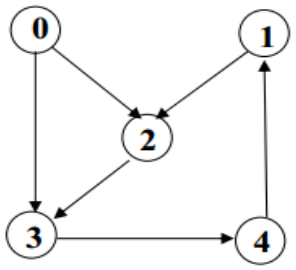
σύνδεση μεταξύ του 0 και 1 (π.χ., δείτε το 0→3→4→1 ή το 0→2→3→4→1 στο Σχήμα 1), αυτή η σύνδεση δεν φαίνεται άμεσα από την Λίστα Γειτνίασης.

Το πρόβλημα που έχετε να λύσετε ανάγεται στην ουσία στο πρόβλημα εύρεσης της μεταβατικής κλειστότητας (**R*,** transitive closure) της λίστας γειτνίασης **R.** Ειδικότερα, η μεταβατική κλειστότητα μπορεί να υπολογιστεί από τον ακόλουθο τετριμμένο ψευδοκώδικα (σημειώστε ότι στον ψευδοκώδικα φαίνεται απλά η λογική της λύσης και όχι η ίδια η λύση η οποία ενδέχεται να χρειάζεται περαιτέρω δομές και μεταβλητές):

**Ψευδοκώδικας Υπολογισμού Μεταβατικής Κλειστότητας**

```
R* = R      // Αρχικοποίησε το R* με το R
Επανέλαβε {
     Για κάθε ακμή (u,v) που ανήκει στο R* {
          Εάν υπάρχει ακμή (v,w) που ανήκει στο R {
               Πρόσθεσε το (u,w) στο R* δεδομένου ότι:
                α)(u,w) δεν υπάρχει ήδη στο R*, και
                β) u != w // απαλοιφή αυτοαναφορικών συνδέσεων
          }
     }
} Όσο το R* μεταβάλλεται σε μια επανάληψη
```

Το αποτέλεσμα υπολογισμού της μεταβατικής κλειστότητας του παραδείγματος μας συνοψίζεται στα ακόλουθα σχήματα. Σημειώστε ότι σε κάθε επανάληψη δεν είναι αναγκαίο να διέλθετε του πίνακα R* από την αρχή, παρά μόνο χρειάζεται να εξετάσετε τα στοιχεία που έχουν προστεθεί στην προηγούμενη επανάληψη.




**Γράφος G**

| Αφετηρία | Προορισμός |
|---|---|
| 0 | 2 |
| 0 | 3 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 1 |

**Λίστα Γειτνίασης R**

| Αφετηρία | Προορισμός | |
|---|---|---|
| 0 | 2 | Αρχικοποίηση |
| 0 | 3 | |
| 1 | 2 | |
| 2 | 3 | |
| 3 | 4 | |
| 4 | 1 | |
| 0 | 4 | Επανάληψη Α' |
| 1 | 3 | |
| 2 | 4 | |
| 3 | 1 | |
| 4 | 2 | |
| 0 | 1 | Επανάληψη Β' |
| 1 | 4 | |
| 2 | 1 | |
| 3 | 2 | |
| 4 | 3 | |

**Μεταβατική Κλειστότητα R***

### Γ. Ζητουμένο

Ζητούμενο αυτής της άσκησης είναι η κατασκευή ενός προγράμματος, `cityLink.c`, στη γλώσσα προγραμματισμού C, το οποίο θα λαμβάνει σαν είσοδο (α) τον πίνακα διασυνδέσεων για κάποιο συγκεκριμένο σύνολο πόλεων και (β) τους αριθμούς δυο πόλεων έστω `i` και `j`, και να εκτυπώνει (α)

το κατά πόσο υπάρχει σύνδεση (απευθείας η όχι) από την πόλη `i` (αφετηρία) προς την πόλη `j` (προορισμός) και (β) ολόκληρη την λίστα της μεταβατική κλειστότητα του R.

Το πρόγραμμα θα δέχεται 4 επιλογές από τη γραμμή εντολών (command line arguments) όπου μόνο η πρώτη είναι υποχρεωτική και οι άλλες προαιρετικές:

- `-i <filename>`: καθορίζει το όνομα αρχείου που θα περιέχει
- `-r <source_city>,<destination_city>`: καθορίζει την πόλη αφετηρίας και την πόλη προορισμού.
- `-p`: καθορίζει ότι επιθυμούμε να δούμε ολόκληρη την λίστα μεταβατικής κλειστότητας R* μετά την ολοκλήρωσή της στην οθόνη.
- `-o`: καθορίζει ότι επιθυμούμε να δούμε ολόκληρη την λίστα μεταβατικής κλειστότητας R* μετά την ολοκλήρωσή της σε αρχείο με όνομα `out-<filename>.txt`.

όπου το `filename` είναι το όνομα του αρχείου που θα περιέχει τον πίνακα διασυνδέσεων, και `source_city` και `destination_city` θα είναι ακέραιοι αριθμοί (αφετηρίας και προορισμού αντίστοιχα). Σημειώστε ότι ο χρήστης μπορεί να δώσει <u>όποια/ες επιλογή/ες θέλει</u> μέχρι και όλες τις επιλογές σε μια κλήση του προγράμματος <u>με οποιαδήποτε σειρά</u>. Για να γίνει αυτό με εύκολο τρόπο διαβάστε προσεκτικά την επόμενη παράγραφο και τους προτεινόμενους συνδέσμους.

Για την επεξεργασία των επιλογών του προγράμματος (command line arguments) σας συνιστούμε ανεπιφύλακτα να κάνετε χρήση της βιβλιοθήκης `getopt`. Για περισσότερες πληροφορίες "`man 3 getopt`", όπου 3 δηλώνει "C library routines". Για παράδειγμα χρήσης μπορείτε να δείτε εδώ. Θα χρειαστεί να ενσωματώσετε τη βιβλιοθήκη `getopt.h`.

Τέλος, το πρόγραμμά σας θα πρέπει να ελέγχει τις επιλογές του χρήστη. Σε περίπτωση που οποιοσδήποτε έλεγχος αποτύχει, το πρόγραμμα να τερματίζει και να παρουσιάζει ανάλογο μήνυμα.

Λάβετε υπόψη τα πιο κάτω:

- Δεν επιτρέπεται να μη δοθεί καμμιά επιλογή στα command line arguments
- Αν δοθεί μη επιτρεπτή επιλογή (π.χ. `-b`), μετά τα μηνύματα λάθους το πρόγραμμα να τερματίζεται.
- Η εντολή `-i` είναι υποχρεωτική και πρέπει να ακολουθεί το όνομα του αρχείου εισόδου δεδομένων. Η ανάγκη για να ακολουθεί το όνομα αρχείου καλύπτεται από την `getopt`.
- Αν δοθεί η επιλογή `-i <filename>` το πρόγραμμα να διαβάζει και να τυπώνει τον πίνακα γειτνίασης.
- Αν το αρχείο δεν υπάρχει να τυπώνεται το μήνυμα `Input file cannot be read!` και να τερματίζεται το πρόγραμμα.
- Η εντολή `-r` δεν είναι υποχρεωτική αλλά να δοθεί πρέπει να ακολουθεί η (χωρίς κενά) συμβολοσειρά `<source_city>,<destination_city>` όπου πρέπει να αποτελείται από δύο ακέραιους χωρισμένους με κόμμα. Η ανάγκη για να ακολουθεί η συμβολοσειρά καλύπτεται από την getopt.
- Αν η σύνδεση μεταξύ source_city και destination_city υπάρχει, τότε να τυπώνεται το μήνυμα `"Yes Path Exists!"` και το μονοπάτι μεταξύ των δυο πόλεων. Στο πρώτο παράδειγμα της διαδρομής μεταξύ (0,1), θα θέλαμε να μας τύπωνε στην οθόνη είτε το 0=>3=>4=>1 ή 0=>2=>3=>4=>1 (δεν χρειάζεται να δώσετε το βραχύτερο μονοπάτι)
- Αν η σύνδεση μεταξύ source_city και destination_city δεν υπάρχει, τότε να τυπώνεται το μήνυμα `"No Path Exists!"`
- Η εντολή `-p` δεν είναι υποχρεωτική
- Η εντολή `-o` δεν είναι υποχρεωτική. Όταν δοθεί, να τυπώνει τη λίστα μεταβατικής κλειστότητας R* μετά την ολοκλήρωσή της σε αρχείο με όνομα `out-<filename>.txt` και στην οθόνη το μήνυμα `Saving out-<filename>.txt...` όπου <filename> είναι το όνομα του αρχείου εισόδου.

Για τα μηνύματα λάθους, δείτε και τα πιο κάτω παραδείγματα εκτέλεσης (με κόκκινο η έξοδος του προγράμματος). Παραδείγματα αρχείων εισόδου δίνονται στο αρχείο as2-supplementary.zip, κάτω από το σύνδεσμο της Εργασίας 2 στο Moodle.

### Δ. Παραδείγματα Εκτέλεσης

```
./cityLink
No command line arguments given!
Usage: <executable> -i <inputfile> [-r <source >,<destination> -p -o]

./cityLink -b
./cityLink: invalid option -- 'b'
Usage: <executable> -i <inputfile> [-r <source >,<destination> -p -o]

./cityLink -i
./cityLink: option requires an argument -- 'i'
Usage: <executable> -i <inputfile> [-r <source >,<destination> -p -o]

./cityLink -r 0,1
No input file given!
Usage: <executable> -i <inputfile> [-r <source >,<destination> -p -o]

./cityLink -i cities.txt
Input file cannot be read!

./cityLink -i cities1.txt
Neighbor table
0 0 1 1 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 1 0 0 0

./cityLink -i cities1.txt -p
Neighbor table
0 0 1 1 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 1 0 0 0

R* table
0 -> 2
0 -> 3
```

```
1 -> 2
2 -> 3
3 -> 4
4 -> 1
0 -> 4
1 -> 3
2 -> 4
3 -> 1
4 -> 2
0 -> 1
1 -> 4
2 -> 1
3 -> 2
4 -> 3
```

```
./cityLink -i cities1.txt -r 0,1
Yes Path Exists!
0=>2=>3=>4=>1
```

```
./cityLink -o cities1.txt -r 4,0
No Path Exists!
```

```
./cityLink -i cities1.txt -opr 0,1
Neighbor table
0 0 1 1 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 1 0 0 0

R* table
0 -> 2
0 -> 3
1 -> 2
2 -> 3
3 -> 4
4 -> 1
0 -> 4
1 -> 3
```

Οι επιλογές μπορούσαν να δοθούν με διάφορους τρόπους όπως για παράδειγμα:

```
-o -p -r 0,1

-r 0,1 -op

-r 0,1 -po

...
```

```
2 -> 4
3 -> 1
4 -> 2
0 -> 1
1 -> 4
2 -> 1
3 -> 2
4 -> 3
Yes Path Exists!
0 => 2 => 3 => 4 => 1
Saving out-cities1.txt...
```

## III. Γενικές Οδηγίες

Τα προγράμματα σας θα πρέπει να συμβαδίζουν με το πρότυπο ANSI C, να περιλαμβάνουν εύστοχα και περιεκτικά σχόλια, να έχουν καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. **Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης doxygen** έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση συναρτήσεων και άλλων τεχνικών δομημένου προγραμματισμού που διδαχτήκατε στο ΕΠΛ131. Επίσης, σας θυμίζουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους. Τέλος το πρόγραμμα σας θα πρέπει να μεταγλωττίζεται στις μηχανές του εργαστηρίου. **Παρακαλώ όπως μελετηθούν ξανά οι κανόνες υποβολής εργασιών όπως αυτοί ορίζονται στο συμβόλαιο του μαθήματος.**

Επίσης να ακολουθήσετε τα πιο κάτω βήματα πριν και κατά την υποβολή της άσκησης σας στο Moodle:

1. Βεβαιωθείτε ότι βάλατε τα πρότυπα των συναρτήσεων.
2. Βεβαιωθείτε ότι βάλατε σχόλια υπό μορφή doxygen τόσο στην αρχή του αρχείου .c όσο και πάνω από κάθε πρότυπο συνάρτησης. Μπορείτε να βάλετε επιπλέον σχόλια μέσα στις συναρτήσεις.
3. Ανεβάστε τα 3 αρχεία cityLink.c, lab2.conf, README.dox ένα προς ένα **(ΜΗΝ υποβάλετε zip file)**

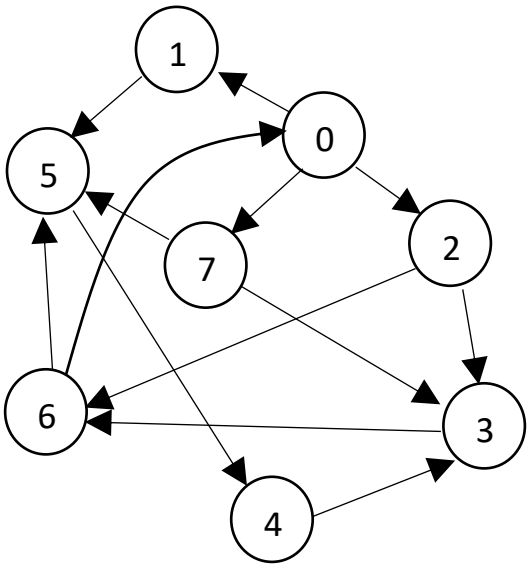
## IV. Κριτήρια αξιολόγησης

| | |
|---|---|
| Ανάγνωση αρχείου εισόδου και εκτύπωση πίνακα γειτνίασης | 25 |
| Υπολογισμός και εκτύπωση λίστας μεταβατικής κλειστότητας | 30 |
| Εύρεση και εκτύπωση διαδρομής μεταξύ αφετηρίας και προορισμού | 25 |
| Εγγραφή λίστας μεταβατικής κλειστότητας σε αρχείο | 10 |
| Γενική εικόνα, στοιχισμένος και ευανάγνωστος κώδικας, εύστοχα ονόματα μεταβλητών, σταθερών και συναρτήσεων, σχολιασμός με doxygen). | 10 |
| **ΣΥΝΟΛΟ** | **100** |

**Καλή Επιτυχία!**

## Παράρτημα

## V. Επιπλέον Παράδειγμα Εκτέλεσης (cities3.txt)




```
8
0 1 1 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 0 0 1 0 1 0 0
```

```
./cityLink -i cities3.txt -r 2,5
Neighbor table
0 1 1 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0
```

```
0 0 0 1 0 1 0 0
Yes Path Exists!
2 => 3 => 6 => 5

./cityLink -i cities3.txt -pr 3,7
Neighbor table
0 1 1 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 0 0 1 0 1 0 0

R* table
0 -> 1
0 -> 2
0 -> 7
1 -> 5
2 -> 3
2 -> 6
3 -> 6
4 -> 3
5 -> 4
6 -> 0
6 -> 5
7 -> 3
7 -> 5
0 -> 5
0 -> 3
0 -> 6
1 -> 4
2 -> 0
2 -> 5
3 -> 0
3 -> 5
4 -> 6
5 -> 3
```

```
6 -> 1
6 -> 2
6 -> 7
6 -> 4
7 -> 6
7 -> 4
0 -> 4
1 -> 3
2 -> 1
2 -> 7
2 -> 4
3 -> 1
3 -> 2
3 -> 7
3 -> 4
4 -> 0
4 -> 5
5 -> 6
6 -> 3
7 -> 0
1 -> 6
4 -> 1
4 -> 2
4 -> 7
5 -> 0
7 -> 1
7 -> 2
1 -> 0
5 -> 1
5 -> 2
5 -> 7
1 -> 2
1 -> 7
Yes Path Exists!
3 => 6 => 0 => 7
```

```
./cityLink -i cities3.txt -or 7,6
Neighbor table
0 1 1 0 0 0 0 1
```

```
0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 0 0 1 0 1 0 0
Yes Path Exists!
7 => 3 => 6
Saving out-cities3.txt...

./cityLink -i cities3.txt -or 7,2
Neighbor table
0 1 1 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 0 0 1 0 1 0 0
Yes Path Exists!
7 => 3 => 6 => 0 => 2
Saving out-cities3.txt...
```