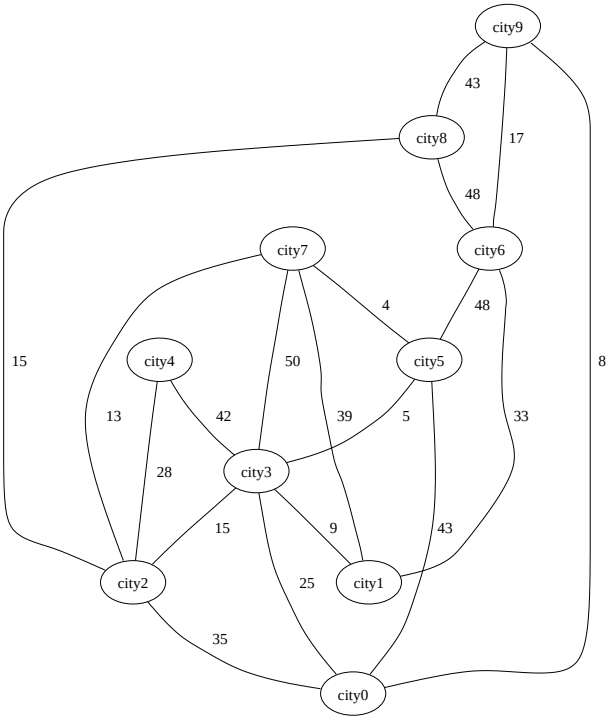
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2018-19)

Εργασία 3

Έστω ότι μας δίνεται ένα σύνολο από πόλεις και τα μήκη των δρόμων που συνδέουν κάποια ζευγάρια από αυτές, όπως φαίνεται στο διπλανό σχήμα. Μιλώντας με μαθηματικούς όρους, το σχήμα αυτό είναι ένας γράφος ή γράφημα (*graph*), όπου οι πόλεις είναι οι κόμβοι (*nodes*) του γράφου και οι δρόμοι είναι οι ακμές (*edges*) του. Στις ακμές του γράφου, μπορούμε να έχουμε και κάποιο κόστος (*cost*), εδώ το μήκος του κάθε δρόμου. Ένα κλασικό πρόβλημα στην Επιστήμη των Υπολογιστών, και ειδικότερα στη Θεωρητική Πληροφορική, είναι να βρεθεί μία συντομότερη διαδρομή (*shortest path*) μεταξύ δύο κόμβων ενός γράφου, δηλαδή κάποια που έχει το μικρότερο δυνατό συνολικό κόστος ακμών. Υπάρχουν διάφοροι αλγόριθμοι για την αντιμετώπιση αυτού του προβλήματος, άλλοι περισσότερο, άλλοι λιγότερο αποδοτικοί. Αντικείμενο της άσκησης αυτής είναι να υλοποιήσετε σε C, και μάλιστα με διαφορετικούς τρόπους, έναν συγκεκριμένο αλγόριθμο εύρεσης συντομότερου μονοπατιού σε ένα γράφο, τον αλγόριθμο *Floyd-Warshall*.



Περί του αλγορίθμου *Floyd-Warshall*

Ο αλγόριθμος αυτός (`http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm`) βρίσκει τα μήκη των συντομότερων μονοπατιών μεταξύ όλων των ζευγαριών κόμβων του δεδομένου γράφου. Έστω ότι έχουμε N κόμβους, που είναι αριθμημένοι από το 0 έως το $N-1$. Αν με P_{ijk} συμβολίσουμε το μήκος του συντομότερου μονοπατιού από τον κόμβο i προς τον κόμβο j, με ενδιάμεσους κόμβους μέσα από το σύνολο $\{0, 1, \ldots, k\}$, τότε η ουσία του αλγορίθμου Floyd-Warshall βρίσκεται στην εξής αναδρομική διατύπωση:

$$P_{ijk} = \begin{cases} min\ \{\ P_{i,j,k-1}, \quad P_{i,k,k-1} + P_{k,j,k-1}\ \}, & \text{αν} \quad k \geq 0 \\ C_{ij}, & \text{αν} \quad k < 0 \end{cases}$$

Η ερμηνεία του παραπάνω είναι ότι για να βρούμε το μήκος του ελάχιστου μονοπατιού P_{ijk} από τον κόμβο i προς τον κόμβο j, με ενδιάμεσους κόμβους μέσα από το σύνολο $\{0, 1, \ldots, k\}$, μπορούμε να βρούμε

- το μήκος του ελάχιστου μονοπατιού $P_{i,j,k-1}$ από τον κόμβο i προς τον κόμβο j με ενδιάμεσους κόμβους μέσα από το σύνολο $\{0, 1, \ldots, k-1\}$ και
- τα μήκη των ελάχιστων μονοπατιών από τον κόμβο i προς τον κόμβο k και από τον κόμβο k προς τον κόμβο j με ενδιάμεσους κόμβους μέσα από το σύνολο $\{0, 1, \ldots, k-1\}$, δηλαδή τα $P_{i,k,k-1}$ και $P_{k,j,k-1}$, αντίστοιχα,

και να πάρουμε το ελάχιστο των $P_{i,j,k-1}$ και $P_{i,k,k-1} + P_{k,j,k-1}$.

Με άλλα λόγια, για να βρούμε το μήκος του ελάχιστου μονοπατιού από τον κόμβο i προς τον κόμβο j, με ενδιάμεσους κόμβους μέσα από το σύνολο $\{0, 1, \ldots, k\}$, είτε δεν θα χρησιμοποιήσουμε σαν ενδιάμεσο κόμβο τον k, είτε θα τον χρησιμοποιήσουμε, αλλά τα μονοπάτια από τον i στον k και από τον k στον j δεν θα τον χρησιμοποιούν. Όποια από τις δύο εκδοχές είναι συντομότερη, αυτή δίνει το ελάχιστο μονοπάτι από τον i στο j, με κόμβους μέχρι τον k.

Φυσικά, αν $k = -1$, τότε δεν έχουμε ενδιάμεσους κόμβους, οπότε το μήκος του ελάχιστου μονοπατιού από τον κόμβο i προς τον κόμβο j είναι το μήκος της ακμής C_{ij} από τον i στον j, εφόσον οι κόμβοι αυτοί έχουν απ΄ ευθείας σύνδεση.

Το μήκος του ελάχιστου μονοπατιού από οποιονδήποτε κόμβο i σε οποιονδήποτε κόμβο j σε γράφο με N κόμβους, χωρίς περιορισμό στους ενδιάμεσους κόμβους, είναι το $P_{i,j,N-1}$.

Πλαίσιο υλοποίησης

Στην εργασία αυτή καλείσθε να υλοποιήσετε εναλλακτικές μεθόδους επίλυσης του προβλήματος εύρεσης συντομότερης διαδρομής μεταξύ των κόμβων ενός γράφου με βάση τον αλγόριθμο *Floyd-Warshall*. Συγκεκριμένα, οι μέθοδοι αυτές είναι:

- Αναδρομική μέθοδος (recursive)
- Αναδρομική μέθοδος με απομνημόνευση (recursive with memoization)
- Επαναληπτική μέθοδος με δυναμικό προγραμματισμό (dynamic programming)

Όλες οι μέθοδοι θα βασισθούν στη μαθηματική προσέγγιση που περιγράφηκε προηγουμένως και θα πρέπει να υλοποιηθούν μέσω της κλήσης από τη `main()` μίας συνάρτησης με πρωτότυπο

```
void solve(int n, int **graph)
```

όπου `n` είναι το πλήθος των κόμβων του γράφου `graph`, ο οποίος κωδικοποιείται σαν ένας δισδιάστατος πίνακας που κάθε στοιχείο του έχει σαν τιμή το μήκος της ακμής μεταξύ των κόμβων που αντιστοιχούν στη γραμμή και τη στήλη του στοιχείου. Αν δύο κόμβοι δεν συνδέονται με ακμή, το αντίστοιχο στοιχείο στον πίνακα έχει τιμή -1. Η συνάρτηση πρέπει να εκτυπώνει τα μήκη των ελάχιστων μονοπατιών μεταξύ όλων των ζευγαριών κόμβων του γράφου. Εννοείται ότι κάθε συνάρτηση `solve()` θα μπορεί να καλεί άλλες συναρτήσεις, όπου το κρίνετε απαραίτητο.

Κάθε συνάρτηση `solve()` που υλοποιεί μία από τις προαναφερθείσες μεθόδους θα πρέπει να βρίσκεται σε διαφορετικό πηγαίο αρχείο `.c`. Εννοείται ότι πρέπει να έχετε δημιουργήσει και αρχείο/α επικεφαλίδας `.h`, όπου αυτά απαιτούνται. Επίσης, σε διαφορετικό πηγαίο αρχείο θα πρέπει να βρίσκεται και η συνάρτηση `main()` του προγράμματός σας. Για να κατασκευάσετε το εκάστοτε εκτελέσιμο, ανάλογα με τη μέθοδο που πρέπει να χρησιμοποιηθεί, θα πρέπει να μεταγλωττίσετε κάθε πηγαίο αρχείο στο αντίστοιχο αντικειμενικό και μετά να συνδέσετε το αντικειμενικό αρχείο της `main()` με το κατάλληλο αντικειμενικό αρχείο της μεθόδου που σας ενδιαφέρει. Ένα παράδειγμα της διαδικασίας που πρέπει να ακολουθήσετε είναι το εξής:

```
$ gcc -c -o main.o main.c
$ gcc -c -o flowarrec.o flowarrec.c
$ gcc -c -o flowarmem.o flowarmem.c
$ gcc -c -o flowardp.o flowardp.c
$ gcc -o flowarrec main.o flowarrec.o
$ gcc -o flowarmem main.o flowarmem.o
$ gcc -o flowardp main.o flowardp.o
```

Αναδρομική υλοποίηση του αλγορίθμου Floyd-Warshall (25%)

Υλοποιήστε τη συνάρτηση `solve()` που υπολογίζει με αναδρομικό τρόπο[1] και εκτυπώνει τα μήκη των ελάχιστων μονοπατιών μεταξύ όλων των ζευγαριών κόμβων του γράφου (αρχείο `flowarrec.c`), σύμφωνα με τον αναδρομικό τύπο που περιγράφηκε προηγουμένως, καθώς και κατάλληλη `main()` που θα την καλεί (αρχείο `main.c`). Η `main()` να διαβάζει από την πρότυπη είσοδο τα δεδομένα του γράφου όπως περιγράφεται στη συνέχεια και να καλεί την `solve()`. Στο πρόγραμμά σας δεν επιτρέπεται να ορίσετε άλλον πίνακα εκτός από αυτόν που χρειάζεται για τη φύλαξη των μηκών των ακμών του γράφου. Όσον αφορά τη μορφή της εισόδου στο πρόγραμμα, αυτή πρέπει να είναι της μορφής:

$\langle N \rangle$
$\langle e_{1\to 0} \rangle$
$\langle e_{2\to 0} \rangle$ $\langle e_{2\to 1} \rangle$
$\langle e_{3\to 0} \rangle$ $\langle e_{3\to 1} \rangle$ $\langle e_{3\to 2} \rangle$
...
$\langle e_{(N-1)\to 0} \rangle$ $\langle e_{(N-1)\to 1} \rangle$ ... $\langle e_{(N-1)\to (N-2)} \rangle$

Το $\langle N \rangle$ είναι το πλήθος των κόμβων του γράφου και το κάθε $\langle e_{i\to j} \rangle$ είναι το μήκος της ακμής από τον κόμβο i προς στον κόμβο j. Αν δεν υπάρχει ακμή που να συνδέει τους κόμβους i και j, τότε σαν $\langle e_{i\to j} \rangle$ να δίνεται το -1.

Αν το εκτελέσιμο πρόγραμμα που θα κατασκευάσετε τελικά ονομάζεται "`flowarrec`", μία ενδεικτική εκτέλεσή του, για τον γράφο της πρώτης σελίδας, φαίνεται στη συνέχεια. Τα δεδομένα για τον γράφο αυτό μπορείτε να τα βρείτε στο `http://www.di.uoa.gr/~ip/hwfiles/flowar/demo.txt`, οπότε μπορείτε να εκτελέσετε το πρόγραμμα σαν "`./flowarrec < demo.txt`".

```
$ ./flowarrec
10
 -1
 35  -1
 25   9  15
 -1  -1  28  42
 43  -1  -1   5  -1
 -1  33  -1  -1  -1  48
 -1  39  13  50  -1   4  -1
 -1  -1  15  -1  -1  -1  48  -1
  8  -1  -1  -1  -1  -1  17  -1  43
^D
From node  1 to node  0: Length of shortest path is 34

From node  2 to node  0: Length of shortest path is 35
From node  2 to node  1: Length of shortest path is 24

From node  3 to node  0: Length of shortest path is 25
From node  3 to node  1: Length of shortest path is 9
From node  3 to node  2: Length of shortest path is 15

From node  4 to node  0: Length of shortest path is 63
From node  4 to node  1: Length of shortest path is 51
From node  4 to node  2: Length of shortest path is 28
```

[1] Επισημαίνεται ότι δεν είναι απαραίτητο η ίδια η `solve()` να είναι αναδρομική. Αυτό που ζητείται είναι να λύνει το πρόβλημα με αναδρομικό τρόπο. Θα μπορούσε, για παράδειγμα, να καλεί άλλη συνάρτηση που να είναι αναδρομική.

```
From node  4 to node  3: Length of shortest path is 42

From node  5 to node  0: Length of shortest path is 30
From node  5 to node  1: Length of shortest path is 14
From node  5 to node  2: Length of shortest path is 17
From node  5 to node  3: Length of shortest path is 5
From node  5 to node  4: Length of shortest path is 45

From node  6 to node  0: Length of shortest path is 25
From node  6 to node  1: Length of shortest path is 33
From node  6 to node  2: Length of shortest path is 57
From node  6 to node  3: Length of shortest path is 42
From node  6 to node  4: Length of shortest path is 84
From node  6 to node  5: Length of shortest path is 47

From node  7 to node  0: Length of shortest path is 34
From node  7 to node  1: Length of shortest path is 18
From node  7 to node  2: Length of shortest path is 13
From node  7 to node  3: Length of shortest path is 9
From node  7 to node  4: Length of shortest path is 41
From node  7 to node  5: Length of shortest path is 4
From node  7 to node  6: Length of shortest path is 51

From node  8 to node  0: Length of shortest path is 50
From node  8 to node  1: Length of shortest path is 39
From node  8 to node  2: Length of shortest path is 15
From node  8 to node  3: Length of shortest path is 30
From node  8 to node  4: Length of shortest path is 43
From node  8 to node  5: Length of shortest path is 32
From node  8 to node  6: Length of shortest path is 48
From node  8 to node  7: Length of shortest path is 28

From node  9 to node  0: Length of shortest path is 8
From node  9 to node  1: Length of shortest path is 42
From node  9 to node  2: Length of shortest path is 43
From node  9 to node  3: Length of shortest path is 33
From node  9 to node  4: Length of shortest path is 71
From node  9 to node  5: Length of shortest path is 38
From node  9 to node  6: Length of shortest path is 17
From node  9 to node  7: Length of shortest path is 42
From node  9 to node  8: Length of shortest path is 43
```

Μπορείτε να δοκιμάσετε το πρόγραμμά σας και με άλλους γράφους, που γεννώνται τυχαία από το πρόγραμμα "`randgr_<arch>`", όπου το `<arch>` είναι `linux`, `windows.exe` ή `macosx`, ανάλογα με το σύστημα που σας ενδιαφέρει. Τα εκτελέσιμα αυτού του προγράμματος για τις προηγούμενες αρχιτεκτονικές μπορείτε να τα βρείτε στο `http://www.di.uoa.gr/~ip/hwfiles/flowar`. Το πρόγραμμα "`randgr_<arch>`" δέχεται τις εξής επιλογές:

`-n <nodes>` : Το πλήθος των κόμβων του γράφου είναι `<nodes>` (default τιμή: 10).

`-w <maxCost>` : Το μέγιστο κόστος ακμής είναι `<maxCost>` (default τιμή: 10).

`-p <prob>` : Η πιθανότητα ύπαρξης ακμής (από 0 έως 100) μεταξύ δύο κόμβων είναι `<prob>` (default τιμή: 100).

`-o <outputFile>` : Τα δεδομένα του γράφου αποθηκεύονται, στη μορφή που πρέπει να διαβάζονται από το πρόγραμμα που θα γράψετε, στο αρχείο `<outputFile>` (αν δεν δοθεί η επιλογή, εκτυπώνονται στην έξοδο).

`-s <seed>` : Το φύτρο της γεννήτριας τυχαίων αριθμών είναι `<seed>` (αν δεν δοθεί η επιλογή, σαν φύτρο χρησιμοποιείται ο τρέχων χρόνος).

`-d <dotFile>` : Τα δεδομένα του γράφου αποθηκεύονται στο αρχείο `<dotFile>` σε "`dot`" μορφή. Η μορφή αυτή μπορεί να χρησιμοποιηθεί από το πρόγραμμα Graphviz, που μπορείτε να κατεβάσετε από το `http://www.graphviz.org`, για την οπτικοποίηση γράφων που αναπαρίστανται σε "`dot`" μορφή, όπως ο γράφος στην πρώτη σελίδα. Περισσότερες λεπτομέρειες για τις διαδικασίες οπτικοποίησης γράφων μπορείτε να βρείτε στην τεκμηρίωση του προγράμματος Graphviz.

`-v <nodesFile>` : Αν δοθεί η επιλογή αυτή, σε συνδυασμό με την προηγούμενη, τα ονόματα των κόμβων για την οπτικοποίηση του γράφου παίρνονται κατά σειρά από το αρχείο `<nodesFile>`. Αλλιώς, σαν ονόματα χρησιμοποιούνται τα `city0`, `city1`, κλπ.

Κάποιες επιπλέον εκτελέσεις του προγράμματος "`flowarrec`"[2], με τυχαίες εισόδους που παράγονται από το πρόγραμμα "`randgr_linux`", φαίνονται στη συνέχεια.

```
$ ./randgr_linux -n 5 -p 20 -s 1 -o test.txt
$ cat test.txt
5
  7
 -1  -1
  6   3  -1
 -1  -1  -1  10
$ /usr/bin/time ./flowarrec < test.txt

From node  1 to node  0: Length of shortest path is 7

From node  2 to node  0: There is no path
From node  2 to node  1: There is no path

From node  3 to node  0: Length of shortest path is 6
From node  3 to node  1: Length of shortest path is 3
From node  3 to node  2: There is no path

From node  4 to node  0: Length of shortest path is 16
From node  4 to node  1: Length of shortest path is 13
From node  4 to node  2: There is no path
From node  4 to node  3: Length of shortest path is 10
0.00user 0.00system 0:00.00elapsed 0%CPU .........
```

[2]Όλες οι εκτελέσεις της εκφώνησης έγιναν στον υπολογιστή `linux29` του εργαστηρίου του Τμήματος.

```
$ ./randgr_linux -n 12 -s 2018 | /usr/bin/time ./flowarrec

From node  1 to node  0: Length of shortest path is 3

From node  2 to node  0: Length of shortest path is 4
From node  2 to node  1: Length of shortest path is 1

From node  3 to node  0: Length of shortest path is 2
.............
From node 11 to node  8: Length of shortest path is 4
From node 11 to node  9: Length of shortest path is 6
From node 11 to node 10: Length of shortest path is 3
0.20user 0.00system 0:00.21elapsed 99%CPU ..........

$ ./randgr_linux -n 16 -s 2019 | /usr/bin/time ./flowarrec

From node  1 to node  0: Length of shortest path is 1

From node  2 to node  0: Length of shortest path is 5
.............
From node 15 to node 14: Length of shortest path is 2
29.41user 0.00system 0:29.42elapsed 99%CPU ..........

$ ./randgr_linux -n 18 -s 1000 | /usr/bin/time ./flowarrec
.............
327.56user 0.00system 5:27.56elapsed 99%CPU ..........
```

Αναδρομική μέθοδος με απομνημόνευση (25%)

Παρατηρείτε ότι όσο μεγαλώνει ο γράφος, τόσο πιο πολύ αργεί η εκτέλεση του προγράμματός σας; Και μάλιστα, η αύξηση του χρόνου εκτέλεσης γίνεται με μεγάλο ρυθμό, που ονομάζεται *εκθετικός*, και είναι τέτοιος που από κάποιο μέγεθος γράφου και πάνω, το πρόγραμμά σας πρακτικά δεν δίνει καθόλου αποτελέσματα. Όπως ίσως μπορείτε να καταλάβετε, αυτό οφείλεται στο ότι στην αναδρομική συνάρτηση που έχετε γράψει, καλείται ο εαυτός της τρεις φορές. Πώς θα μπορούσαμε να το διορθώσουμε αυτό; Αρκεί να χρησιμοποιήσουμε ένα τρισδιάστατο πίνακα στον οποίο να αποθηκεύεται κάθε P_{ijk} την πρώτη φορά που υπολογίζεται και όταν χρειάζεται πάλι, να μην επαναϋπολογίζεται, αλλά να λαμβάνεται η τιμή του από τον πίνακα. Η λογική της βελτιωμένης μεθόδου εξακολουθεί να είναι αναδρομική, βασισμένη στη μαθηματική σχέση που δόθηκε, απλώς κάθε P_{ijk} υπολογίζεται ακριβώς μία φορά και φυλάσσεται στον πίνακα για μελλοντική χρήση. Υλοποιήστε τη συνάρτηση `solve()` και με βάση αυτήν την προσέγγιση (αρχείο `flowarmem.c`). Ενδεικτικές εκτελέσεις:

```
$ ./randgr_linux -n 12 -s 2018 | /usr/bin/time ./flowarmem

From node  1 to node  0: Length of shortest path is 3

From node  2 to node  0: Length of shortest path is 4
From node  2 to node  1: Length of shortest path is 1
```

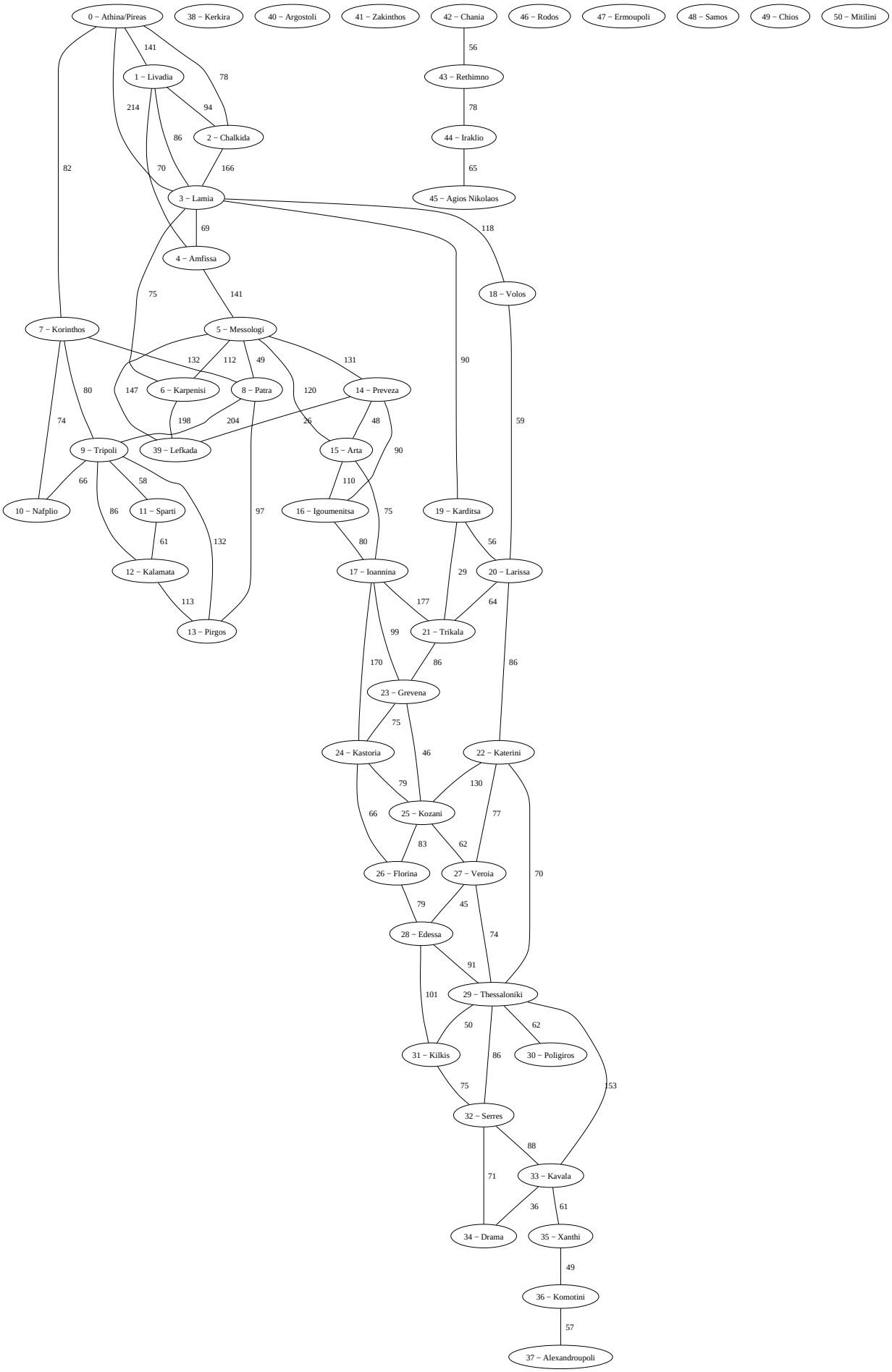
```
From node  3 to node  0: Length of shortest path is 2
.............
From node 11 to node  8: Length of shortest path is 4
From node 11 to node  9: Length of shortest path is 6
From node 11 to node 10: Length of shortest path is 3
0.00user 0.00system 0:00.00elapsed 66%CPU ..........

$ ./randgr_linux -n 16 -s 2019 | /usr/bin/time ./flowarmem

From node  1 to node  0: Length of shortest path is 1

From node  2 to node  0: Length of shortest path is 5
.............
From node 15 to node 14: Length of shortest path is 2
0.00user 0.00system 0:00.00elapsed 100%CPU ..........

$ ./randgr_linux -n 18 -s 1000 | /usr/bin/time ./flowarmem
.............
0.00user 0.00system 0:00.00elapsed 66%CPU ..........

$ ./randgr_linux -n 300 -s 999 | /usr/bin/time ./flowarmem
.............
1.26user 0.15system 0:01.61elapsed 87%CPU
```

Επαναληπτική μέθοδος με δυναμικό προγραμματισμό (25%)

Στην αναδρομική μέθοδο με απομνημόνευση, η λογική του υπολογισμού του μήκους του ελάχιστου μονοπατιού μεταξύ των κόμβων i και j, δηλαδή του $P_{i,j,N-1}$, είναι με σειρά *από-επάνω-προς-τα-κάτω* (top-down), δηλαδή αρχίζουμε από το $k = N - 1$, οπότε απαιτείται ο υπολογισμός των τιμών για μικρότερα k, μετά για ακόμα μικρότερα, κ.ο.κ. μέχρι να φτάσουμε να μην χρειάζεται αναδρομή. Μία αντίστροφη λογική από αυτή είναι να υπολογίζεται το κάθε $P_{i,j,N-1}$ με σειρά *από-κάτω-προς-τα-επάνω* (bottom-up). Δηλαδή, αρχίζουμε τους υπολογισμούς από τα μικρά k προς τα μεγαλύτερα και τελικά καταλήγουμε στην περίπτωση του $k = N - 1$. Θα χρειαστεί, βέβαια, να έχουμε και έναν επιπλέον πίνακα Q, που τελικά σε κάθε στοιχείο του Q_{ij} θα αποθηκεύεται το μήκος του ελάχιστου μονοπατιού μεταξύ των κόμβων i και j, δηλαδή το $Q_{ij} = P_{i,j,N-1}$. Η προσέγγιση αυτή χαρακτηρίζεται στη βιβλιογραφία ως *δυναμικός προγραμματισμός* (dynamic programming). Υλοποιήστε τη συνάρτηση `solve()` και με βάση τη λογική αυτή (αρχείο `flowardp.c`).

Πλέον, με την επαναληπτική υλοποίηση, μπορείτε να εφαρμόζετε τον αλγόριθμο Floyd-Warshall για να βρίσκετε τα μήκη των ελάχιστων μονοπατιών μεταξύ όλων των ζευγαριών κόμβων του γράφου, ακόμα και για πολύ μεγάλους γράφους. Εκτελέστε το πρόγραμμά σας, στην επαναληπτική εκδοχή του, για τον γράφο που φαίνεται στην επόμενη σελίδα. Ο γράφος αυτός έχει σαν κόμβους τις πρωτεύουσες των νομών της Ελλάδας και περιλαμβάνει ένα μεγάλο πλήθος από δρόμους που τις συνδέουν, μαζί με τα αντίστοιχα μήκη των δρόμων. Τον γράφο αυτό, στη μορφή που απαιτεί το πρόγραμμα μπορείτε να τον βρείτε στο `http://www.di.uoa.gr/~ip/hwfiles/flowar/greece.txt`. Κάτω από το `http://www.di.uoa.gr/~ip/hwfiles/flowar` μπορείτε να βρείτε και τα αρχεία `greece07.txt`, `greece14.txt`, `greece18.txt`, `greece22.txt` και `greece38.txt`, που αναπαριστούν υποσύνολα του πλήρους γράφου (οι πρώτες 7, 14, 18, 22 και 38 πρωτεύουσες, αντίστοιχα — όλες είναι 51). Οπότε, μπορείτε να δοκιμάσετε το πρόγραμμά σας και για αυτές τις ενδιάμεσου μεγέθους εισόδους.

Παραδείγματα εκτέλεσης φαίνονται από τη μεθεπόμενη σελίδα και μετά.


0 − Athina/Pireas
38 − Kerkira
40 − Argostoli
41 − Zakinthos
42 − Chania
46 − Rodos
47 − Ermoupoli
48 − Samos
49 − Chios
50 − Mitilini
1 − Livadia
43 − Rethimno
2 − Chalkida
44 − Iraklio
3 − Lamia
45 − Agios Nikolaos
4 − Amfissa
18 − Volos
7 − Korinthos
5 − Messologi
6 − Karpenisi
8 − Patra
14 − Preveza
9 − Tripoli
39 − Lefkada
15 − Arta
10 − Nafplio
11 − Sparti
16 − Igoumenitsa
19 − Karditsa
12 − Kalamata
17 − Ioannina
20 − Larissa
13 − Pirgos
21 − Trikala
23 − Grevena
24 − Kastoria
22 − Katerini
25 − Kozani
26 − Florina
27 − Veroia
28 − Edessa
29 − Thessaloniki
31 − Kilkis
30 − Poligiros
32 − Serres
33 − Kavala
34 − Drama
35 − Xanthi
36 − Komotini
37 − Alexandroupoli
141
78
56
214
94
78
86
70
166
82
65
69
118
75
141
90
59
132
112
49
131
80
147
120
74
198
204
48
90
66
58
110
86
97
75
61
132
80
56
29
113
177
64
99
86
170
86
75
46
79
130
66
77
83
62
70
79
45
74
91
101
50
62
86
153
75
88
71
36
61
49
57

```
$ ./flowardp < greece14.txt

From node  1 to node  0: Length of shortest path is 141

From node  2 to node  0: Length of shortest path is 78
From node  2 to node  1: Length of shortest path is 94

From node  3 to node  0: Length of shortest path is 214
From node  3 to node  1: Length of shortest path is 86
From node  3 to node  2: Length of shortest path is 166

From node  4 to node  0: Length of shortest path is 211
From node  4 to node  1: Length of shortest path is 70
From node  4 to node  2: Length of shortest path is 164
From node  4 to node  3: Length of shortest path is 69

From node  5 to node  0: Length of shortest path is 263
From node  5 to node  1: Length of shortest path is 211
From node  5 to node  2: Length of shortest path is 305
From node  5 to node  3: Length of shortest path is 187
From node  5 to node  4: Length of shortest path is 141

From node  6 to node  0: Length of shortest path is 289
From node  6 to node  1: Length of shortest path is 161
From node  6 to node  2: Length of shortest path is 241
From node  6 to node  3: Length of shortest path is 75
From node  6 to node  4: Length of shortest path is 144
From node  6 to node  5: Length of shortest path is 112

From node  7 to node  0: Length of shortest path is 82
From node  7 to node  1: Length of shortest path is 223
From node  7 to node  2: Length of shortest path is 160
From node  7 to node  3: Length of shortest path is 296
From node  7 to node  4: Length of shortest path is 293
From node  7 to node  5: Length of shortest path is 181
From node  7 to node  6: Length of shortest path is 293

From node  8 to node  0: Length of shortest path is 214
From node  8 to node  1: Length of shortest path is 260
From node  8 to node  2: Length of shortest path is 292
From node  8 to node  3: Length of shortest path is 236
From node  8 to node  4: Length of shortest path is 190
From node  8 to node  5: Length of shortest path is 49
From node  8 to node  6: Length of shortest path is 161
From node  8 to node  7: Length of shortest path is 132

From node  9 to node  0: Length of shortest path is 162
From node  9 to node  1: Length of shortest path is 303
From node  9 to node  2: Length of shortest path is 240
```

```
From node  9 to node  3: Length of shortest path is 376
From node  9 to node  4: Length of shortest path is 373
From node  9 to node  5: Length of shortest path is 253
From node  9 to node  6: Length of shortest path is 365
From node  9 to node  7: Length of shortest path is 80
From node  9 to node  8: Length of shortest path is 204

From node 10 to node  0: Length of shortest path is 156
From node 10 to node  1: Length of shortest path is 297
From node 10 to node  2: Length of shortest path is 234
From node 10 to node  3: Length of shortest path is 370
From node 10 to node  4: Length of shortest path is 367
From node 10 to node  5: Length of shortest path is 255
From node 10 to node  6: Length of shortest path is 367
From node 10 to node  7: Length of shortest path is 74
From node 10 to node  8: Length of shortest path is 206
From node 10 to node  9: Length of shortest path is 66

From node 11 to node  0: Length of shortest path is 220
From node 11 to node  1: Length of shortest path is 361
From node 11 to node  2: Length of shortest path is 298
From node 11 to node  3: Length of shortest path is 434
From node 11 to node  4: Length of shortest path is 431
From node 11 to node  5: Length of shortest path is 311
From node 11 to node  6: Length of shortest path is 423
From node 11 to node  7: Length of shortest path is 138
From node 11 to node  8: Length of shortest path is 262
From node 11 to node  9: Length of shortest path is 58
From node 11 to node 10: Length of shortest path is 124

From node 12 to node  0: Length of shortest path is 248
From node 12 to node  1: Length of shortest path is 389
From node 12 to node  2: Length of shortest path is 326
From node 12 to node  3: Length of shortest path is 446
From node 12 to node  4: Length of shortest path is 400
From node 12 to node  5: Length of shortest path is 259
From node 12 to node  6: Length of shortest path is 371
From node 12 to node  7: Length of shortest path is 166
From node 12 to node  8: Length of shortest path is 210
From node 12 to node  9: Length of shortest path is 86
From node 12 to node 10: Length of shortest path is 152
From node 12 to node 11: Length of shortest path is 61

From node 13 to node  0: Length of shortest path is 294
From node 13 to node  1: Length of shortest path is 357
From node 13 to node  2: Length of shortest path is 372
From node 13 to node  3: Length of shortest path is 333
From node 13 to node  4: Length of shortest path is 287
From node 13 to node  5: Length of shortest path is 146
```

```
From node 13 to node  6: Length of shortest path is 258
From node 13 to node  7: Length of shortest path is 212
From node 13 to node  8: Length of shortest path is 97
From node 13 to node  9: Length of shortest path is 132
From node 13 to node 10: Length of shortest path is 198
From node 13 to node 11: Length of shortest path is 174
From node 13 to node 12: Length of shortest path is 113

$ ./randgr_linux -n 300 -s 999 | /usr/bin/time ./flowardp
.............
0.23user 0.12system 0:00.78elapsed 46%CPU
```

Υπολογισμός ελάχιστων μονοπατιών (25%)

Να επεκτείνετε την επαναληπτική υλοποίηση του αλγορίθμου Floyd-Warshall ώστε να υπολογίζονται και τα ελάχιστα μονοπάτια (εκτός από τα ελάχιστα κόστη) μεταξύ όλων των ζευγαριών κόμβων του γράφου. Για να μπορέσετε να ανακτήσετε και τα ίδια τα ελάχιστα μονοπάτια, εκτός από τα μήκη τους, θα χρειαστείτε, εκτός από τον Q, και ένα δεύτερο δισδιάστατο πίνακα. Το ποια θα πρέπει να είναι τα περιεχόμενα αυτού του πίνακα αφήνεται να το σκεφτείτε εσείς. Το αν θα εκτυπώνονται ή όχι τα μονοπάτια από τη συνάρτηση της επαναληπτικής υλοποίησης να εξαρτάται από το αν έχει ορισθεί η συμβολική σταθερά `PATH`. Τον κώδικα εκτύπωσης των μονοπατιών στη συνάρτηση θα πρέπει να τον περικλείσετε μέσα στις οδηγίες προς τον προεπεξεργαστή "`#ifdef PATH`" και "`#endif`" (δείτε τις σελίδες 158-159 των σημειώσεων/διαφανειών του μαθήματος). Αν θέλετε να εκτυπώνονται και τα ελάχιστα μονοπάτια, θα πρέπει είτε να έχετε κάνει `#define` την `PATH`, είτε να μεταγλωττίσετε το πρόγραμμά σας με την εντολή "`gcc -DPATH ...`" ώστε να μεταγλωττισθεί και ο κώδικας της εκτύπωσης των μονοπατιών. Ένα παράδειγμα εκτέλεσης της επαναληπτικής μεθόδου και με υπολογισμό μονοπατιών είναι το εξής:

```
$ gcc -DPATH -o flowardppath main.c flowardp.c
$ ./flowardppath < greece07.txt

From node  1 to node  0: Length of shortest path is 141
        Shortest path is: 1 -> 0

From node  2 to node  0: Length of shortest path is 78
        Shortest path is: 2 -> 0
From node  2 to node  1: Length of shortest path is 94
        Shortest path is: 2 -> 1

From node  3 to node  0: Length of shortest path is 214
        Shortest path is: 3 -> 0
From node  3 to node  1: Length of shortest path is 86
        Shortest path is: 3 -> 1
From node  3 to node  2: Length of shortest path is 166
        Shortest path is: 3 -> 2

From node  4 to node  0: Length of shortest path is 211
        Shortest path is: 4 -> 1 -> 0
From node  4 to node  1: Length of shortest path is 70
```

```
        Shortest path is: 4 -> 1
From node  4 to node  2: Length of shortest path is 164
        Shortest path is: 4 -> 1 -> 2
From node  4 to node  3: Length of shortest path is 69
        Shortest path is: 4 -> 3

From node  5 to node  0: Length of shortest path is 352
        Shortest path is: 5 -> 4 -> 1 -> 0
From node  5 to node  1: Length of shortest path is 211
        Shortest path is: 5 -> 4 -> 1
From node  5 to node  2: Length of shortest path is 305
        Shortest path is: 5 -> 4 -> 1 -> 2
From node  5 to node  3: Length of shortest path is 187
        Shortest path is: 5 -> 6 -> 3
From node  5 to node  4: Length of shortest path is 141
        Shortest path is: 5 -> 4

From node  6 to node  0: Length of shortest path is 289
        Shortest path is: 6 -> 3 -> 0
From node  6 to node  1: Length of shortest path is 161
        Shortest path is: 6 -> 3 -> 1
From node  6 to node  2: Length of shortest path is 241
        Shortest path is: 6 -> 3 -> 2
From node  6 to node  3: Length of shortest path is 75
        Shortest path is: 6 -> 3
From node  6 to node  4: Length of shortest path is 144
        Shortest path is: 6 -> 3 -> 4
From node  6 to node  5: Length of shortest path is 112
        Shortest path is: 6 -> 5
```

Παραδοτέο

Θα πρέπει να δομήσετε το πρόγραμμά σας σε ένα σύνολο από **πηγαία αρχεία** C (με κατάληξη `.c`), ένα με τη συνάρτηση `main()` και από ένα για κάθε μία από τρεις μεθόδους που ζητείται να υλοποιήσετε, και **τουλάχιστον ένα αρχείο επικεφαλίδας** (με κατάληξη `.h`).

Για να παραδώσετε το σύνολο των αρχείων που θα έχετε δημιουργήσει για την εργασία αυτή, ακολουθήστε την εξής διαδικασία. Τοποθετήστε όλα τα αρχεία μέσα σ΄ ένα κατάλογο που θα δημιουργήσετε σε κάποιο σύστημα Linux, έστω με όνομα `flowar`. Χρησιμοποιώντας την εντολή `zip` ως εξής

```
zip -r flowar.zip flowar
```

δημιουργείτε ένα συμπιεσμένο (σε μορφή `zip`) αρχείο, με όνομα `flowar.zip`, στο οποίο περιέχεται ο κατάλογος `flowar` μαζί με όλα τα περιεχόμενά του.[3] Το αρχείο αυτό είναι που θα πρέπει να υποβάλετε μέσω του eclass.[4]

[3] Αρχεία `zip` μπορείτε να δημιουργήσετε και στα Windows, με διάφορα προγράμματα, όπως το WinZip.

[4] Μην υποβάλετε ασυμπίεστα αρχεία ή αρχεία που είναι συμπιεσμένα σε άλλη μορφή εκτός από `zip` (π.χ. `rar`, `7z`, `tar`, `gz`, κλπ.), γιατί δεν θα γίνουν δεκτά για αξιολόγηση.