Άσκηση Ε

Σε ένα γράφημα, του οποίου οι κορυφές συνδέονται με ακμές οι οποίες έχουν δοθέντα βάρη, ζητείται να βρεθεί ένα μονοπάτι το οποίο να περνά **μια μόνο φορά** από κάθε κορυφή και να έχει το μικρότερο κόστος (Πρόβλημα του περιοδεύοντας εμπόρου). Ως κόστος για το μονοπάτι ορίζεται το άθροισμα των βαρών των ακμών από τις οποίες διέρχεται.. Ένας αλγόριθμος που θα έδινε μια καλή λύση (όχι τη βέλτιστη) ξεκινά από μια κορυφή εκκίνησης και ως επόμενη κορυφή για το μονοπάτι ορίζει αυτή με την οποία η κορυφή εκκίνησης συνδέεται με την ακμή που έχει το μικρότερο βάρος. Στη συνέχεια η κορυφή εκκίνησης αφαιρείται από το γράφημα και η διαδικασία επαναλαμβάνεται θεωρώντας ως κορυφή εκκίνησης την επόμενη κορυφή που εντοπίστηκε με το πιο πάνω κριτήριο. Ο αλγόριθμος τερματίζεται όταν το μονοπάτι περάσει από όλες τις κορυφές ή όταν διαπιστώσει ότι δεν υπάρχουν κορυφές οι οποίες να συνδέονται με την κορυφή που στο συγκεκριμένο στάδιο εκτέλεσης του αλγόριθμου θεωρείται ως κορυφή εκκίνησης.

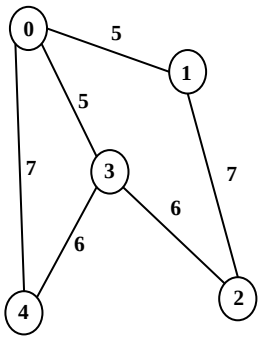
Να γραφεί το πρόγραμμα που να υλοποιεί τον πιο πάνω αλγόριθμο.

Για την εισαγωγή ενός γραφήματος, που αποτελείται από **N** κορυφές, να αριθμούνται οι κορυφές από το **0** έως το **N-1**. Στη συνέχεια να ορίζεται ο πίνακας **connection,** δύο διαστάσεων, σε κάθε γραμμή του οποίου να αντιστοιχεί και μια κορυφή του γραφήματος. Κάθε γραμμή του πίνακα να έχει τόσες θέσεις όσες είναι και οι κορυφές του γραφήματος που συνδέονται με την κορυφή στην οποία αντιστοιχεί η γραμμή (βαθμός της κορυφής). Στις θέσεις αυτές να καταχωρούνται οι αύξοντες αριθμοί των κορυφών που συνδέονται με την κορυφή που αντιστοιχεί στη γραμμή. Για την καταχώρηση των βαρών των ακμών που συνδέουν τις κορυφές να ορίζεται ο πίνακας **weights,** όμοιος με τον πίνακα **connection,** στον οποίο, σε κάθε γραμμή, να αντιστοιχεί επίσης μια κορυφή. Στον πίνακα αυτόν να έχουν αντικατασταθεί οι αύξοντες αριθμοί των κορυφών που αναγράφονται στον πίνακα **connection** με τα βάρη των ακμών που τις συνδέουν με την κορυφή στην οποία αντιστοιχεί η γραμμή στην οποία βρίσκονται.

Στο πρόγραμμα να ορίζεται η συνάρτηση **void next_vertex(…)** η οποία να δέχεται τον αύξοντα αριθμό της κορυφής εκκίνησης και να βρίσκει την επόμενη κορυφή του μονοπατιού σύμφωνα με τον πιο πάνω αλγόριθμο. Στη συνέχεια, μέσα από μια **recursion** διαδικασία, να καλεί τον εαυτό της δίνοντας ως κορυφή εκκίνησης την κορυφή που εντόπισε ως επόμενη κορυφή για το μονοπάτι.

Το πρόγραμμα αφού διαβάσει τον αριθμό **N** των κορυφών του γραφήματος και τον βαθμό της κάθε κορυφής, να δεσμεύει **δυναμικά την ελάχιστη απαραίτητη μνήμη** για την καταχώρηση των πινάκων που ορίζουν το γράφημα και να διαβάζει και να καταχωρεί τις αντίστοιχες τιμές στους πίνακες. Στη συνέχεια να διαβάζει τον αύξοντα αριθμό για την κορυφή εκκίνησης και αφού καλέσει τη συνάρτηση **next_vertex()**, να τυπώνει το μονοπάτι που βρέθηκε καθώς και το αντίστοιχο κόστος. Στην περίπτωση που ο αλγόριθμος δε μπόρεσε να περάσει το μονοπάτι από όλες τις κορυφές του γραφήματος να τυπώνεται αντίστοιχο μήνυμα.

Παράδειγμα



connection

1	3	4
0	2	
1	3	
0	2	4
0	3	

weights

5	5	7
5	7	
7	6	
5	6	6
7	6	

Κορυφή εκκίνησης 0. Μονοπάτι 0 1 2 3 4. Κόστος 24. Το μονοπάτι ολοκληρώθηκε.

Κορυφή εκκίνησης 2. Μονοπάτι 2 3 0 1. Κόστος 16. Το μονοπάτι δεν ολοκληρώθηκε.

```
int i, j, n, kor_ek, **connection, *degree, *T;
float **weights, sum_weight=0;


void next_vertex(int **conection,float **weights,int *degree,int kor_ek,int n,int *T,float sum_weight)
    {
        int i, j, sum=0, counter=0, temp;
        float min_weight=1000000;

        for (j=0; j<degree[kor_ek]; j++) {
            if (conection[kor_ek][j]!=-1){
                if (weights[kor_ek][j]<min_weight) {
                    min_weight=weights[kor_ek][j];
                    temp=conection[kor_ek][j];
                }
            }
        }
        sum_weight+=min_weight;

        for (i=0; i<n; i++) {
            for (j=0; j<degree[i]; j++) {
                if (conection[i][j]==kor_ek) {
                    conection[i][j]=-1;
                }
            }
        }




        T[temp]=1;
        kor_ek=temp;

        printf("%d ",kor_ek);

        for (j=0; j<degree[kor_ek]; j++) {
            sum+=conection[kor_ek][j];
        }

        for (i=0; i<n; i++) {
            if (T[i]==1) {
                counter++;
            }
        }




        if (counter==n) {
            printf(". Kostos %f. ",sum_weight);
            printf("To monopati oloklhrw8hke.\n");
        }
        else if (sum==(-1)*degree[kor_ek]) {
            printf(". Kostos %f. ",sum_weight);
            printf("To monopati den oloklhrw8hke.\n");
        }
        else
            next_vertex(conection,weights,degree,kor_ek,n,T,sum_weight);
```

```
}
```