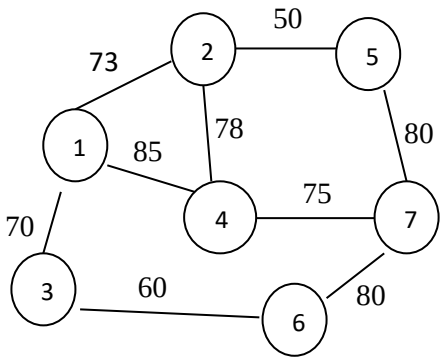
811312A Tietorakenteet ja algoritmit 2019-2020, Python-kielinen harjoitustyö

Tämä harjoitustyö toteutetaan Python-kielellä. Autoilija K:lla on kuljetusliike vuoristoisen maan eräässä kaupungissa, josta hän kuljettaa kalibroituja tarkkuusinstrumentteja muualle maahan. Mikäli instrumentit käyvät kuljetuksen aikana liian korkealla, ne joudutaan kalibroimaan uudestaan. Ottaessaan työn vastaan K. haluaa siksi tietää, mikä on päämäärään menevistä reiteistä sellainen, jonka korkein kohta on mahdollisimman matalalla. Tätä varten reittivaihtoehdot muotoillaan **suuntaamattomana verkkona**, jonka solmuja ovat kaupungit ja väleinä niiden väliset tiet. Verkko on suuntaamaton, koska teitä voidaan ajaa molempiin suuntiin. Kaikkien kaupunkien välillä ei kulje suoraa yhteyttä. Väli painotetaan tien korkeimman kohdan mukaan. Korkeudet merkitään täysinä metreinä. Kaupungit numeroidaan kokonaisluvuilla alkaen luvusta 1 siten, että 1 on aina kuorman lähtöpaikka. Esimerkiksi alla olevassa 7 kaupungin verkossa



paras reitti kaupunkiin 7 on 1->2->4->7. Tämän reitin korkein kohta on 78 metriä. Muille reiteille sattuu korkeampi kohta.

Tässä työssä laaditaan autoilija K:lle apuohjelma, jolle syötetään tieverkko korkeuksineen tekstitiedostona seuraavasti: Ensimmäisellä rivillä on välilyönnillä erotettuna kaupunkien lukumäärä ja teiden lukumäärä. Sitten luetellaan kukin omalla rivillään tiet korkeuksineen lukukolmikkoina: lähtökaupunki, maalikaupunki, tien korkeus. Viimeisellä rivillä on kaupunki, johon reittiä haetaan. Esimerkiksi edellä olevan tieverkon esitys määränpäänään kaupunki 7 olisi tiedostona

```
7 9
1 2 73
1 3 70
1 4 85
2 4 78
2 5 50
3 6 60
4 7 75
5 7 80
6 7 80
7
```

Ensimmäisellä rivillä sanotaan siis, että kaupunkeja on 7 ja tieosuuksia 9. Näin ollen seuraavat 9 riviä antavat verkon välit ja niiden painot. Viimeinen rivi kertoo määränpään, joten verkosta haetaan reittiä kaupungista 1 kaupunkiin 7. Ohjelmalle annetaan syötetiedoston nimi esimerkiksi käyttäjän syötteenä. **Ohjelma tulostaa parhaan reitin ja sen korkeuden** tai ilmoituksen, että reittiä ei ole. Tällainen on mahdollista, ellei verkossa ole lainkaan reittiä kotikaupungista annettuun. Huomaa, että paras reitti ei välttämättä ole yksikäsitteinen, mutta parhaan reitin korkeus on.

Ohjelman tulee selviytyä kohtuullisessa ajassa verkoista, joissa on 100 kaupunkia.

HUOM! Ongelmaa EI VOI ratkaista käymällä kaikki verkon reitit läpi, koska jo 20 solmun verkossa voi olla yli 6 000 000 000 000 000 erilaista reittiä kahden solmun välillä. Näin ollen ratkaisut, joissa generoidaan kaikki reitit rekursiivisesti, palautetaan automaattisesti korjattavaksi.

Opastus: Ratkaisussa on syytä käyttää jotakin sopivaa tunnettua verkkoalgoritmia. Ongelmaa voi tarkastella esimerkiksi määränpääsolmun saavutettavuusongelmana!

Mittaa ohjelmasi suoritusaikoja testisyötteillä. **Arvioi ohjelmasi aikakompleksisuutta**, kun syötteen koon mittana käytetään kaupunkien lukumäärää. Arvioi tämän ja mittauksiesi perusteella, kuinka monen kaupungin muodostamia verkkoja ohjelmalla voidaan käsitellä 15 sekunnissa.

Ohjelmakoodin lisäksi palautetaan työselostus, jossa kuvataan

1. ratkaisu ja
2. ohjelman suorituskyvyn analyysi, kuten edellä mainittiin.

Analyysi sisältää siis käytetyn algoritmin analyysin ja ohjelman suoritusaikojen mittaukset.

Ohjelmakoodin tulee olla asiallisesti kommentoitu, mutta muuta dokumentointia ei vaadita. Työselostukseen liitetään luonnollisesti myös nimi ja opiskelijanumero. Mikäli haluat, voit antaa työselostuksessa myös palautetta työstä, esimerkiksi tehtävään käytetty työmäärä, työn helpot ja hankalat asiat ja mitä opit työtä laatiessasi. Muutakin palautetta voi antaa.

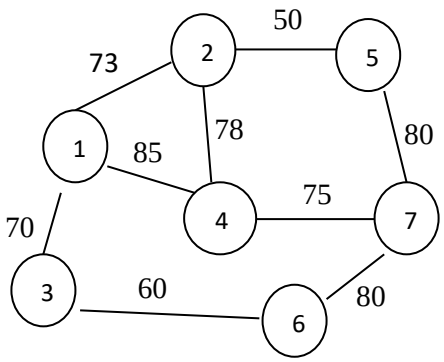
Muista pakata tiedostot yhdeksi pakkaukseksi ennen palauttamista.

Tehtävän ratkaisu on palautettava viimeistään 14.2.2020.

811312A Data Structures and Algorithms 2019-2020, Python Assignment

This assignment shall be implemented in Python. Carrier K. transports calibrated high precision measuring instruments from his home city to various cities in a mountainous country. If the carrier takes the instruments to a too high altitude during the transport, they must be re-calibrated. Thus, when K. takes a transport task, he wants to know which route to destination has its highest point as low as possible. For this, all route options are formulated as an **undirected graph**, whose nodes are cities and edges are roads connecting them. The graph is undirected, because one can drive a road in both directions. The weight of an edge will be the altitude of the highest point of the corresponding road. The altitudes are given in meters.

There is necessarily no direct road connecting a pair of cities. The cities are numbered with consecutive integers starting from 1, where 1 is always K's home city and thus the origin of the route. For example, in the following network of 7 cities



the optimal route to city 7 is 1->2->4->7. The altitude of its highest point is 78 meters. Other routes have higher highest points.

In this assignment you shall help K by implementing a program that takes as an input the road network and its properties as a text file as follows: First line of the file will contain two integers: the number of cities and the number of road segments. The next lines will each contain three integers: first two are city numbers and the third is the altitude of the highest point on the road between these cities. Last line will contain one integer representing the destination city. For example the previous network with destination city 7, would be input as the following text file:

```
7 9
1 2 73
1 3 70
1 4 85
2 4 78
2 5 50
3 6 60
4 7 75
5 7 80
6 7 80
7
```

First line tells that there are seven cities and nine roads. Thus, the following nine lines define the roads and their highest points. The last line tells that we want to know the best route from city 1 to city 7. The name of the input text file can be given as a user input to the program. The program shall **print the best route and its maximum altitude** or announce that there is no possible route. This latter can happen, if there are no routes from origin to destination. Note that the best route might not be unique but the best altitude is always unique.

Your program should be able to handle graphs with up to 100 cities in a reasonable time.

NOTE! It IS NOT possible to solve the problem by going through every route between the two cities, because already a graph with 20 nodes can have more than 6 000 000 000 000 000 different routes between two nodes. Hence, all solutions that generate recursively all routes between the cities will automatically be returned for revision.

Hint: Use some well-known graph algorithm. For example, you can consider the problem as a connectivity problem!

Measure the running times of your program with test inputs. **Estimate the time complexity of your program**, when the size of the input is the number of cities in the road network. Based on this and your measurements, estimate the maximum number of cities your program could deal with in 15 seconds.

In addition to program code, you shall return a report that describes

1. the solution, and
2. analysis of program's performance as mentioned above.

Analysis contains thus the analysis of the used algorithm and the measurements of program's running times. The code shall be commented but no other documentation is required. The report shall naturally contain student's name and student number. If you wish, you can also give feedback in your report.

Remember to zip the files before returning.

The assignment shall be returned by 14.2.2020.