Útvonalkeresés

Az elmúlt évben tapasztalható globális chiphiány a GPS gyártókat is megviselte, ezért az egyikük azt a megoldást választotta, hogy egy könnyebben beszerezhető, kisebb teljesítményű chipet alkalmaz majd következő modelljében. Emiatt mérnökök egy csoportját azzal a feladattal bízták meg, hogy fejlesszenek az új GPS modellhez egy olyan hatékony útkereső algoritmust, ami a gyengébb processzoron sem lassítja le a működést.

1. Feladatleírás

A feladat egy kapott úthálózatban különböző pontpárok között megtalálni az optimális utat. A bemenet a standard inputon tabulátorokkal (\t) és sorokkal elválasztva érkezik az alábbi sorrendben:

- azon pontpárok száma, melyek közt utat kell keresni (***p***)
- az úthálózat kereszteződéseinek (vagy csúcsainak) száma (***n***)
- az úthálózat útszakaszainak száma (***e***)
- egy üres elválasztó sor
- ***p*** számú sor, melyek mindegyikében tabulátorral elválasztva két egész szám található. Az első szám a kiinduló kereszteződés, a második pedig a célkereszteződés azonosítóját jelenti
- még egy üres elválasztó sor
- ***n*** számú sor, melyek mindegyikében tabulátorral elválasztva két egész szám található. Az első szám a kereszteződés szélességi, a második szám a hosszúsági koordinátája egy síkon. Az első sor a 0 azonosítójú kereszteződést írja le, a további sorok pedig értelemszerűen az 1,2… azonosítójúakat.
- még egy üres elválasztó sor
- ***e*** számú sor, melyek mindegyikében tabulátorral elválasztva két egész szám található. A két szám azt jelzi, hogy az adott útszakasz melyik azonosítójú kereszteződésekhez tartozik. Minden út kétirányú. Az út hossza a két végpontjának a légvonalbeli távolsága.

A p darab útvonal mindegyikére az úthossz szerinti optimális utat kell kiszámítani a megadott úthálózatban haladva.

A megoldást a standard outputra kell kiírni. A kimenet egy ***p*** számot tartalmazó tabulátorokkal elválasztott sor, ahol a számok a legrövidebb út hosszai. Az eredményt két tizedesjegy pontossággal kell megadni.

2. Feladat

Valósítsa meg az útkeresést Java vagy Python nyelven! Az algoritmus tetszőlegesen megválasztható, kódot viszont nem emelhet át külső forrásból.

2.1. Java

A megoldás tartalmazzon egy *Main* osztályt, ezen belül pedig egy *main()* függvényt. A bemenetet a standard inputon várja, a kimenetet a standard outputra írja. A program forráskódját *zip* fájlba tömörítve töltse fel a HF portálon (https://hf.mit.bme.hu).

2.2. Python

A megoldás egyetlen python fájlt tartalmazzon, amely a bemenetet a standard inputon várja, a kimenetet a standard outputra írja. A kiértékelő interpreter verziója 3.9.1, csakis a sztenderd könyvtárak használhatók a megoldásban, tehát például Numpy NEM használható. A *zip* fájlba tömörített egyetlen python fájlt töltse fel a HF portálon (https://hf.mit.bme.hu).

3. Példa feladat

3.1. Bemenet

Az alábbi példabemenetben 2 útvonalat kell kiszámolni, a térképen 3 kereszteződés és 3 él található. Az 1. útvonalat a 0-ból a 2-es azonosítójú csúcsba kell kiszámítani. A kereszteződések azonosítói és koordinátái: 0: (2,0), 1: (-4,1), 2: (6,3). Az 1-0, 1-2 és 0-2 csúcsok közt fut út, ezek hossza rendre: 6.08, 10.20 és 5.0 .

```
2
3
3

0       2
1       2

2       0
-4      1
6       3

1       0
1       2
0       2
```

3.2. Kimenet

Mivel az úthálózatunk jelen esetben egy háromszög, ezért az optimális út a csúcsok közti közvetlen útszakasz. Így a példához tartozó kimenet:

```
5.00    10.20
```

4. Értékelés

A kiértékelés három különböző térképen, automatikusan történik. Egy térképen maximum 60 útvonalat kell megkeresni. Egy térképre csak akkor jár pont, ha minden megtalált útvonal optimális hosszúságú és a program a rendelkezésre álló 60 másodperc CPU idő alatt visszatér. A végleges pontszám a három térképen szerzett pontszám összege. Sikertelenség esetén visszajelzést adunk arról, hogy mely esetekben bukott el a program, ekkor természetesen lehet újra próbálkozni.

5. Leadás

A feladat leadásának rendes határideje: **2021. október 28.**

A késedelmes leadás határideje: 2021. december 17.