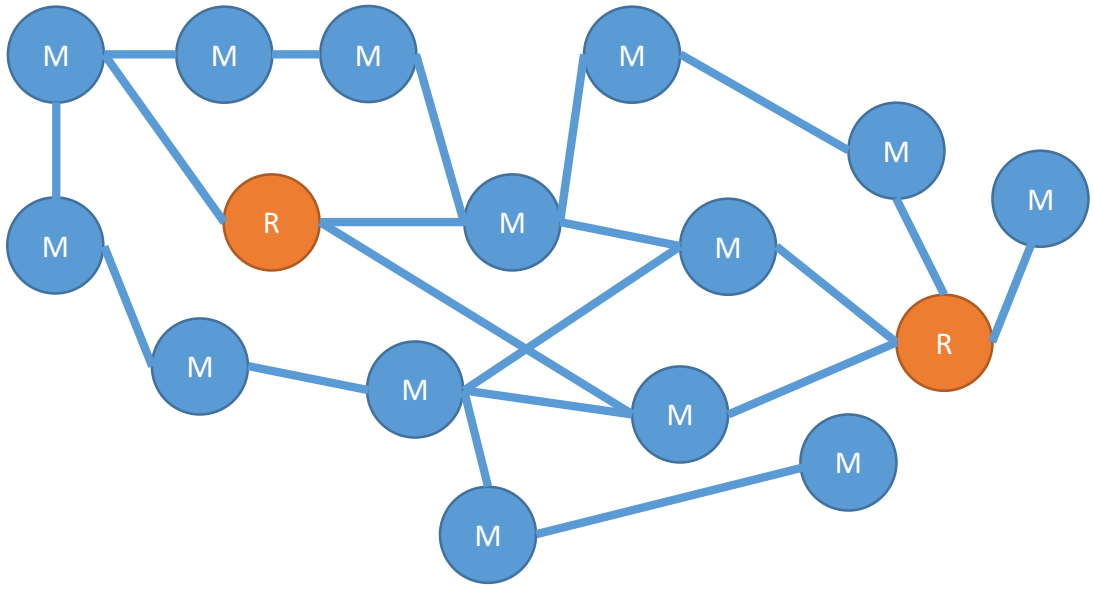
Csomag kiszállítás

A második házi egy csomagkiszállító szimulációs feladat implementálása.



A fenti gráf egy város térképét reprezentálja, ami két féle csomópontot tartalmaz:

- Kék alapon 'M' - > megrendelő (háztartás)
- Narancs alapon 'R' -> raktár (termék gyártó)

Ha két csomópont között vezet él, akkor el lehet jutni az egyikből a másikba. Minden él súlyozott, ami azt adja meg, hogy mennyi az eljutás költsége. Egy szimulációs lépés abból áll, hogy a háztartások megrendelhetnek valamilyen terméket, majd azt a futárcég kiszállítja a raktárakból.

Minden terméknek két tulajdonsága van:

- Hova kell kiszállítani (egy id, ami azonosítja a háztartást)
- Mennyi a termék súlya

A kiszállítást adott számú kamion végzi, melyek adott teherbírással rendelkeznek. Azaz egy kamion csak bizonyos mennyiségű terméket képes kiszállítani.

Feladat:

1. Olvassuk be a kapott graph.txt fájlból a gráf szerkezetét és építsük fel azt láncolt megvalósítással. (Mátrixos megvalósítás esetén nagy gráfoknál a mátrix nagyon ritkás lehet, ami miatt túl sok memóriára lenne szükség.)

Az egyes gráf csúcspontokat érdemes eltárolni valamilyen konténerben. Így a gráf destruktora könnyebben implementálható.

2. Olvassuk be a kapott simulation.txt fájlból, a kamionok számát, teherbírását, kiindulási helyüket és a megrendelt termékeket. (Egy megrendelő több terméket is rendelhet.)
3. Próbáljuk minél több terméket kiszállítani. Azaz a termékeket úgy helyezzük el a kamionokban, hogy minél kevesebb maradjon a raktárakban. (Ez egy optimalizációs feladat, ami NP teljes is lehet, ezért nem várunk tökéletes megoldást, csak egy okos, minél jobban közelítőt.) Ha csak egyszerűen sorba pakoljuk a kamionokba a termékeket, az nem tartozik az ötletes megoldások közzé ☺.
4. A kamionok a kiszállítás során, mindig a lehető legrövidebb úton közlekedjenek az egyes megrendelők között. A kamion tudja melyik csomóponban van éppen, és tudja, hogy melyik csomópontba kell a terméket kiszállítani. A legrövidebb út megkereséséhez implementáljátok a **Dijkstra algoritmust.**
5. A szimuláció végén írjátok ki, hogy melyik kamion melyik termékeket szállította ki és mennyi utat tett meg az egyes kiszállítások között. Továbbá írjátok ki, hogy mennyi termék nem lett kiszállítva.

Implementálási követelmények:

1. A gráf lácolt ábrázolással legyen reprezentálva. (Gráf osztály, belső node struktúrával.)
2. Egy metódus, ami beolvassa a gráf szerkezetét, majd felépíti azt.
3. Egy metódus, ami beolvassa a szimulációt tartalmazó adatokat.
4. Egy metódus, ami megvalósítja a Dijkstra keresést, a kiinduló és a cél node között, majd visszaadja az útvonalat.
5. Egy metódus, ami, megpróbálja minél optimálisabban a kamionokba pakolni a termékeket.

graph.txt szerkezete (példa):

5 // ennyi node van a gráfban
1 2 2 3 1 5 4 // az 1. node a 2,3,5 noddal van összekötve (fekete), az élek súlyai: 2,1,4
2 1 2// a 2. node az 1 noddal van összekötve, de ezt a fenti sorból is látható a súly: 2
3 1 1 4 5 // további összekötések
4 3 5 5 1 // további összekötések
5 1 4 // további összekötések
2 // kettő raktár van
2 // raktar1 id
5// raktar2 id

simulation.txt szerkezete (példa):

2 // ennyi kamion van
4 1// első kamion teherbírása, kiindulási node (raktár)
5 5// második kamion teherbírása, kiindulási node (raktár)
6 // ennyi terméket rendeltek
1 2 // első termék id ahova ki kell szállítani, termék súlya
1 1 // második termék id ahova ki kell szállítani, termék súlya
2 1 // további termékek adatai
5 1 // további termékek adatai
4 3 // további termékek adatai
4 2 // további termékek adatai

A példából látható, hogy a két kamion 9 súlyú terméket képes kiszállítani, de 10 súlyú a termékek összsúlya. (Tehát mindent nem tudunk kiszállítani.)

Dijkstra algoritmus: link