HF03 - Ellátási lánc logisztika

Kisvári Benedek

2025. március 10.

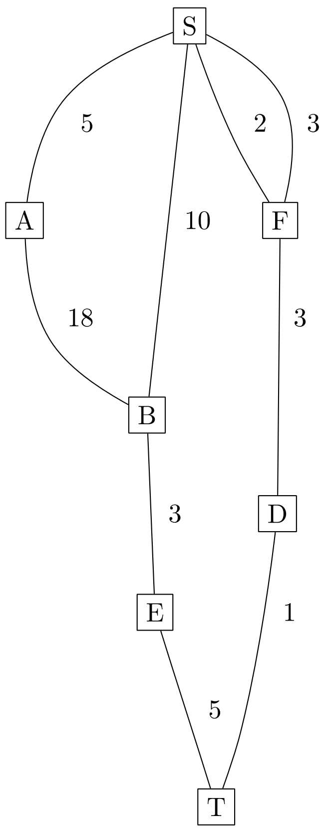
Leadási Határidő: 2025. március 24. 23:59

1. Feladat

Adott egy tetszőleges súlyozott, irányítatlan gráf. A házi feladat két részből áll. Az első rész egy adatszerkezet készítése, ami képes egy ilyen gráfot hatékonyan, referenciák segítségével eltárolni. A feladat második része egy algoritmus implementációja, ami képes meghatározni a kezdő - és végpontok között "legszélesebb" utat, azaz azt az utat, amiben a legkisebb súlyú él maximális értékű. Az utak "szélességét" (az adott időegység alatt maximálisan átvihető áru mennyiségét) az élek súlyai reprezentálják.

1.0.1. Példa

Adott a következő gráf:



1. ábra. Példa gráf

Ezt a következő függvényhívásokkal építenénk fel:

```
Graph graph = new Graph("S", "T");
graph.addNode("S", "A", 5);
```

```
graph.addNode("S", "B", 10);
graph.addNode("B", "E", 3);
graph.addNode("A", "B", 18);
graph.addNode("S", "F", 2);
graph.addNode("S", "F", 3);
graph.addNode("F", "D", 3);
graph.addNode("D", "T", 1);
graph.addNode("E", "T", 5);
```

A "legszélesebb" út pedig 5 súlyú és a következő node-okból áll: $S \to B \to E \to T$

1.1. Node osztály

A node osztályban a gráf belső (azaz nem kezdő - és végpontok) csomópontjait tároljuk el. Tartalmazza a node azonosítóját, valamint, hogy milyen más node-okhoz vezet innen él. Másik node-okra referenciák formájában hivatkozzon. Az éleket implicit módon, a node-ban tároljuk, ne készüljön hozzájuk külön objektum.

1.2. Gráf osztály

A gráf osztályban kizárólag két node-ra, a kezdő - és végpontokra legyen referencia (a példában ezek S és T). A többit a gráf bejárásával lehessen csak elérni. Készüljenek el a következő tagfüggvények az osztályhoz:

- `Graph(String startNodeName, String targetNodeName)`
  Konstruktor amivel megadjuk a kezdő és végpontok azonosítóit.

- `Integer addNode(String startPoint, String endPoint, Integer weight)`
  Ezzel lehessen megadni egy élet két node között. Ha az egyik node még nem létezik a gráfban, akkor a függvényhívással együtt jöjjön létre. A függvényhívás visszatérési értéke legyen a gráf csúcsainak száma az él és node(ok) hozzáadása után.

- `String getGraph()` Ez a függvény adja vissza a gráfot string formában, dot [1] nyelven reprezentálva. A példában lévő gráf a következő visszatérési értéket eredményezné:

  ```
  graph{
      S -- A [label=5]
      S -- B [label=10]
      B -- E [label=3]
      A -- B [label=18]
      S -- F [label=2]
      S -- F [label=3]
      F -- D [label=3]
      D -- T [label=1]
      E -- T [label=5]
  }
  ```

  Az élek sorrendje nem számít. Tesztelésre érdemes lehet ezeket használni:

  – https://magjac.com/graphviz-visual-editor
  – https://dreampuf.github.io/GraphvizOnline

1.3. Szűk keresztmetszet algoritmus

A gráf osztályhoz kerüljön implementálásra egy algoritmus, ami a korábban elkészített reprezentáció segítségével képes megválaszolni azt, hogy a kezdő és végpontok közötti utak közül melyik az, amelyikben a minimális súlyú él maximális értékű az utak között.

Ha több út is maximális szélességű, akkor a legrövidebbet adjuk vissza, azaz azt, amelyik a legkevesebb node-ot tartalmazza. Ha ezek között is van egyenlő, akkor tetszőleges, hogy melyik út kerül kiválasztásra közülük.

1.3.1. Formálisan megfogalmazva:

Legyen adott $G = (V, E, w)$ gráf, ahol V a csúcsok halmaza, E az élek halmaza, $w : E \to \mathbb{R}$ függvény, ami minden élhez hozzárendeli a súlyát. Adott $S \in V$ kezdőpont és $T \in V$ végpont.

Egy P út a gráfban S és T között a csúcsok egy sorozata, amelyet $P = (v_0, v_1, \ldots, v_k)$ jelöl, ahol $v_0 = S$ és $v_k = T$, valamint teljesül, hogy $(v_i, v_{i+1}) \in E$ minden $i = 0, 1, \ldots, k-1$ esetén.

Egy adott P úthoz tartozó élek súlyának multihalmaza $w(P) = \{w(e)|e \in E_P\}$ ahol $E_P \subseteq E$ azon élek halmaza, amelyekre teljesül, hogy $\exists (v_i, v_{i+1}) \in P$

A feladat a következő útvonal megtalálása:

$$\mathcal{P} = \arg \max_{P \in P_{all}(S,T)} \min w(P)$$

ahol P_{all} az összes S-ből T-be menő utat jelenti. Ha $n(\mathcal{P}) > 1$, azaz több mint 1 maximális szélességű út létezik, akkor a megoldás:

$$\text{path} = \min_{P \in \mathcal{P}} n(P)$$

Ha $n(\text{path}) > 1$ azaz több mint egy minimális hosszúságú út létezik, akkor tetszőleges út lehet a megoldás a minimális hosszúságúak közül.

$$\text{solution} \in_R \text{path}$$

Miért szükséges ez? *Kritikus fontosságú szoftverek esetén fontos, hogy az adott algoritmusról tudjuk bizonyítani működőképességét minden esetben. Ezt akkor lehetséges, ha formálisan is meg van fogalmazva milyen elő - és utófeltételeknek kell teljesülni a program futásakor. Jelen feladat megértéséhez és megoldásához bőven elég a szövegesen specifikált feladatot elolvasni a megvalósításhoz, de érdemes lehet ezt a specifikációt is átolvasni a tapasztalatszerzés érdekében.*

2. Követelmények

2.1. Minimum követelmények

A beadott házi feladatoknak minden alapkövetelményt teljesítenie kell, különben a feladat 0 pontos eredményű lesz.

- A beadott projektnek fordulnia és futnia kell.
- A kód működését érteni kell. A javító esetleges felmerülő kérdéseire tudni kell válaszolni. *(Kérdések felmerülése esetén a házi feladatok bemutatása gyakorlaton kívül, a javítóval előre egyeztetett időpontban történik.)*

2.2. További követelmények/javaslatok

- A kódhoz készüljön javadoc, ami hatékonyan magyarázza a működését. *(A javadoc-ot nem szükséges kiexportálva külön beadni, elég a kódba beleírni azt.)*
- A feladat megoldásához sok segítséget nyújt a collections framework már kész adatszerkezet implementációkkal. Ahol lehetséges, ezek kerüljenek felhasználásra.
- A programhoz készüljön két teszt tetszőleges gráfokkal, amik bemutatják a helyes működést.

2.3. Pontozási szempontok

A feladat 40 **pontos.**

- Gráf osztály implementációja *(40%)*
  - Helyes collections framework használat (kellően hatékony adatszerkezetek választása)
  - OOP elvek betartása (enkapszuláció, megfelelő tagfüggvények)
- Szűk keresztmetszet algoritmus implementációja *(40%)*

- Kódminőség *(10%)*
- Dokumentáció *(5%)*
- Tesztek *(5%)*

Hivatkozások

[1] „DOT language”, Graphviz. Section: docs. (), cím: https://graphviz.org/doc/info/lang.html (elérés dátuma 2025. 03. 10.).