# MINIMÁLIS KÖLTSÉGŰ FOLYAM

Beadandó feladat

Kovács Péter
kpeter@inf.elte.hu

## Formai követelmények

- A beadandó programot **C, C++, C#, Java** nyelven lehet megírni (konzolos programnak kell lennie). Aki ettől eltérő nyelvet szeretne használni, az előtte mindenképpen egyeztessen velem!
- **A programnak szigorúan be kell tartania a feladat specifikációját!**
- Feltehető, hogy **az input formátuma helyes**. Ezt nem kell ellenőrizni.
- A forráskódnak tartalmaznia kell a megértést segítő **megjegyzéseket (kommenteket)**.
- A program parancssori paramétere legyen a bemeneti és kimeneti fájl neve.
- A programot alaposan teszteljétek le, mielőtt beadjátok!

## 1. Feladat

Adott egy szöveges fájl, amely egy minimális költségű folyam problémát ír le az ún. DIMACS formátumban. Írjunk programot, amely egy ilyen inputfájlt beolvas, meghatároz egy minimális költségű megengedett folyamot, és azt egy szöveges kimeneti fájlba kiírja.

### 1.1. Input formátum

A feladatban szereplő hálózat egy irányított gráf $n$ csúccsal és $m$ éllel. Párhuzamos élek és hurokélek nincsenek a hálózatban. Minden bemeneti adat véges egész szám. Több termelő és fogyasztó csúcs is lehet, az előjeles termelés értékek összege nulla. A feladatnak nem feltétlenül létezik megengedett megoldása.

A bementi fájlok formátuma megegyezik az ún. DIMACS formátummal:
http://lpsolve.sourceforge.net/5.5/DIMACS_mcf.htm

Ez egy egyszerű szöveges formátum, amelyben az alábbi típusú információs sorok lehetnek. Minden sor első karaktere meghatározza a típusát. A sorokban szereplő adatokat egy vagy több szóköz karakter választja el.

- Problémadefiníciós sor:

  ```
  p min <n> <m>
  ```

  Ez a sor megadja a csúcsok és élek számát. A csúcsokat 1 és $n$ közötti egész számokkal azonosítjuk, az éleket pedig a két végpontjuk megadásával.

– Egy csúcsot leíró sor:

```
n <id> <termelés>
```

Azokat a csúcsokat, amelyeknek 0 a termelése, nem feltétlenül kell feltüntetni a bementben!

– Egy élt leíró sor:

```
a <i> <j> <alsó korlát> <kapacitás> <költség>
```

– Megjegyzés sorok, amelyeket a feldolgozás során figyelmen kívül kell hagyni:

```
c <tetszőleges szöveg>
```

A problémadefiníciós sor mindenképpen megelőzi az összes csúcs- és éldefiníciós sort, de ettől eltekintve a sorok sorrendje tetszőleges.

*Megjegyzés:* A bemeneti adatok tárolásához egy standard előjeles egész számtípus elegendő (pl. 32 bites), de nagy hálózatok esetén a folyam összköltsége jóval nagyobb lehet, így ennek kiszámításához nagyobb számtípust érdemes használni.

## 1.2. Output formátum

Ha a megadott feladatnak létezik megengedett megoldása, akkor a szöveges kimeneti fájl első sorába írjuk ki egy optimális folyam összköltségét. Az ezt követő $m$ sorban pedig soroljuk fel a gráf éleit lexikografikus sorrendben az alábbi formátumnak megfelelően:

```
a <i> <j> <alsó korlát> <kapacitás> <költség> <folyam>
```

Amennyiben nem létezik megengedett megoldás, akkor az alábbi szöveget írjuk ki a kimeneti fájlba:

```
INFEASIBLE
```

## 1.3. Példák

**Input:**

```
p min 4 5
n 1 1
n 4 -1
a 1 2 0 1 20
a 1 3 0 1 10
a 2 4 0 1 10
a 3 2 0 1 5
a 3 4 0 1 20
```

**Output:**

```
25
a 1 2 0 1 20 0
a 1 3 0 1 10 1
a 2 4 0 1 10 1
a 3 2 0 1 5 1
a 3 4 0 1 20 0
```

**Input:**

```
p min 4 5
n 1 2
n 4 -2
a 1 2 0 1 20
a 1 3 0 1 10
a 2 4 0 1 10
a 3 2 0 1 5
a 3 4 0 1 20
```

**Output:**

```
60
a 1 2 0 1 20 1
a 1 3 0 1 10 1
a 2 4 0 1 10 1
a 3 2 0 1 5 0
a 3 4 0 1 20 1
```

**Input:**

```
p min 4 5
n 1 3
n 4 -3
a 1 2 0 1 20
a 1 3 0 1 10
a 2 4 0 1 10
a 3 2 0 1 5
a 3 4 0 1 20
```

**Output:**

```
INFEASIBLE
```

**Input:**

```
c Simple min-cost flow problem
p min 5 6
n 1 5
n 4 -3
n 5 -2
a 1 2 0 5 40
a 1 3 0 3 20
a 2 4 0 4 20
a 3 2 0 4 10
a 3 5 0 2 30
a 4 5 0 3 10
```

**Output:**

```
270
a 1 2 0 5 40 2
a 1 3 0 3 20 3
a 2 4 0 4 20 3
a 3 2 0 4 10 1
a 3 5 0 2 30 2
a 4 5 0 3 10 0
```