Longest Flight Route

Uolevi nyert egy versenyen, és a díj egy ingyenes repülőút, amely egy vagy több járatból állhat, különböző városokon keresztül. Természetesen Uolevi olyan utat szeretne választani, amely a lehető legtöbb várost érinti.

Uolevi **Syrjälä** városból szeretne eljutni **Lehmälä** városba úgy, hogy a lehető legtöbb várost látogassa meg útközben. A lehetséges repülőjáratok listája adott, és tudjuk, hogy a repülőhálózatban nincsenek irányított körök.

Bemenet
Az első sor két egész számot tartalmaz: n (városok száma) és m (járatok száma). A városok sorszáma 1,2,…,n. Az 1-es város **Syrjälä**, az n-edik város **Lehmälä**.
Ezután mm sor következik, amelyek egy-egy járatot írnak le. Minden sorban két egész szám a és b: van egy repülőjárat a-ból b-be. Minden járat egyirányú.

Kimenet
Először nyomtasd ki az útvonalon szereplő városok maximális számát. Ezt követően nyomtasd ki a városokat abban a sorrendben, ahogyan meglátogatja őket. Bármelyik érvényes megoldás elfogadható.
Ha nem létezik megoldás, nyomtasd ki az IMPOSSIBLE szót.

Korlátok

- $2 \le n \le 10^5$
- $1 \le m \le 2 \cdot 10^5$
- $1 \le a,b \le n$

Példa

Bemenet:

```
5 5
1 2
2 5
1 3
3 4
4 5
```

Kimenet:

```
4
1 3 4 5
```

Probléma elemzése:
Adott egy irányított, ciklusmentes gráf, amely városokat és egyirányú repülőjáratokat tartalmaz. Uolevi szeretne Syrjälä városból (1-es város) eljutni Lehmälä városba (n-edik város) úgy, hogy a lehető legtöbb várost érintse az út során. A cél a leghosszabb útvonal megtalálása a kezdővárostól a célvárosig, ha létezik ilyen útvonal.

Algoritmus kidolgozása
Gráf reprezentációja: A városokat egy gráf csúcsaiként, a repülőjáratokat pedig a gráf irányított éleiként kezeljük. Az élek egyirányúak, tehát csak az egyik irányba lehet haladni a csúcsok között.

Elérhetőség ellenőrzése: Első lépésként ellenőrizni kell, hogy a célváros egyáltalán elérhető-e az indulóvárosból. Ehhez mélységi keresést (DFS) használunk az indulóvárostól indulva, hogy bejárjuk az összes lehetséges útvonalat. Ha a célvárost nem érjük el, akkor nincs megoldás, és az „IMPOSSIBLE” szót írjuk ki.

Leghosszabb út keresése: Ha a célváros elérhető, akkor újabb mélységi keresést végzünk, amely során kiszámítjuk az indulóvárostól a célvárosig vezető leghosszabb utat. Dinamikus programozással dolgozunk, ahol a dp[node] a kezdővárostól az adott csúcsig vezető leghosszabb út hossza. A gráfot rekurzívan bejárva minden csúcsra kiszámítjuk a maximális hosszúságú utat úgy, hogy minden lehetséges szomszédot megvizsgálunk, és azt választjuk, amelyik a leghosszabb utat adja.

Útvonal rekonstruálása: A leghosszabb út kiszámítása során minden csúcsnál eltároljuk, hogy melyik következő városból érkezett a leghosszabb út. Ezt egy child tömbben rögzítjük, ahol a child[node] megmutatja, hogy melyik városból haladtunk tovább a leghosszabb út során. A kezdővárostól indulva a child tömb segítségével visszakövetjük az optimális útvonalat a célvárosig.

Kimenet: Először kiírjuk az útvonalon érintett városok számát. Ezután kiírjuk a városokat a helyes sorrendben.

Példa:
Bemenet:
5 5
1 2
2 5
1 3
3 4
4 5

Gráf reprezentációja:
$1 \rightarrow 2$
$2 \rightarrow 5$
$1 \rightarrow 3$
$3 \rightarrow 4$
$4 \rightarrow 5$

Elérhetőség ellenőrzése: Elindulunk az 1-es városból, és bejárjuk a gráfot. A célvárost (5-ös város) elérjük, tehát van megoldás.

Leghosszabb út kiszámítása: Az 1-es várostól a következő utak lehetségesek: 1 → 2 → 5 (3 város), 1 → 3 → 4 → 5 (4 város). A leghosszabb út: 1 → 3 → 4 → 5, amely négy várost érint.

Útvonal rekonstruálása: A child tömb segítségével az 1-es várostól indulva visszakövetjük az optimális útvonalat: 1, 3, 4, 5.

Kimenet:
4
1 3 4 5

Hatékonyság:

Az algoritmus időbonyolultsága O(n + m), mivel a gráfot egyszer bejárjuk az elérhetőség ellenőrzése során, majd újra a leghosszabb út kiszámításakor. A térbonyolultság O(n + m), mivel a gráfot szomszédsági listával tároljuk, és kiegészítő tömböket használunk a dinamikus programozáshoz és az útvonal rekonstruálásához.

Python megvalósítás

https://github.com/pipdom/L_Algoritmusok_es_adatszerkezetek/blob/main/flight.py

CSES teszt eredmények

Submission details

Task:	Longest Flight Route
Sender:	pipdom
Submission time:	2024-12-15 11:10:39 +0200
Language:	Python3 (PyPy3)
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.05 s	»
#2	ACCEPTED	0.05 s	»
#3	ACCEPTED	0.05 s	»
#4	ACCEPTED	0.05 s	»
#5	ACCEPTED	0.05 s	»
#6	ACCEPTED	0.19 s	»
#7	ACCEPTED	0.19 s	»
#8	ACCEPTED	0.22 s	»
#9	ACCEPTED	0.21 s	»
#10	ACCEPTED	0.19 s	»
#11	ACCEPTED	0.16 s	»
#12	ACCEPTED	0.81 s	»
#13	ACCEPTED	0.05 s	»
#14	ACCEPTED	0.05 s	»
#15	ACCEPTED	0.57 s	»
#16	ACCEPTED	0.05 s	»
#17	ACCEPTED	0.57 s	»
#18	ACCEPTED	0.13 s	»
#19	ACCEPTED	0.05 s	»
#20	ACCEPTED	0.74 s	»
#21	ACCEPTED	0.05 s	»
#22	ACCEPTED	0.05 s	»
#23	ACCEPTED	0.76 s	»
#24	ACCEPTED	0.22 s	»
#25	ACCEPTED	0.05 s	»