Tourist Reform

Innen: Algowiki

Tartalomjegyzék

Feladat

Berland ország *N* városból áll ($2 \leq N \leq 4 * 10^5$), ezeket a városokat pedig *M* darab kétirányú út köti össze ($1 \leq M \leq 4 * 10^5$). Mindegyik út pontosan két várost köt össze, és két város között legfeljebb csak egy út fut. Az utakon eredetileg bármelyik városból el lehet jutni bármelyik városba.

Az országban egy turisztikai reform keretein belül **az összes kétirányú utat egyirányúvá szeretnék tenni** úgy, hogy a gráfban a minimális r_i értéket maximalizáljuk, ahol r_i jelöli az *i*-edik városból elérhető városok számát. Egy város önmagából elérhetőnek számít, és egy *a* városból elérhető *b* város, hogyha *a*-ból vezet *b*-be irányított út.

Feladat: Határozzuk meg ezt a maximális értéket, és adjunk meg egy ilyen egyirányúsított út konfigurációt, ahol a minimális r_i érték ennyi.

Az eredeti feladat

Az eredeti feladat elérhető a Codeforces-on itt (https://codeforces.com/contest/732/problem/F).

Megoldási ötletek

Helyes, de lassú megoldások

Brute force: Állítsuk elő az összes irányított út kombinációt, majd minden ilyen konfigurációra számítsuk ki a minimum r_i-t, és ezek közül válasszunk maximumot. Ez rettenetesen lassú, hiszen már csak ha az útforgatások kombinációit vesszük, az 2^m eset, melyre csak rárakódik még az r_i-k kiszámítása.

Segítségek

1. Ábrázoljuk az országot gráfként, ahol a csúcsok a városok, és az élek az utak.
2. Tekintsünk pár olyan egyszerű gráfot, amikben van kör, és amikben nincs. Mond ez valamit a keresett maximum értékről?
3. Mi lenne, ha ismernénk az elvágó éleket?
4. Az elvágó élek kivételével keletkezett komponensekben mondhatunk-e valamit biztosan a csúcsok r_i értékéről?

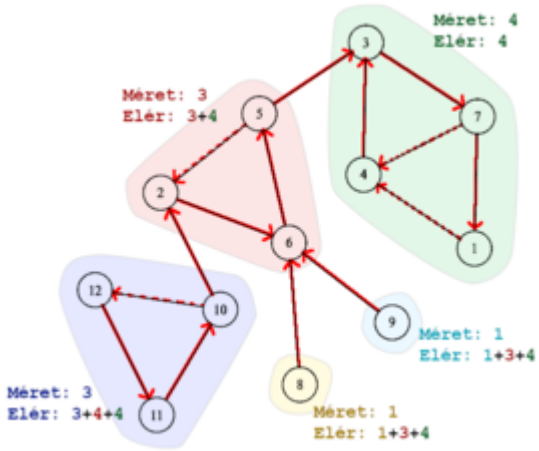
Megoldás

Fontos gondolatok

A feladat szövege miatt az ország ábrázolható egy **egyszerű, összefüggő, irányítatlan gráf**ként, ahol a **csúcsok jelentik a városokat**, az **élek pedig az utakat**.

Tekintsük az elvágó élek, és az általuk elvágott komponensek által képzett fát.

A komponensek ekkor 2-szeresen élösszefüggőek, tehát létezik erősen összefüggő élirányításuk (lásd Robbins-tétel (https://en.wikipedia.org/wiki/Robbins%27_theorem) egyik iránya), azaz egy komponensen belül minden csúcs elér minden másik csúcsot. Tehát a csúcsoknak megfeleltetett városok r_i értéke a jelenlegi komponensre szorítkozva a lehető legnagyobb, és mindegyik egyenlő a komponens csúcsszámával.



A második (élforgató) gráfbejárás az 1-es csúcsból indult. A komponensek színezve, méretük és a csúcsaikból elérhető más csúcsok száma feltüntetve. Pirossal a feszítőfa élei, szaggatott pirossal pedig a visszaélek láthatóak.

Ha G_x komponensből az elvágó él irányítása miatt elérhető a G_y, akkor G_x összes csúcsa hozzáfér majd G_y összes csúcsához, viszont G_y sohasem fog hozzáférni G_x csúcsaihoz. Tehát G_x minden csúcsára igaz, hogy r_i értékeik minimum a két komponens csúcsszámainak összegével egyenlőek.

Mivel az eredeti gráf véges, ezért ez a fa is véges, tehát lesz minimum egy olyan csúcs (komponens), ahonnan nem indul ki irányított él (elvágó él). Ekkor nyílván ezek a komponensek adják a gráfban a keresett r_i minimum értékét, tehát úgy célszerű irányítani az elvágó éleket, hogy a legnagyobb csúcsszámú komponensbe vezessen az összes út.

Részletes megoldás

Tarjan-algoritmussal keressük meg az elvágó éleket, illetve az ezen élek kivételével keletkező komponensek közül a legnagyobbnak válasszuk ki az egyik csúcsát.

Most **forgassuk be a hidakat** és **alkossunk irányított köröket a komponensekben** úgy, hogy például ebből a kiválasztott csúcsból indítsunk egy mélységi bejárást. Az így kapott feszítőfa éleit aztán irányítsuk úgy, hogy az összes út a gyökérbe vezessen, azaz gyerekből szülőbe. A kapott visszaéleket pedig irányítsuk mindig szülőből gyerekbe.

A hidak ekkor biztosan jó irányban fognak állni, a komponenseken belül pedig mivel nincs elvágó él, ezért bármelyik két csúcs között fogunk tudni találni egy oda- és vissza élekből álló kört, tehát bármelyik komponensbeli csúcsból bármelyikbe el fogunk tudni jutni.

Komplexitás

Műveletigény

Mivel a megoldás két mélységi gráfbejárást használ (egyet a Tarjan-algoritmussal, egyet pedig az élek beforgatása), ezért a műveletigény $O(N + M)$.

Memóriaigény

Az éleket egy tömbben, a gráfot pedig éllistás formában tároljuk. Tehát adódik az $O(N + M)$ memóriaigény.

Implementáció

A Tarjan-algoritmusnál többféleképpen meg lehet határozni a komponensek méretét és ez alapján a legnagyobbnak egy csúcsát. Az itt feltüntetett implementáció a rekurzív függvényhívások visszatérési értékével teszi ezt meg.

Egy implementáció C++ nyelven megtalálható itt (https://pastebin.com/sBwwQpdT).