Dinamikus gráfok II

az éltörlés visszavág

Iván Szabolcs

2016 ősz

Recap: probléma

A megoldandó problémák alakja általában:

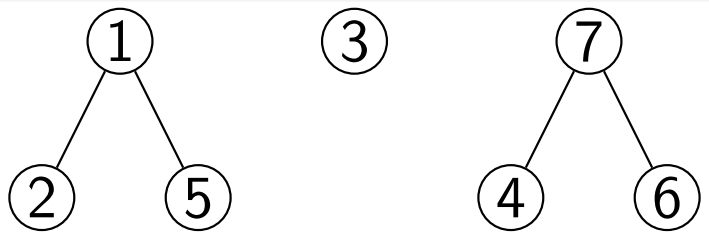
- kiindulunk egy G gráfból, erre kapjuk utasítások egy (hosszú) sorozatát
- egy utasítás lehet insert(u, v), erase(u, v) vagy query(u, v)
- az insert beszúrja, az erase törli az (u, v) élt, ők az update utasítások
- most: a query(u, v) azt kell megválaszolja, hogy u-ból v elérhető-e
- most: a gráfok irányítatlanok
- a csúcsok száma n, az éleké m, a parancsoké q
- ha a ,,szokásos" módon felépítjük a gráfot és mindig ,,from scratch" újraszámoljuk a queryt, akkor az időigény $O(q \cdot (n + m))$
- ennél szeretnénk jobbat

Recap: adatszerkezetek

Union-Find forest (inkrementális)

- Egy N-elemű halmaz egy partícióját (osztályozását) tárolja
- Támogatott műveletek:
  - create(n): létrehozza az $\{\{1\}, \{2\}, \ldots, \{n\}\}$ partíciót
  - find(u): visszaadja az u-t tartalmazó osztály egy reprezentánsát
  - union(u, v): egyesíti az u-t és a v-t tartalmazó osztályokat
- Időigények:
  - create(n): $O(n)$
  - find(u): amortizált $\alpha(n)$
  - union(u, v): amortizált $\alpha(n)$

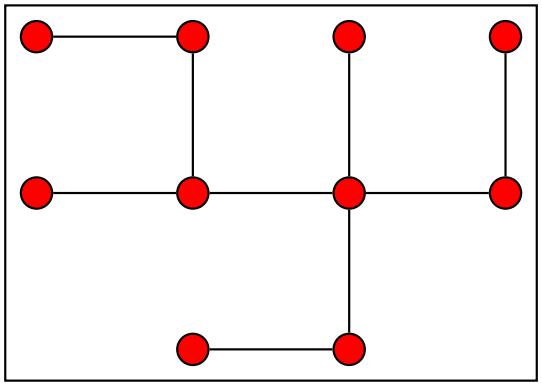
ahol $\alpha(n)$ az ún. „Ackermann-függvény” inverze, a gyakorlatban konstansnak is kezelhetjük, annyira lassan nő.

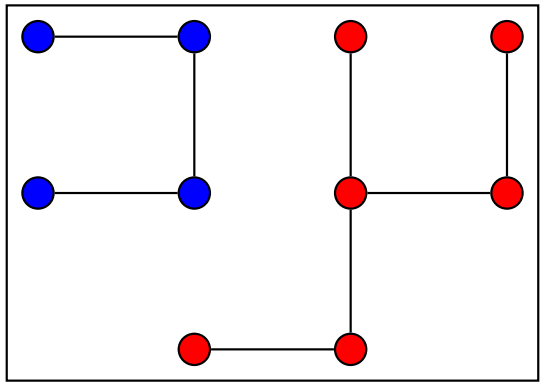


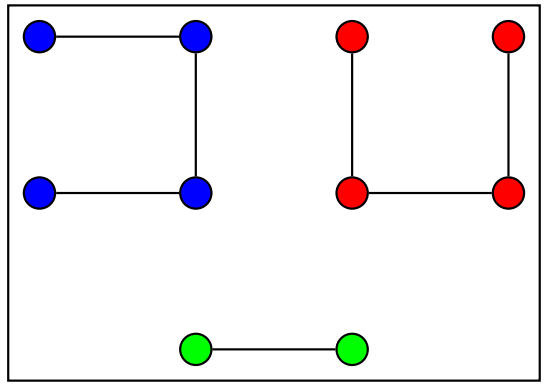
Recap: adatszerkezetek

Flood Fill forest (dekrementális)

- Egy n-csúcsú erdőt tárol
- Támogatott műveletek:
  - create(E): létrehozza az erdőt az E élekkel, $O(n+m)$ időben
  - find(u): visszaadja az u csúcs fájának unique ID-jét $O(1)$ időben
  - remove(u, v): törli az (u, v) élt az erdőből amortizált $O(\log n)$ időben



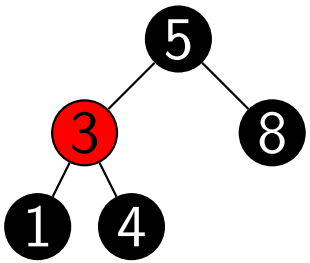


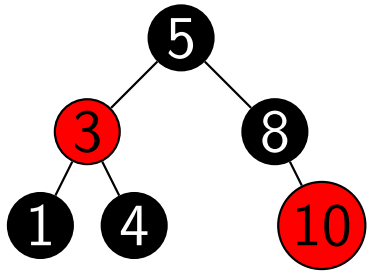


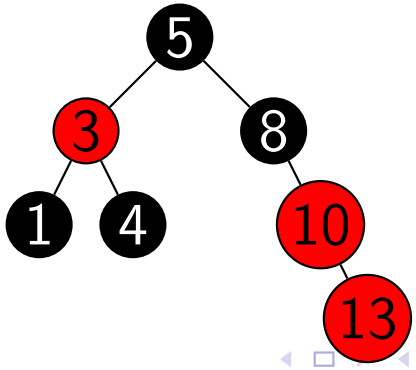
Recap: adatszerkezetek

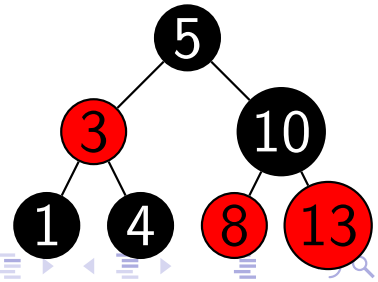
Piros-Fekete Fa (full) (C++ <set>,<map>, Java TreeSet)

- Kulcs-érték párokat tárol, ahol a kulcsokon van egy rendezés (pl. számok, stringek)
- Támogatott műveletek és időigényük n pár esetén:
  - init: létrehozza üresen, $O(1)$ idő
  - insert(K, V): beszúrja a (K, V) párt, $O(\log n)$ idő
  - remove(K): törli a K kulcsot és a hozzá tartozó értéket, $O(\log n)$ idő
  - min: visszaadja a minimális kulcsot a halmazban, $O(\log n)$ idő
  - next(K): a K után következőt a rendezés szerint, $O(\log n)$ idő
  - union(T_1, T_2): egyesít két fát, $O(\log n)$ idő
  - split(T, x): szétvág egy fát x-nél, $O(\log n)$ idő







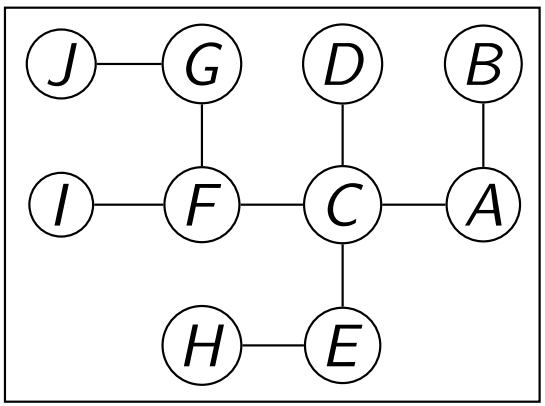


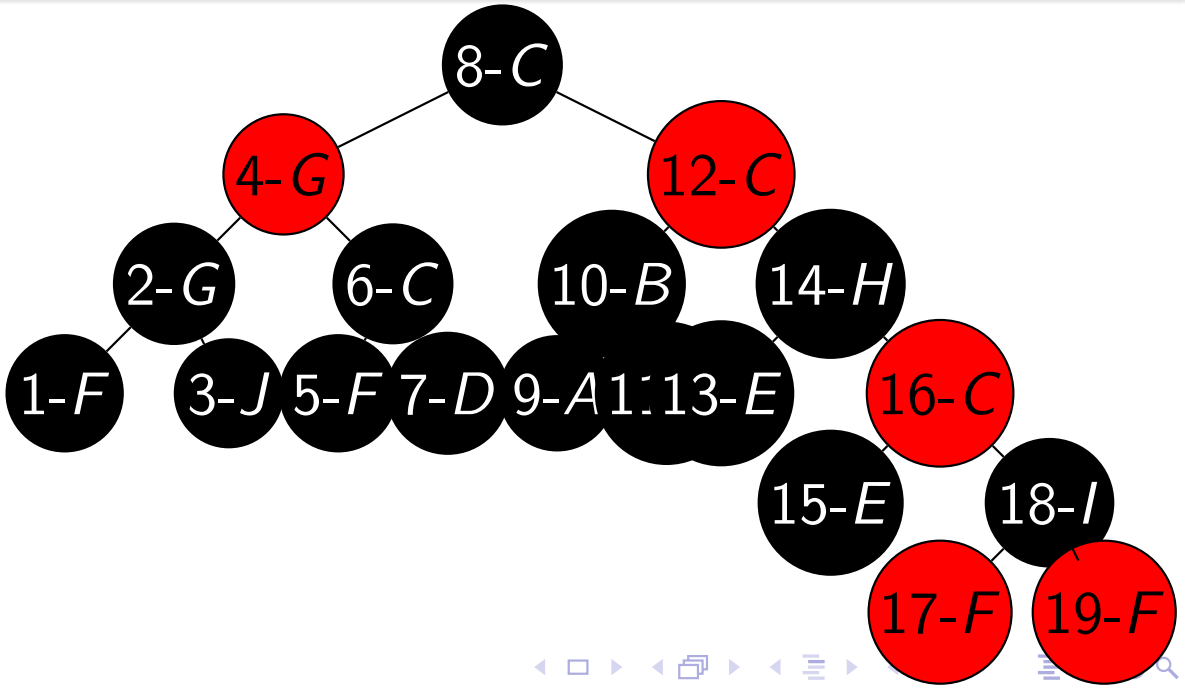
Recap: adatszerkezetek

Euler Tour Forest

- Egy n-csúcsú erdőt tárol
- Támogatott műveletek:
  - init: létrehozza üresen, $O(1)$ idő
  - insert(u, v): behúzza az (u, v) élt az erdő két fája közé $O(\log n)$ időben
  - remove(u, v): törli az (u, v) élt $O(\log n)$ időben
  - find(u): visszaadja u fájának egy reprezentánsát $O(\log n)$ időben

FGJGFCDCABACDCEHECF





Fully dynamic connectivity

- Input: egy iniciálisan üres G gráf
- Parancsok: insert(u, v), remove(u, v) és query(u, v)

Ötlet

- Nyilvántartjuk G egy feszítőerdőjét (ET fákkal, mint az erdős esetben is)
- Ekkor könnyű két erdő közé az insert: össze kell őket olvasztani
- Ekkor könnyű egy erdőben az insert: akkor a feszítő erdő nem változik
- Ekkor könnyű egy él törlése, ami nem az erdőben van: akkor a feszítő erdő nem változik
- Nehéz viszont egy erdőbeli él törlése!

Törlés feszítő erdőből

Ötlet

- Minden (i, j) élnek lesz egy $\ell(i, j)$ szintje.
- Egy él szintje 1 és $\log N$ közti egész szám lesz.
- Egy él szintje az idővel csak csökkenhet majd.
- G_i: a G-nek az a részgráfja, melyben a legfeljebb i szintű élek szerepelnek.
- F_i: a G_i-nek egy feszítőerdője.

Invariánsok

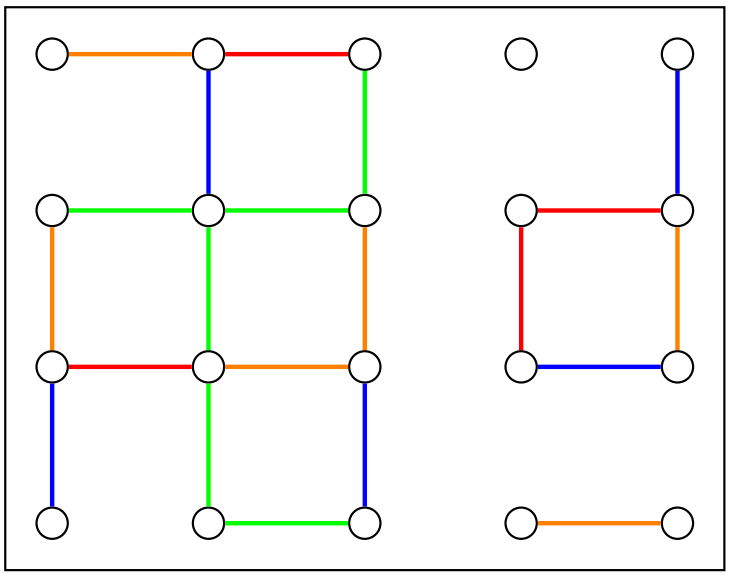
Az algoritmus során végig fenn fogjuk tartani a következő tulajdonságokat:

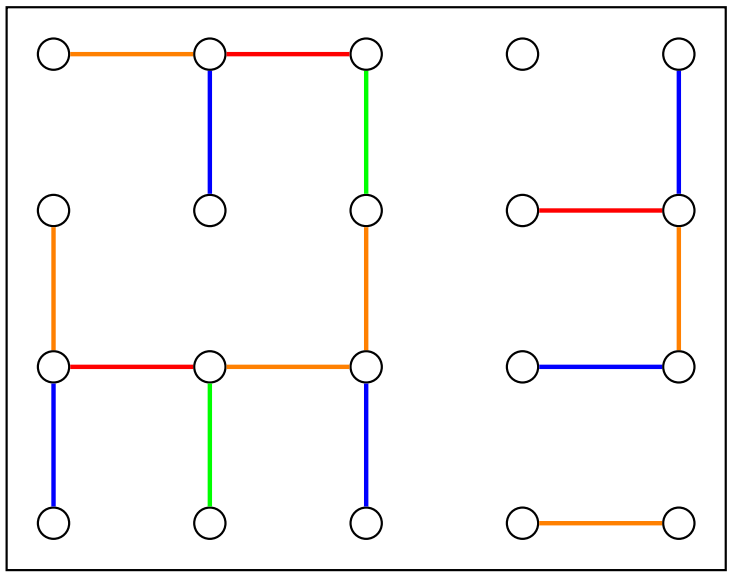
- A G_i gráf minden komponense legfeljebb 2^i méretű.
- $F_1 \subseteq F_2 \subseteq \ldots \subseteq F_{\log N}$.

Ezt az utóbbit úgy is mondhatjuk, hogy $F_i = G_i \cap F_{\log N}$, és $F_{\log N}$ pedig egy minimális feszítőfája G-nek (ahol a szintek az élsúlyok).

Példa

$n = 20$ csúcs, az élek szintjei egy, kettő, három vagy négy. (lehetne még 5 is, olyan él itt épp nincs.)
A kék komponensek mérete legfeljebb 2, a kék-narancs komponenseké legfeljebb 4, a kék-narancs-piros komponenseké legfeljebb 8, stb.





Törlés feszítő erdő-hierarchiából

- A G_i gráfok szomszédsági mátrixait is tároljuk.
- insert(i,j): beszúrjuk az (i,j) élt, $\ell(i,j) := \log N$.
  Ha i és j az $F_{\log N}$-nek különböző fáiban vannak, akkor egybe mergeljük a két fát és behúzzuk $F_{\log N}$-be az (i,j) élt. Ez $O(\log N)$ idő.
- query(i,j): a két csúcs akkor van egy komponensben, ha $F_{\log N}$-nek egy fájában vannak. Ez $O(\log n)$ idő.
- delete(i,j):
  - Kivesszük az (i,j) élt (G szomszédsági listáiból és a G_i-k szomszédsági mátrixaiból).
  - Ha (i,j) nincs benne $F_{\log N}$-ben, nincs gond, megyünk tovább.
  - Ha igen, akkor benne van az összes F_k-ben, amire $k \geq \ell(i,j)$. Töröljük ki belőlük.
  - Ezek után keresünk egy élt, amivel össze tudjuk kötni az i és j fáját.

Csere-él keresése

- Mivel minden F_k egy minimális feszítőerdő, így $\ell(i,j)$-nél kisebb szintű éllel nem tudjuk összekötni (i,j) helyett a széteső fákat.
- Tehát a $k = \ell(i,j), \ldots, \log N$ szintű élek közt keresünk összekötő élt, ilyen sorrendben.
- Legyen T_i és T_j az i-t és j-t tartalmazó fa F_k-ban úgy, hogy $|T_i| \leq |T_j|$.
- Mivel eredetileg az invariáns miatt $|T_i| + |T_j| \leq 2^k$, így $|T_i| \leq 2^{k-1}$.
- Járjuk be az összes olyan (x,y) élt, amire $x \in T_i$ és $\ell(x,y) = k$.
- Ha $y \in T_j$, akkor adjuk hozzá az (x,y) élt a $T_k, \ldots, T_{\log N}$ fákhoz és végeztünk.
- Különben csökkentsük $\ell(x,y)$-t eggyel.