⇒ a-ból b-be el lehet jutni v-n keresztül

gráf: erősen öf. komponensek, két komp. között csak 1 irányban mehet él; komponensek gráfja (csúcsok = komponensek) = dag

• Algo. erősen öf. komponensek megtalálására

G-ben DFS

G_{ford}-ban DFS: ∀ fát a még bejáratlan pontok közül a legnagyobb befejezési számúval kezdünk.
↑
1. bejárásból származó

Á: Az erősen öf. komponensek a 2. bejárás fái.
/nem biz./

→ O(n+e)

Minimális költségű feszítőfa

G(V,E) irányítatlan, egyszerű, összefüggő, súlyozott
c: E → ℝ

cél: F ⊆ E keresése: F feszítő fa és súlya minimális

04.22.

• Min. (költségű) feszítőfa

Adott: G = (V,E), irányítatlan, összefüggő, egyszerű,

élsúlyok: c: E → ℝ

Kell: F ⊆ E: F feszítőfa (öf. körmentes részgráf, amely ∀ csúcsot tartalmaz) & F súlya min.

• Piros-kék algoritmus

kék-szabály: X ⊆ V, (X ≠ ∅, X ≠ V), ha X-ből nem megy (V−X)-be kék él, akkor egy legkisebb súlyú kimenő színtelen élet kékre színezzük

piros-szabály: Ha egy C körben még nincs piros él, akkor egy legnagyobb súlyú színtelen élét pirosra színezünk.

→ Algoritmus: kezdetben G $\forall$ éle színtelen

a kék és piros szabályt alkalmazzuk tetsz. sorrendben,

tetsz. választással

Vége: ha egyik sem használható.

• T: Az algo. végén a kék élek egy min. feszítőfát alkotnak

B: S: színtelen élek

K: kékek P: pirosak

Kezdetben $S = E$, $K = P = \emptyset$

$E = S \cup K \cup P$ felbontás takaros színezés, ha van olyan

F min. feszítőfa G-ben: $F \supseteq K$, $F \cap P = \emptyset$

lemma

L1: az algo. $\forall$ lépése után takaros a színezés

L2: az algo. végén $S = \emptyset$

Ezekből a tétel biz.-a: végén $S = \emptyset \Rightarrow E = K \cup P$ illetve

L1 szerint ez a színezés takaros $\Rightarrow \exists F$ min. feszítőfa:

$F \supseteq K$, $F \cap P = \emptyset \Rightarrow F = K$ ✓

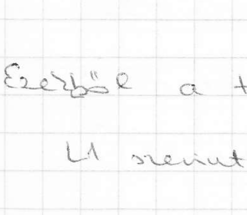
B (L1): kezdetben $E = S$ - takaros ✓
(indukció)

utána $E = S \cup K \cup P$ takaros $\rightarrow \exists F$ min. feszítőfa: $F \supseteq K$, $F \cap P =$

1. eset: kék szabályt használjuk $X \subseteq V$ halmazra, $g \in E$ élet

színezzük kékre → ha $g \in F$: akkor F továbbra is jó → színezés

takaros ✓   → ha $g \notin F$: a fában van út g végpontjai



között: ennek van olyan éle, ami X és $V-X$

között megy. Legyen g' ilyen $\Rightarrow F' = F - \{g'\} \cup \{g\}$

is feszítőfa.

g' színe: nem piros mi $g' \in F$ és nem kék a kék-szabály

miatt $\Rightarrow g'$ színtelen $\xRightarrow{\text{kék-szabály}} c(g') \geq c(g) \Rightarrow c(F) \geq c(F') \Rightarrow$

$c(F) = c(F')$

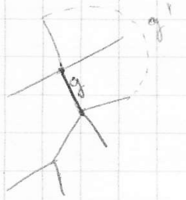
F min. volt

⇒ F' min. feszítőfa, ami mutatja, hogy az új színezés is takaros

2. eset: piros-szabályt használjuk a C körre és a $g \in E$ élét színezzük pirosra

Ha $g \notin F$, akkor F továbbra is jó – takaros ✓

Ha $g \in F$, akkor van olyan $g' \in C$: $F - \{g\} \cup \{g'\}$ is feszítőfa.

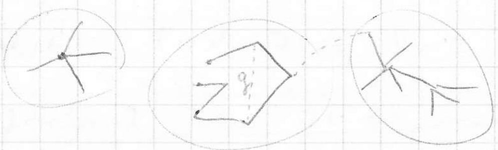


g' színe: nem piros (piros-szabály miatt), nem kék, mert $g' \notin F \Rightarrow g' \in S \xRightarrow{\text{piros szab.}} c(g') \leq c(g) \Rightarrow$

$c(F') \leq c(F) \xRightarrow{F \text{ min. volt}} c(F') = c(F)$ min. feszítőfák ⇒ tehát az új színezés is takaros ✓

B(L2): Legyen $g \in S$

A kék élek mindig feszítőerdőt alkotnak



Ha g egy kék fán belül van, akkor g + néhány fabeli él kört alkot → piros szabály alkalmazható

Ha g két kék fa között ⇒ $X :=$ egyik kék fa pontjai → kék szabály alkalmazható

⇒ algo. végén $S = \emptyset$ ✓

Megj: Ha már van $|V| - 1$ kék él, akkor abbahagyhatjuk, ezek már feszítőfát adnak

- Alkalmazás

JARNIK-PRIM algoritmus

$s \in V$; $X = \{s\}$ erre kék-szabály

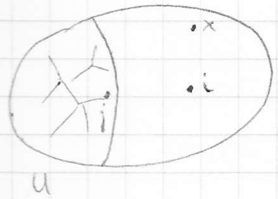
kék élek által elért pontok = X → erre is kék-szabály

A piros-kék tételből köv., hogy ez jó.

ZH: A–J : ChMax , K–L: StNagy , M–S: Ko51 , Sz–Zs : E.I.B

U: kész fa pontjai

$$\text{KÖZEL}[i] = \begin{cases} * & \text{ha } i \in U \\ j & j \in U,\ c(i,j) \text{ min ha } i \notin U \end{cases}$$



kész szabály alkalmazásakor szóba jövő élek

$\{i, \text{KÖZEL}[i]\}$ $i \notin U$

választás: ezek közül min

Utána: KÖZEL frissítése

$x \notin U$ KÖZEL[x] ha $c(i,x) < c(x, \text{KÖZEL}[x])$ akkor KÖZEL[x] = i

KÖZEL[i] = *

Lépésszám: $\underbrace{(n-1)(n-1)}_{\text{minimumok}} + (n-1)\,O(n) = O(n^2)$

- kupaccal: U és V–U közötti élekből (+ további maradékok)

MINTÖR által adott élről ellenőrizni kell, hogy U és V–U között megy-e. Ha nem: újabb MINTÖR

(Ha mi kerülne be kezül és nem kell → kidobjuk, azon nem előre távolítanánk mi. kupacból közepéről nem igazán tudunk)

kupac előállítás megadásánál:

$O(e) + 2e \cdot O(\log e) = O(e \cdot \log e) = O(e \cdot \log n)$

BESZÚR
MINTÖR

- <u>BORŮVKA</u> – algoritmus:

Kezdetben $\forall$ csúcs egy kész fa.

Minden kész fából kivezető legkisebb élet kékre színezi.

(baj van, ha vannak azonos súlyú élek → lehet kör: egyszerűen javítható (pl. élsúlyok $\frac{1}{1000}$-del való megvált. stb.))

$\forall$ menetben feleződik a kész fák száma → $\log n$ menet

→ $O(e \cdot \log n)$

• **KRUSKAL**-algoritmus : Növekvő sorrendben az éleket kékre színezzük, ha lehet, azaz nem hoz ~~létre~~ létre kék kört.

→ kék vagy piros szabály. → rendezés vagy kupac → $O(e \cdot \log e) = O(e \cdot \log n)$

Kialakul-e kör?

→ Adatszerkezet : unió-holvan : X alaphz.

$A_1, \dots, A_k \subseteq X$: $A_1 \cup \dots \cup A_k = X$, $A_i \cap A_j = \emptyset$ ha $i \neq j$

2 művelet:

1) **UNIÓ** (i,j) : A_i és A_j helyett $A_i \cup A_j$ lesz

2) HOLVAN $(x) = i$, ha $x \in A_i$ $(x \in X)$

Kruskal-algo : rendezés + élenként 2 db HOLVAN + $(n-1)$ db UNIÓ

Az unió-holvan megvalósítása :

(1) tömbbel $T[x] = i$, ha $x \in A_i$ → HOLVAN : $O(1)$
UNIÓ : $O(n)$ $n = |X|$

⇝ Kruskal : $O(e \cdot \log n) + e \cdot O(1) + (n-1) O(n) = O(e \log n + n^2)$

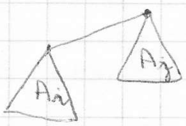
(2) fákkal : A_i → gyökeres fa

x elem → csúcs ; i → gyökér (név)

HOLVAN : csúcsból gyökérbe felmegyünk, lépésszám:

O (magasság)

UNIÓ :



Ha $|A_j| \geq |A_i|$ → $O(1)$

Kritikus út : leghosszabb út

→ tetsz. gráfban nehéz a leghosszabb utakat (vagy hosszukat) megtalálni, de dagra igen könnyű.

7.35.

$c_1, \dots, c_n$ egy top. sorrend

$H(i)$: hány út vezet c_1-ből c_i-be

$H(i) = \sum_{(j,i) \in E} H(j)$

$O(n+e)$

7.33. HF
7.39.
7.40

A': Ha kezdetben $\forall$ A_i 1 elemű $\Rightarrow$ a magasságuk mindig $O(\log n)$.

B: Hányszor távolodik a gyökértől: ahányszor beleolvad egy legalább akkora halmazba. A távolodás mindig 1.

Ilyenkor a halmaz mérete duplázódik $\rightarrow$ $\log n$ ilyen lehet

Kruskal:

$$O(e \cdot \log n) + e \cdot O(\log n) + (n-1)O(1) = O(e \cdot \log n)$$