Bonyolultságelmélet gyakorlat – 10
Approximálás I.

Definíció

Egy *optimalizálási* (most: minimalizálási vagy maximalizálási) problémát úgy tekintünk, mint amiben minden egyes I inputhoz tartozik *lehetséges megoldások* egy $S(I)$ halmaza, és minden $s \in S(I)$-nek adott egy $c(s)$ költsége, ami egy valós szám.

Maximalizálási problémánál $f(I) = \arg\max_{s \in S(I)} c(s)$, azaz egy maximum*hely* visszaadását várjuk el, minimalizálásinál pedig $f(I) = \arg\min_{s \in S(I)} c(s)$, egy minimumhely visszaadását várjuk.

Definíció

Egy A polinomidejű algoritmus *α-approximálja* az f optimalizálási problémát, ha minden I inputra $A(I) \in S(I)$, azaz mindig egy lehetséges megoldást ad vissza, és

- minimalizálási probléma esetén $c(A(I)) \leq \alpha \cdot c(f(I))$,
- maximalizálási probléma esetén $c(f(I)) \leq \alpha \cdot c(A(I))$,

azaz pl. maximalizálási problémánál egy 2-approximáló algoritmus legalább az optimum értékének felét eléri, minimalizálásinál egy 3/2-approximáló legfeljebb az optimális költség másfélszeresét költi el.

1. Feladat

Algoritmus a Súlyozott Csúcslefedés problémára:

Súlyozott csúcslefedés

- **Input**: egy $G = (V, E)$ gráf és minden $v \in V$ csúcsnak egy $c(v) > 0$ ára.
- **Output**: a G gráf egy $X \subseteq V$ lefogó csúcshalmaza.
- **Egy megoldás költsége**: az X halmaz ára $c(X) = \sum_{v \in X} c(v)$, a halmazbeli elemek árának összege.

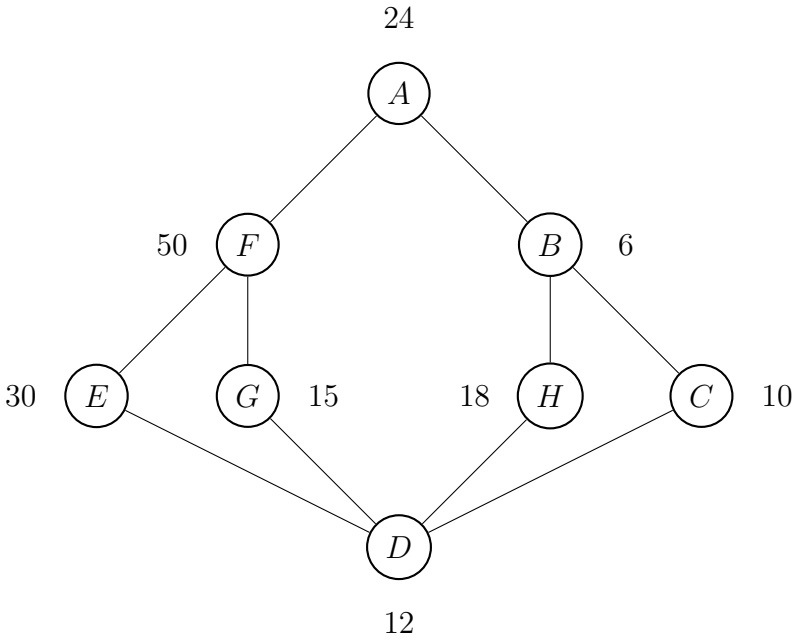
Mivel a $c(v) = 1$ *uniform* árakkal számolva ez épp a Csúcslefedés (mert akkor az X halmaz ára a mérete lesz, tehát ekkor a legkisebb lefogó csúcshalmazt keressük), így ez a probléma általánosabb, ezért szintén **NP**-nehéz.

Egy 2-approximáló algoritmus:

- Legyen $X = \emptyset$.
- Amíg van él G-ben:
  - válasszunk egy (u, v) élt (mindegy, melyiket). Legyen $c(u) \leq c(v)$.
  - Ha $c(u) = c(v)$, akkor vegyük be u-t is és v-t is X-be.
  - Ha $c(u) < c(v)$, akkor vegyük be u-t X-be, és csökkentsük $c(v)$-t $c(u)$-val.
  - Töröljük G-ből az újonnan bevett csúcs(ok)ra illeszkedő összes élt.
- Ha elfogytak az élek, adjuk vissza X-et.

Példa:

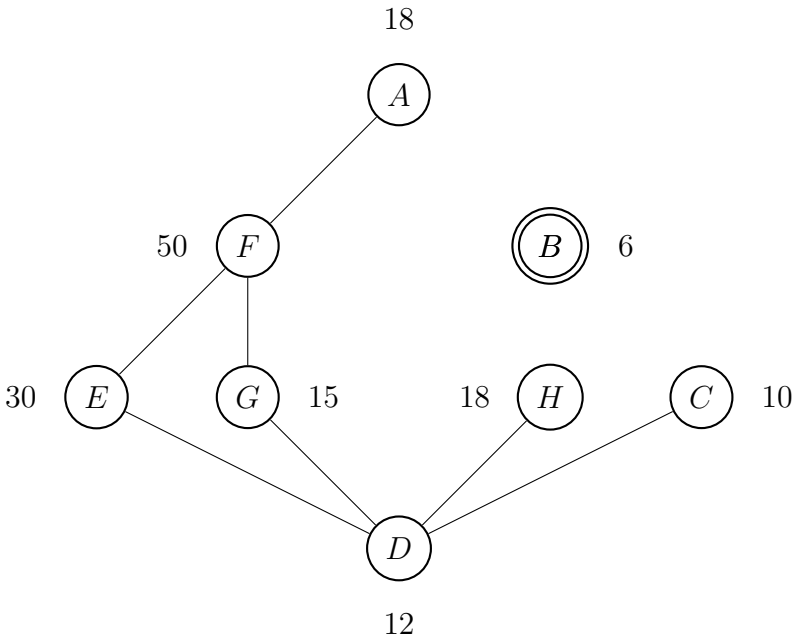
Nézzük ezt egy példán, a csúcsok mellé írva azok költségét:



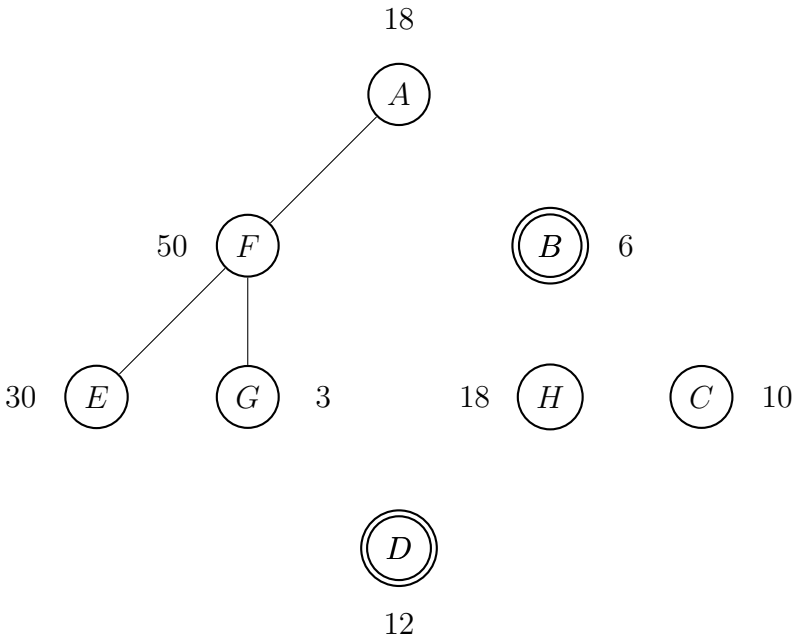
Megoldás

Ekkor az algoritmus egy futása lehet pl:

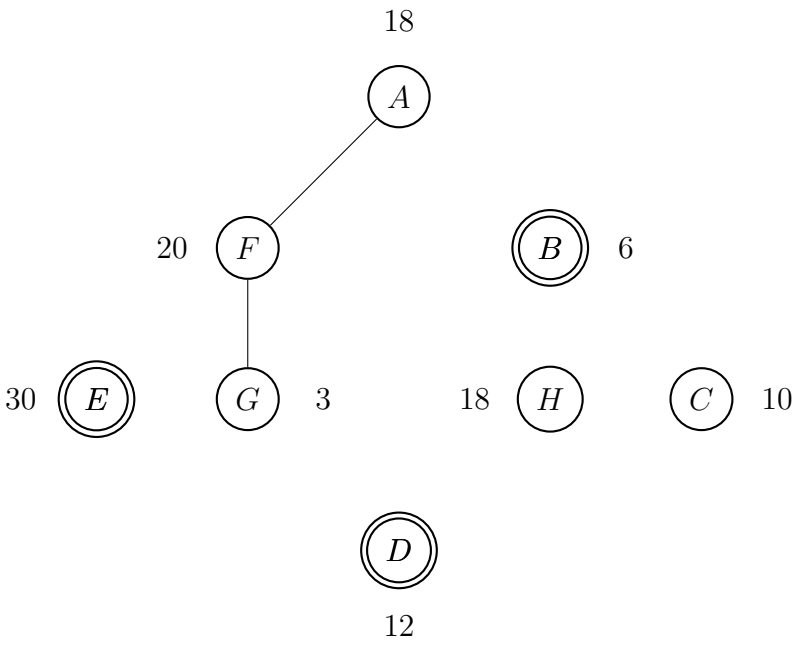
Vegyük az (A, B) élt, ennek a kisebb súlyú végpontja B. Bevesszük X-be B-t, A árát csökkentjük B árával, $c(A) = 18$ és töröljük az (A, B), (B, H), (B, C) éleket.



Ezután választhatjuk például a (D, G) élt, ennek D az olcsóbb végpontja, azt beválasztjuk X-be, G ára csökken 12-vel, töröljük a D-re illeszkedő éleket:



Most választhatjuk például az (E, F) élt, ennek E az olcsóbb végpontja, azt beválasztjuk, F ára csökken 30-cal, töröljük az E-re illeszkedő éleket:



Ha ezek után választjuk pl. az (A, F) élt, az A ára a kisebb, kiválasztjuk A-t, F ára 2-vé válik, töröljük az A-ra illeszkedő egyetlen (A, F) élt, majd az (F, G) élt választhatjuk csak, ekkor annak minimális árú végpontja már a 2 árú F, így azt választjuk.

Az algoritmus által szolgáltatott megoldás tehát: $X = \{A, B, D, E, F\}$, melynek összköltsége 122.

Mivel ez egy 2-approximáló algoritmus, ez azt is jelenti, hogy az optimum értéke legalább $122/2 = 61$. Ha egy másik algoritmussal megtaláljuk pl. a $\{B, D, F\}$ lefogó halmazt (aminek az ára 68), akkor ebből tudhatjuk, hogy annál ,,sokkal" már nem lehet lejjebb vinni a költséget.

Erre is jó lehet egy 2-approximáló algoritmus: mérni, hogy más algoritmusok "mennyire" mehettek közel az adott feladat optimumához.

2. Feladat

Algoritmus a HÁTIZSÁK probléma optimalizálási változatára:

HÁTIZSÁK

- **Input**: (w_i, c_i) pozitív egész (súly-érték) számpárok egy $i = 1, \ldots, n$ sorozata és egy W összkapacitás.
- **Output**: Válasszuk ki a tárgyak egy $I \subseteq \{1, \ldots, n\}$ részhalmazát, hogy az $\sum_{i \in I} w_i$ összsúlyuk legfeljebb W legyen, ezen belül pedig $c(I) = \sum_{i \in I} c_i$ összértékük (azaz az I megoldás haszna) maximális!

Töredékes hátizsák

A töredékes változatot (mikor is a tárgyak törhetőek, az értékük a törés arányában skálázódik) optimálisan oldja meg a *mohó* algoritmus:

- rendezzük a tárgyakat a $\frac{c_i}{w_i}$ fajlagos értékük szerint csökkenőbe
- az így kapott sorrendben rakjuk be a tárgyakat szép sorban, amíg csak beférnek. (Ha valaki nem fér be, nézzük meg a sorrendben utána következőket is, azok hátha beférnek.)

Ez *nem* konstans approximál: ha az input $(1, 1)$ és $(W, W-1)$, akkor az algoritmus az első tárgyat beteszi (mivel annak 1 a fajlagos értéke, a másodiknak pedig $\frac{W-1}{W} < 1$). Ekkor viszont nem marad hely a második tárgynak $\rightarrow$ az eredményben az összérték 1. Az optimumé pedig $W - 1$, ami nem konstansszor nagyobb...

Viszont azzá tehető úgy, hogy egy kis részén brute force-olunk az inputnak, majd a maradékot mohóval töltjük fel. Legyen $k > 0$ egy egész.

Algoritmus:

- Minden olyan legfeljebb k méretű I részhalmazára a tárgyaknak, ami befér a zsákba, számoljuk ki a következő értéket:
  - az I-beli összértéket,
  - plusz azt az összértéket, amit a hátizsák *maradék részének* a *maradék tárgyakból* mohó módon való feltöltésével kapjuk.
- Amelyik a legnagyobb, azt az elrendezést választjuk.

Ez $k = 1$-ben azt jelenti, hogy ahányféleképp csak lehet, kinézünk egy tárgyat, amit mindenképp elviszünk és a többi üres helyre a többi tárgyból mohó módon, fajlagos érték szerint csökkenőbe rakunk értékeket.

Az algoritmus-család egyébként $1 + \frac{1}{k}$ approximáló.

Példa:

	A	B	C	D
Súly	100	10	50	50
Érték	150	20	55	50

és $W = 100$.

Megoldás

Ekkor a mohó szerinti sorrend $\frac{20}{10} \geq \frac{150}{100} \geq \frac{55}{50} \geq \frac{50}{50}$.

Lehetséges esetek:

1. Ha a $\frac{20}{10}$-et vesszük be, akkor a $\frac{150}{100}$ nem fér be, a $\frac{55}{50}$ igen, az $\frac{50}{50}$ megint csak nem, ez összesen 75 haszon és $I = \{B, C\}$.
2. Ha a $\frac{150}{100}$-at vesszük be, amellé már semmi más nem fér, ekkor a haszon 150 és $I = \{A\}$.
3. Ha az $\frac{55}{50}$-et, akkor ha ezt bevesszük, még befér mellé a $\frac{20}{10}$, de más nem, ekkor 75 a haszon és $I = \{B, C\}$.
4. Ha az $\frac{50}{50}$-et, akkor ha ezt bevesszük, még befér mellé a $\frac{20}{10}$, de más nem, ekkor 70 a haszon és $I = \{B, D\}$.

A megoldásunk ez esetben $I = \{A\}$, amihez tartozó haszon 150.

A $k = 1$ miatt ez 2-approximál, vagyis az optimum legfeljebb kétszer ennyi lehet.