Algoritmuselmélet és bonyolultságelmélet feladatok
MSc hallgatók számára

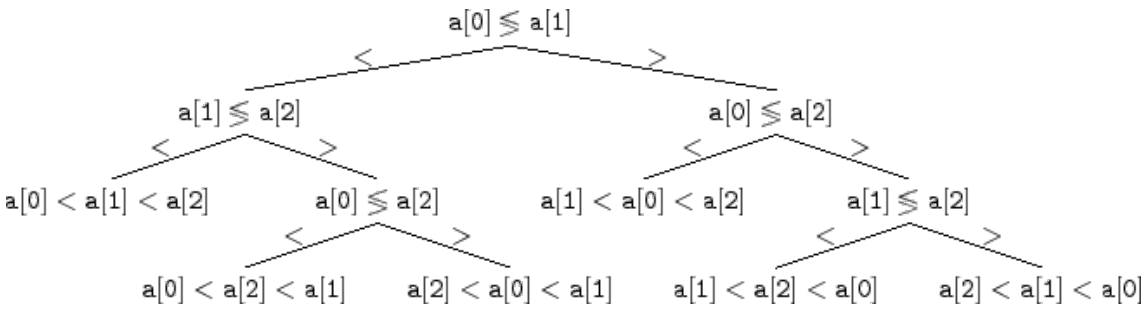
Alapok, Dinamikus programozás

2017. Előadó: Hajnal Péter

Definíció. Egy összehasonlító-fa inputja $x_1, x_2, \ldots, x_n$ valós számok (feltesszük, hogy egyenlő értékéknél van egy döntetlen feloldó stratégián, azaz bármely két elem közül az egyik nagyobb mint a másik; ekvivalems módon feltesszük, hogy az inputszámok különbözőek). A fa egy gyökres bináris fa (minden csúcsnak 2 vagy 0 gyereke van, 0 gyerek *equiv* (gyökeresfa) levél). A nem levél csúcsokban egy „$x_i?x_j$" kérdés/teszt ($i \neq j$) szerepel. A két gyerekhez vezető élen $< / >$ címkék találhatók. A levelek a kiszámított információt tartalmazzák (ez lehet a minimális érték indexe, a medián indexe, a rendezett sorrend index-sorozata, ...).

Minden input érték a gyökérben megkezd egy lefelé (a levelek irányába) történő utat. A gyökérben megnézzük az ottani teszt eredményét és ennek megfelelően lép az input az egyik levezető élen az egyik gyerekbe. Az ott található teszt alapján lép tovább, amíg egy lvélbe nem jut. Az ott lévő információ amit a fa/algoritmus kiszámolt.

Egy összehasonlító-fa megold egy problémát (példaul: maximum meghatározás, minimum meghatározás, medián meghatározás, rendezés, ...), ha minden inputnál a számítási út végén kapott válasz a probléma helyes megoldása.



A fa legrosszabb analízis alapján vett bonyolultsága a fa mélysége (leghosszabb gyökér/levél út hossza).

1. Feladat. *Milyen mélységű a buborék rendezést leíró összehasonlító-fa?*

2. Feladat. *Milyen mélységű az összefésülő rendezést leíró összeshasonlító-fa?*

3. Feladat. *Igazoljuk, hogy minden rendezést elvégző összehasonlító-fa mélysége* $\Omega(\log n)$. *(Azaz legalább* $\alpha \cdot \log n$, *alkalmas* $\alpha > 0$ *konstasra).*

4. Feladat. *A buborék rendezés alapján adjunk összehasonlító fát, amely meghatározza a maximális elemet. Mi ennek a mélysége?*

5. Feladat. *A USOpen főtáblájának szervezése alapján adjunk egy másik összehasonlító-fát, amely meghatározza a maximális elemet. Mi ennek a mélysége?*

6. Feladat. *(i) Egy n elem közül a maximális elemet meghatározó összehasonlítási-fa legalább $n-1$ összehasonlítást végez minden számítási úton.*

(ii) Egy n pontú fának $n-1$ éle van.
Formalizáljunk egy kapcsolatot köztük.

7. Feladat. *Megverhető-e a fenti két algoritmus? Ha igen, mutassunk egy hatékonyabbat; ha nem, akkor indokoljuk meg miért nincs jobb algoritmus.*

8. Feladat. *Vizsgáljuk a következő két problémát*

(A) Határozzuk meg a maximális és minimális elemet egyszerre.

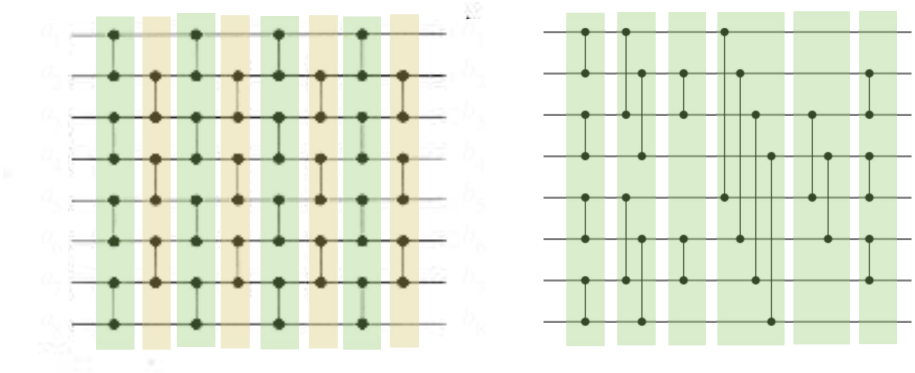
(B) Határozzuk meg a maximális és a második legnagyobb elemet egyszerre.

Mindkét probléma megoldható a korábbiak alapján a maximális elem meghatásozásával, majd a maradék elemek maximális/minimális elemének meghatározásával, ami $(n-1)+(n-2)=2n-3$ lépéssel megoldható. Van-e jobb algoritmus?

9. Feladat. *A fenti (A) és (B) feladatnál határozzuk meg a legkisebb mélységet, amivel rendelkező fa/algoritmus a két feladatot kezelni tudja.*

$\star$

Definíció. Rendező hálózat inputja $x_1, x_2, \ldots, x_n$ valós számok szám n-ese. A hálózatnak n vízszintes síne/huzala van különböző magasságokban. Az i-edik sín kezdetben x_i-t tartalmazza. Idővel az értékek a sínen balról jobbra haladnak. Az x koordináta (az idő) fázisokra van osztva. Minden fázisban a sinek egy párosítása adja az algoritmust. Az első fázisban az első párosítás minden sínpárjára a következő történik: a sínpár a fázis elején két értéket „szállít” (a két érték két különböző magasságban helyeződik el). A fázis során a nagyobb érték átkerül (ha szükséges mozgatás) a magasabb sínre, ezzel együtt a kisebb érték az alacsonyabb huzalon halad tovább.



1. ábra. Két példa rendező hálózatra. A fázisok színes téglalapokban találhatók. Az inputok a vízszintes síneken balról jobbra haladnak és minden fázisban (ha párosítottak) párjukkal rossz sorrend esetén sínt váltanak.

A hálózat rendező hálózat, ha minden inputra a hálózaton való futtatás után (a sínek bal oldali végén) fentről lefelé haladva az értékeket is csökkenő sorrendben látjuk.

10. Feladat. *A buborék rendezést valósítsuk meg hálózattal. Próbáljuk „tömöríteni" az algoritmust. Hány fázis szükséges?*

11. Feladat (0-1 **Lemma).** *Igazoljuk, hogy egy rendező hálózat akkor és csak akkor rendező hálózat, ha* 0-1 *értékű inputokat rendez.*

12. Feladat. *Páratlan fázis, egy olyan fázis ahol minden páratlan indexű (magasságú) s_i sínt s_{i+1}-gyel párosítunk (feltéve, hogy $i+1 \leq n$). Páros fázis, egy olyan fázis ahol ha minden páros indexű s_i sínt s_{i+1}-gyel párosítunk (feltéve, hogy $i+1 \leq n$).*

Egy hálózat felváltva páratlan/páros/páratlan/páros/páratlan/páros/... fázis hajt végre. Az $n = 8$ esetet az 1. ábra bal oldalán láthatjuk.

Igazoljuk, hogy páros n esetén, n fázist használva rendező hálózathoz jutunk.

Mi a helyzet pártalan n esetén?

13. Feladat. *Mit mondhatunk olyan rendező hálózat fázisainak számáról, ami párosításai csak szomszédos sínpárokat tartalmazhatnak?*

14. Feladat. *Az összefésülő rendezést megoldjuk a rendező hálózatok keretében. Tegyük fel, hogy $n = 2^\ell$ és a síneken eljutottunk oda, hogy a 2^s méretű sínblokkon ($2^{\ell-s}$ darab) az ott lévő számok rendezve vannak. Az egymás melletti blokkokat ($2^{s-\ell-1}$ darab pár) összefésüljük. Egy blokkpár tartalmazza a $b_1, b_2, \ldots, b_{2^\ell}$, $b_{2^\ell+1}$, $b_{2^\ell+2}$, ..., $b_{2^{\ell+1}}$ számokat. $b_1, b_2, \ldots, b_{2^\ell}$ és $b_{2^\ell+1}, b_{2^\ell+2}, \ldots, b_{2^{\ell+1}}$ egy-egy blokk, azaz rendezettek. A dupla méretű teljes rendezett sor kialakításához a hálózat a következőket végzi el: $b_1, b_3, \ldots, b_{2^\ell-1}, b_{2^\ell+1}, \ldots, b_{2^{\ell+1}-1}$ (páratlan indexű) és b_2, b_4, ..., b_{2^ℓ}, $b_{2^\ell+2}$, ..., $b_{2^{\ell+1}}$ (páros indexű) számokat rekurzívan rendezi. Mindkét esetben az input egy-egy rendezett féllel rendelkezik, amit össze kell fésülni. A két rendezés/összefésülés fázisai közös fázisokban végezhetők el. Majd egyetlen plusz fázis következik, amelyben a párosítás a következő: b_2b_3, b_4b_5, ..., $b_{2^{\ell+1}-2,2^{\ell+1}-1}$. Az algoritmus $n = 8$ esetén az 1. ábra jobb oldalán látható.*

Igazoljuk, hogy a plusz egy fázis után a blokkpár teljesen rendezett, azaz az algoritmus korrekt.

Hány fázis kell $n = 2^\ell$ szám rendezéséhez?

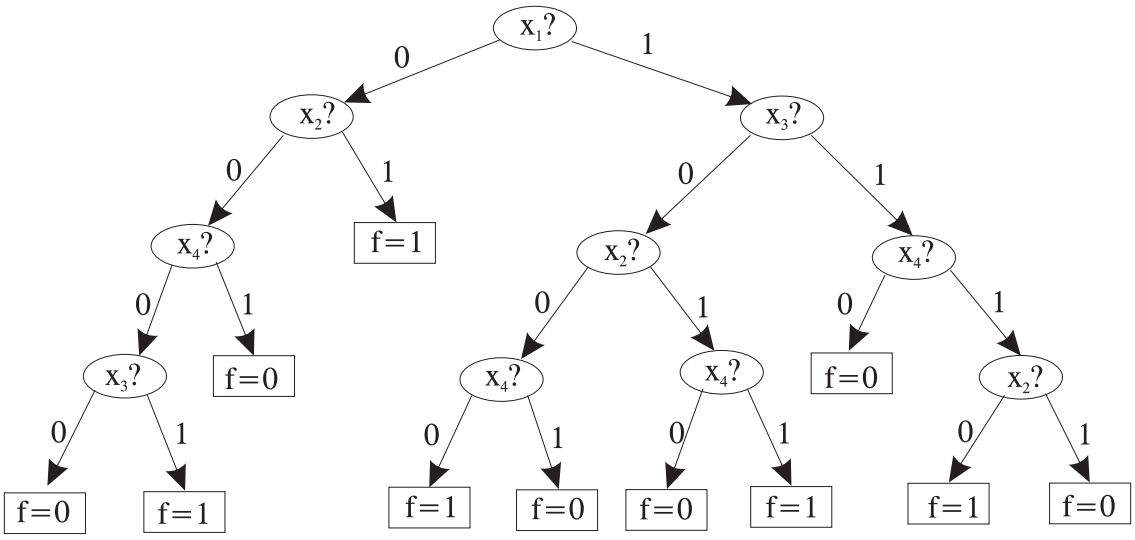
15. Feladat. *Minden n szám rendezésére alkalmas rendező hálózat $\Omega(\log n)$ fázisú.*

⋆

Definíció. $x_1, x_2, \ldots, x_n$ változókat használó $f(x_1, x_2, \ldots, x_n) : \{0,1\}^n \to \{0,1\}$ Boole-függvény kiszámítására szolgáló *döntési-fa* egy D gyökereztetett bináris fa. A fa levél csúcsaihoz egy-egy címkét rendelünk, amely a $\{0,1\}$ halmaz egy eleme. A fa nem levél csúcsaihoz egy-egy címkét rendelünk, amely az $\{x_1?, x_2?, \ldots, x_n?\}$ halmaz egy eleme. Minden élhez 0 vagy 1 címkét rendelünk úgy, hogy minden nem levél csúcs esetén a két gyerekhez kifutó két él egyikének 0, másikának 1 legyen a címkéje. A fa egy kérdezési eljárást ír le, amely során az egyes változók értékeit „gyűjtjük össze" (az aktuális csúcs címkéje az aktuális teszt/kérdés).

A döntési-fa gyökerének címkéje írja le a kérdezési eljárás első kérdését. A kérdésünkre 0 vagy 1 választ kapunk. Ez a gyökeret elhagyó két él címkéje. A két fiú és leszármazottjai a kérdezés további menetét írják le abban az esetben, ha az első válasz a hozzájuk vezető él címkéje. Így azon fiú címkéje, amelyhez a gyökérből a 0 címkéjű él vezet megadja azt a kérdést, amit az eljárás akkor jelöl ki, ha az első

kérdésre (ez a gyökér címkéje) adott válasz 0. Egy konkrét input esetén az eljárás követése a döntési fában pontosan egy a gyökérből egy levélbe vezető utat jár be. Ezt az utat nevezzük a konkrét inputhoz tartozó számítási útnak. A szokott geometriai ábrázolásban a számítási utak lefelé, egyre alacsonyabb szintre jutnak.



D egy f Boole-függvényt számol ki, ha minden $x_1, x_2, \ldots, x_n$ változók értékadására a megfelelő számítási út végső csúcsán (egy levélen) lévő címke $f(x_1, x_2, \ldots, x_n)$.

A D döntési-fa bonyolultsága a mélysége. Egy probléma döntési-fa bonyolultsága a problémát megoldó döntési fák/algoritmusok minimális bonyolultsága.

16. Feladat. *Egy döntési-fa kiszámolja, hogy páros vagy páratlan sok* 1*-es van-e az input bitsorozatban. Mit mondhatunk a fa bonyolultságáról, mit mondhatunk a fáról?*

17. Feladat. $f(x_1, \ldots, x_n, y_0, y_1, \ldots, y_{2^n-1})$ *Boole-függvény az első* n *bitet egy* i *természets szám kettes számrendszerbeli felírásának tekinti és kiadja* y_i*-t. Adjunk hatékony döntési fát, amely kiszámítja* f*-et.*

18. Feladat. *Egy* $g(x_1, x_2, \ldots, x_n)$ *Boole-függvény minden változójától függ (azaz minden* i*-re van olyan* $x^{(i)}$ *input, amelyet csak az* i*-edik bitben változtatva olyan* $\widetilde{x^{(i)}}$ *inputhoz jutunk, amelyre* $f(x^{(i)}) \neq f(\widetilde{x^{(i)}})$*. Adjunk alsó becslést* g *döntési-fa bonyolultságára.*

Definíció. Egy v pontú egyszerű gráfot $\binom{v}{2}$ bittel kódolhatunk: minden csúcspárnak megfelel egy bit, ami a csúcspár összekötöttségét írja le.

v pontszámú egyszerű gráfok tulajdonságai (amelyekről minden gráf esetén eldönthető, hogy igaz-e) azonosíthatók $\binom{v}{2}$ változójú Boole-függvényekkel.

19. Feladat. *Írjunk le egy döntési-fát, amely eldönti egy gráfról, hogy összefüggő-e (ÖSSZEFÜGGŐSÉG tulajdonság).*

20. Feladat. *Határozzuk meg az ÖSSZEFÜGGŐSÉG tulajdonság döntési-fa bonyolultságát.*

21. Feladat. *Írjunk le egy döntési-fát, amely eldönti egy gráfról, hogy van-e benne izolált csúcs (IZOLÁLT tulajdonság).*

22. Feladat. *Határozzuk meg az IZOLÁLT tulajdonság döntési-fa bonyolultságát.*

⋆ ⋆ ⋆

23. Feladat. *Adott egy konvex n-szög. Húzzunk be egymást nem metsző átlókat úgy, hogy ezek háromszögekre osszák a sokszöget (ezt a sokszög háromszögelésének nevezzük).*

(i) Hányféleképpen tehetjük ezt meg?

(ii) Tervezzünk algoritmust, amely megkeresi azt a háromszögelést, amelyben a kialakuló háromszögek kerületeinek összege minimális.

24. Feladat. *Egy táblázat elemei egész számok. A bal felső sarokból a jobb alsóba kell jutnunk le és jobb elemi lépésekkel (egy elemi lépésnél szomszédos elemre lépünk).*

(i) Hányféleképpen tehetjük meg ezt?

(ii) Tervezzünk algoritmust, amely megkeresi azt az utat, amely során érintett számok összege a lehető legnagyobb lesz.

25. Feladat. *Adott n szám sorozata. Keressük meg azt a folyamatos részsorozatát, amelyben az elemek összege maximális.*

Algoritmusunk lineáris-e? Ha nem, akkor javítsuk.

26. Feladat. *Tervezzünk algoritmust, ami egy adott számsorozatra meghatározza a leghosszabb monoton növő részsorozat hosszát.*

27. Feladat. *Adott pénzérmék egy sorozata. Egy barátunkkal játszunk. Felváltva lépünk és minden lépésünkben az egyik szélső érmét vessyük el (lépésünk az a döntés, hogy a két szélső érme közül melyiket választjuk). Tervezzünk algoritmust, amely meghatározza mi a legnagyobb összeg, amit a játék során összegyűjthetünk. (Ez két feladat aszerint, hogy ki kezd.)*

28. Feladat. *Adottak érme-névértékek. Ezek pozitív egészek és köztük van az '1' is. Adott egy n értékű papírpénz. Szeretnénk érmékre felváltani (mindegyik névértékből van elég érménk). Tervezzünk algoritmust, mely meghatározza meg mi a legkevesebb érme, amire fel tudjuk váltani pénzünket.*