Kazi Sándor
KAZI(AT)CS.BME.HU

Kiegyensúlyozott keresőfák - Hash - DFS

1. Illesszük be egy piros-fekete fába sorban a $8, 2, 4, 7, 5, 3, 1, 6, 9$ elmeket. Ezután töröljük az 5, majd a 7 elemet.
2. Melyik az a 10 pontú piros-fekete fa, aminek a legnagyobb a magassága?
3. Egy piros-fekete fában valamelyik, a gyökértől egy levélig vezető úton sorban az alábbi színű pontok vannak: fekete, piros, fekete, fekete. Mennyi a fában tárolt elemek számának a minimuma?
4. Igazolja, hogy egy 80 elemet tároló piros-fekete fában a gyökér magassága nagyobb, mint a gyökér fekete magassága.
5. Hogyan változhat egy piros-fekete-fában található piros csúcsok száma egyetlen elem beszúrásakor?
6. Egy k értéket tároló piros-fekete fának hány csúcsa lehet? Segítség: (Euler-tétel)
7. Egy piros-fekete fában az 1,4,7,10 elemek vannak. Hány forgatásra lehet szükség a 13 elem beszúrásához, ha a szükséges forgatások száma
   (a) a 0-t beszúrva 3
   (b) a 2-t és a 3-at ebben a sorrenben beszúrva összesen 1 lenne

---

8. Építsünk 2-3 fát a következő elemekből, ebben a sorrendben: D, B, E, A, C, F, G! Ezután töröljük D-t és B-t!
9. Ugyanezt csináljuk meg a B_4 fával.
10. Az $[1, 178]$ intervallum összes egészei egy 2-3 fában helyezkednek el. Tudjuk, hogy a gyökérben két kulcs van, és az első kulcs a 17. Mi lehet a második? Miért? (ZH vmikor régen)
11. Az S_1 és S_2 kulcshalmazokat kiegészített 2-3-fákban tároljuk. Ezek az eredeti 2-3-fától abban különböznek csak, hogy minden csúcsban fel van jegyezve az onnan induló részfa magassága. Tegyük még fel, hogy az S_1-beli kulcsok mind kisebbek az S_2-belieknél. Javasoljunk hatékony algoritmust a két fa egyesítésére. A cél tehát egy olyan kiegészített 2-3-fa, amelyben a kulcsok $S_1 \cup S_2$ elemei.
12. Egy 2-3 fába egymás után 1000 új elemet illesztettünk be. Mutassa meg, hogy ha ennek során egyszer sem kellett csúcsot szétvágni, akkor a beillesztések sorozata előtt már legalább 2000 elemet tároltunk a fában. (ZH 2003. márc. 31.)

---

13. Egy sportklub teniszezői kialakítottak egy erősorrendet. Ezt a következő szabályok szerint tartják karban: új játékos a sorrend végére kerül; a rangsor szerinti i-edik játékos kihívhatja az $i-1$-ediket; ha legyőzi, helyet cserélnek. Tervezzünk olyan hatékony adatszerkezetet, mely lehetővé teszi a rangsor számítógépes kezelését! A szükséges funkciók a következők: 1) BESZÚR(név): az új jövevényt a rangsor végére teszi; 2) KIHÍV(név): az $i-1$. helyen álló személy nevét adja, ha a "név"-vel azonosított személy az i. helyen áll és $i > 1$. 3) KICSERÉL(i): kicseréli a rangsor i. és $i-1$. helyén levő személyeket, ha $i > 1$. (fs: 4/31.)

A kezdetben üres $M = 9$ méretű hashtáblába a $h(x) = x \pmod 9$ hash-függvény segítségével az adott sorrendben rakja be a 4, 27, 18, 13, 9, 10, 30, 11 elemeket
(a) lineáris próbával;
(b) kvadratikus próbával.
(c) kettős hasheléssel ($h'(x) = x^2 - 1 \pmod 9$) Mindhárom esetben minden lépés után írja le a kapott tömböt és jelölje, hogy az aktuális elem hol okozott ütközést.

14. Mi a baja az $f(K) = K^2 \pmod 7$ hash-függvénynek, ahol 7 a táblaméret?
15. Az 1 és 91 közötti összes 3-mal osztható egész számot valamilyen sorrendben egy M méretű hash-táblába raktuk a $h(x) = x \pmod M$ hash-függvény segítségével, lineáris próbával. Ennek során hány ütközés fordulhatott elő, ha $M = 35$, illetve ha $M = 36$? (ZH 2008. 06. 03.)
16. (*Gondolkodtató:*) A kakukk-hash ötlete a következő. Adott egy M méretű tábla és két hash függvény (h_1 és h_2), melyek értékkészlete $\{0, 1 \ldots, M-1\}$. A cél, hogy bármely x elem vagy a $h_1(x)$, vagy a $h_2(x)$ sorszámú cellába kerüljön. Ehhez x beszúrása során a következő módon járunk el: ha a $h_1(x)$ vagy a $h_2(x)$ sorszámú cella üres, akkor elvégezzük a beszúrást, különben x „kilöki" ezen két cella közül az egyikből az ott lévő elemet (a két cella közül véletlenszerűen választva). Ha egy y elemet kilöknek az egyik hash függvény szerinti helyéről, akkor a másik hash-függvény által meghatározott helyére kerül, kilökve ezzel esetleg egy újabb elemet. (Az így kilökött elem is hasonlóan jár el.) Adjunk algoritmust, amely detektálja, hogy a tábla egy adott kitöltési helyzeténél van-e olyan elem, melynek beszúrásakor a fenti algoritmus sohasem áll le!

17. Előfordulhat-e nyitott címzéses hash-elés esetén, hogy az $n > 3$ méretű táblában csak 3 elem van, de a keresés lépésszáma n ? (vz: 2006.04.07./1.)

18. (kicsit *best-practice* feladat:) Szeretnénk minél hatékonyabban kezelni, de minél inkább ad-hoc a tankolásainkat. A cél, hogy utazás közben amikor jelez a üzemanyagszint-mérő, akkor meg tudjuk nézni, hogy milyen benzinkutak vannak 1 kilométeres körzetben. Javasoljunk adatstruktúrát, ami erre jó lehet.

19. Egy 10 méretű hash-táblába kettős hash-elést használva helyezzük el a következő elemeket: $12, 5, 20, 11, 25, 2$. A használt hash-függvény legyen $h(x) = 3x \pmod{10}$, a második hash-függvény pedig $h'(x) = (x \; mod \; 9) + 1$. (PPZH 2010. tavasz) Mi a baj a $h'(x)$ értékeivel?

20. Egy M méretű hash-táblába $n < M$ elemet raktunk be nyitott címzéssel, kvadratikus próbával. Ennek során t_1 ütközés történt, azaz ennyiszer kellett tovább próbálkoznunk – egy elem beszúrása során több ütközés is lehetett. Ugyanezt az n elemet ugyanabban a sorrendben beszúrtuk egy M^2 méretű hash-táblába is, de most lineáris próbával, $M \cdot h(x) + 1$ hash-függvénnyel, ekkor t_2 ütközés történt. Igazoljuk, hogy $t_2 \leq t_1$! (ZH 2010. tavasz)

---

21. A 6 pontú G gráf csúcsait jelölje x, y, z, u, v, w. A gráf egy mélységi bejárásánál a mélységi, ill. a befejezési számok a következők: x: 1,6; y: 2,4; z: 6,5; u: 3,3; v: 4,1; w: 5,2. Adjuk meg a bejáráshoz tartozó mélységi feszítőfa éleit. Rekonstruálható-e G az előző számok ismeretében?

22. Legyen G egy irányítatlan összefüggő gráf. Igaz-e, hogy
    (a) G minden f éléhez van G-nek olyan mélységi bejárása, amelyben f egy faél?
    (b) G minden f éléhez van G-nek olyan szélességi bejárása, amelyben f egy faél?
    (c) G minden F feszítőfájához van G-nek olyan mélységi bejárása, amelyben F minden éle faél?
    (d) G minden F feszítőfájához van G-nek olyan szélességi bejárása, amelyben F minden éle faél?

23. Tekintsük az olyan G irányított gráfokat, amelyekben ha eltekintünk az élek irányításától, akkor a kapott irányítatlan G' gráf összefüggő. A G gráf egy mélységi bejárásánál maximálisan hány olyan csúcs lehet, amelyre a mélységi és a befejezési szám megegyezik?

---

24. (logikai:) Egy vadász elindul délnek és megy 100 métert, utána keletre fordul és megy 200 métert, majd ismét 100 métert megy északnak. Ezzel visszaérkezett a kiindulási helyére. Ebben a pillanatban meglát egy medvét és lepuffantja szegényt. Milyen színű volt a medve?

25. (logikai:) 3 barát megszáll egy szállodában egy éjszakára. Fejenként fizetnek 10-10 dollárt, majd felmennek a szobájukba. Később a szállodatulajdonos rájön, hogy régi törzsvendégekről van szó, ezért visszaküld nekik 5 dollárt a szobaszervízzel. A három férfi meglepetve konstatálja a kapott összeget, de mivel nem tudják elosztani, ezért 1-1 dollárt tesznek csak el, a maradék kettőt visszadják a pincérnek. Tehát mindhárom férfi 9 dollárt fizetett a szobáért, a pincérnél van még 2, de ez csak 29. Hogy lehet?

26. (logikai:) Adott egy börtön, melyben a cellák hermetikusan elzártak, a cellák közt semmilyen kommunikáció nem lehetséges. Bekerül a börtönbe x db rab. A rabok együtt érkeznek és ismerik a börtön adottságait. A börtönőrök viszik sétálni a rabokat, naponta akár többet is, de teljes véletlenszerűséggel. Egy nap akár egy rabot többször is levisznek, de lehet olyan rab aki akár egy hónapig, vagy tovább nem sétál. Egyszerre egy rab sétál. Az udvaron van egy kapcsoló, melynek két állása van, A és B (a kapcsoló eredetileg az A állásban van). A rabok ezzel a kapcsolóval kommunikálhatnak: a séta alatt átkapcsolhatják. A börtöno rök nem nyúlnak a kapcsolóhoz. A rabok akkor szabadulnak ki, ha valamelyik rab kijeleni, hogy: ”Már minden rab volt legalább egyszer sétálni!” és a kijelentés IGAZ. Ha a kijelentés nem igaz, soha többé nem szabadulnak ki. Tehát csak egyszer lehet ilyen kijelentést tenni. Hogyan szabadulhatnak ki?

27. (programozás:) Egészítsd ki az alábbi függvénykezdeményt, hogy a két szám összegét adja. Csak ezeket a karaktereket használhatod: harc*&()?:
    int osszead(int a, int c){
    return ;
    }
    Nagy könnyítés: használható [és]
    Kisebb könnyítés: nem használható kettőspont