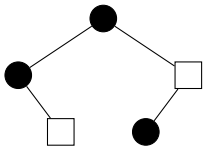
Piros-fekete fák, 2-3 fák, vödrös hash

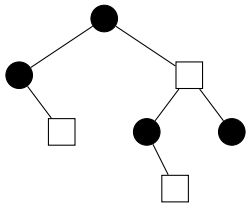
ALGORITMUSELMÉLET

8. gyakorlat

2024.

1. Lehetséges-e hogy az alábbi ábrákon egy piros-fekete fa csúcsait ábrázoltuk? (Az üres leveleket nem rajzoltuk fel, a fekete körök fekete csúcsokat, fehér négyzetek piros csúcsokat jelölnek.)





2. Egy piros-fekete fában a $2010, 42, 100\pi, 1848, 3$ elemeket tároljuk úgy, hogy a gyökérben levő elem a 42. Hogyan nézhet ki a fa? (Adja meg az összeset és indokolja meg, hogy más nincs.)

3. Lehetséges-e, hogy egy piros-fekete fából a preorder bejárás 15, 10, 70, 30, 20, 60, 85 sorrendben olvassa ki az elemeket? Ha igen, akkor adjuk meg az összes ilyen fát, majd az egyikbe szúrjuk be az 50-et.

4. 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 minimuma?

5. Határozzuk meg egy piros-fekete fa azon v csúcsait, melyek esetén a v csúcsú részfa is egy piros-fekete fa.

6. Egy 2-3 fa kezdetben csak a 6, 8, 13 elemeket tárolja. Rajzoljuk le ezt a fát, majd szúrjuk be a 2, 5, 1 elemeket, végül töröljük a 8, 2 elemeket.

7. Egy különböző egész számokat tároló 2-3 fa gyökerében két útjelző van, a 101 és a 117. Legfeljebb hány elemet tárolhat a fa?

8. 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?

9. Adott n darab intervallum $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$; az intervallumok végpontjai racionális számok és szokás szerint $a_i < b_i$. Tegyük fel, hogy egyik intervallum sem tartalmazza teljes egészében a másikat. Az intervallumokat a kezdő a_i koordinátájuk szerint szervezett 2-3 fában tároljuk. Hogyan lehet ennek segítségével egy adott x pontra $O(\log n)$ lépésben meghatározni, hogy a megadottak között van-e olyan intervallum, ami az x pontot tartalmazza?

10. Vödrös hash-t használva akarunk számokat tárolni, a $h(k) = k \pmod{11}$ hash függvényt használva.

    (a) Szúrjuk be a kezdetben üres, 11-méretű hash-táblába a $2, 5, 12, 1, 3, 88, 23, 43, 10, 34$ elemeket.

    (b) Hogyan zajlanak a feltöltött táblában KERES(1), KERES(20), KERES(45) műveletek?

    (c) Hogyan zajlik a TÖRÖL(23) művelet?

11. A vödrös hash-elésnél az egyes vödrök tartalmát lapok egy-egy láncolt listájában tároljuk. Legyen a vödörkatalógus mérete M és a hash-táblában tárolt lapok összes száma L.

    (a) Legrosszabb esetben nagyságrendileg hány lépést kell tennünk egy keresés során?

    (b) Legrosszabb esetben nagyságrendileg hány lépést kell tennünk egy beszúrás során?

    Most tároljuk az egyes vödrök tartalmát láncolt lista helyett egy-egy 2-3 fában.

    (c) Legrosszabb esetben nagyságrendileg hány lépést kell tennünk egy keresés során?

    (d) Legrosszabb esetben nagyságrendileg hány lépést kell tennünk egy beszúrás során?

12. Tervezzünk adatstruktúrát a következő feltételekkel. Természetes számokat kell tárolni, egy szám többször is szerepelhet, és a szükséges műveletek a következők.

    BESZÚR(i): az i szám egy újabb példányát tároljuk;

    TÖRÖL(i): az i szám egy példányát töröljük;

    MINDTÖRÖL(i): az i szám összes példányát töröljük;

    DARAB(i): visszaadja, hogy hány példány van az i számból.

    Az adatstruktúra legyen olyan, hogy ha m-féle elemet tárolunk, akkor mindegyik művelet lépésigénye $O(\log m)$.

13. Vázoljuk a 2-3 fának (és műveleteinek) egy olyan módosítását, amiben továbbra is van KERES, BESZÚR, TÖRÖL, MIN, MAX művelet, és ezeken kívül van még RANG és K-ADIK művelet is, ahol RANG(x) azt adja vissza, hogy a tárolt elemek között az x a rendezés szerint hányadik elem, a K-ADIK(i) pedig, hogy a rendezés szerint a tárolt elemek közül melyik az i-edik. A módosítás során a felsorolt szokásos műveletek lépésszámának nagyságrendje ne változzon, és mindkét új művelet lépésszáma legyen $O(\log n)$, ahol n a tárolt elemek száma.

14. Ha adott n szám, akkor hívjuk közülük középső elemnek a rendezés szerinti $\lceil n/2 \rceil$-ediket.

    Kezdetben adottak az $a_1, a_2, \ldots, a_n$ egész számok, amikről tudjuk, hogy az a_1 a középső elem, egyébként a számok rendezetlenek. Ezekből építsünk fel egy adatszerkezetet, amiben az alábbi két művelet van.

    BESZÚR: egy új elemet illeszt az adatszerkezetbe;

    KÖZÉPTÖR: az aktuális középső elemet törli.

    Mindkét művelet megvalósítása $O(\log k)$ összehasonlítást használjon, amikor k tárolt elem van, az adatszerkezet kezdeti felépítése legyen $O(n)$ összehasonlítás.