ANALIZA I DIZAJN ALGORITAMA II
zadaci sa vežbi

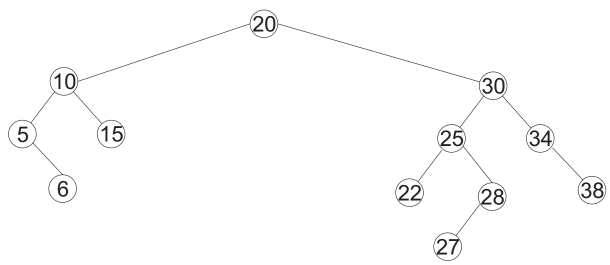
Vesna Pavlović

02. novembar 2010.

4 Strukture podataka - AVL stabla (nastavak)

[1]

1. Demonstrirati brisanje čvora 15 iz datog AVL stabla.



2. Neka su T_1 i T_2 dva proizvoljna binarna stabla sa po n čvorova. Dokazati da postoji niz od najviše $2n$ rotacija koje stablo T_1 transformišu u stablo T_2.

3. Neka je $S = \{s_1, s_2, \ldots, s_m\}$ vrlo veliki skup, izdeljen u k disjunktnih blokova. Pretpostavimo da nam je na raspolaganju procedura `Koji_blok`(s_i) koja za dati element s_i daje redni broj bloka koji sadrži s_i; ova procedura radi u konstantnom vremenu.

   Cilj je omogućiti rad sa malim podskupovima $T \subset S$ i to sledeće operacije:

   (a) `Umetni`(s_i)

   (b) `Obrisi`(s_i)

   (c) `Obrisi_blok`(j) - briše sve elemente koji pripadaju bloku j.

   Inicijalno je T prazan skup. Složenost svake od operacija treba da bude $O(\log n)$ u najgorem slučaju, gde je n tekući broj elemenata u T. Operacija `Obrisi_blok` samo uklanja elemente iz strukture podataka - nije

[1] Materijal je osmišljen na osnovu knjige: Algoritmi, M. Živkovića

neophodno fizičko brisanje svakog od tih elemenata. Oba broja, i m i k, mogu biti vrlo veliki pa se ne može koristiti tabela veličine m ili k. Medjutim n je relativno malo i na raspolaganju je prostor veličine $O(n)$.

5 Skip liste

1. Osnovne operacije sa skip-listom:

Algoritam Search(S, k)

Ulaz S (data skip lista), k (ključ elementa koji tražimo)

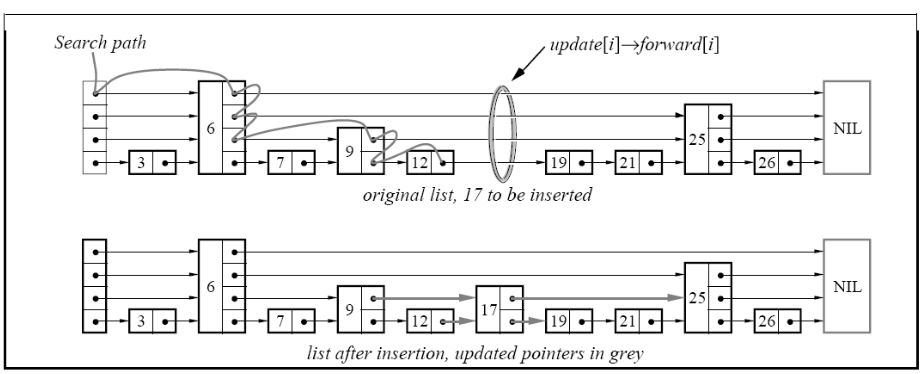
Izlaz vrednost elementa sa ključem k

```
begin
    x := S → header;
    for i := S → level downto 1
        while (x → forward[i] → key < k) do
            x := x → forward[i];
    x := x → forward[1];
    if (x → key == k) then return x → value;
        else return failure;
end
```

Algoritam randomLevel()

Izlaz nivo novog čvora generisan na slučajan način tako da važi da procenat p čvorova nivoa i ima $(i+1)$-vi pokazivač

```
begin
    |v| := 1;
    while (random() < p and |v| < maxLevel) do
        |v| := |v| + 1;
    return |v|;
end
```



Algoritam Insert(S, k, v)

Ulaz S (data skip lista), k (ključ elementa koji dodajemo), v (vrednost elementa koji dodajemo)

Izlaz skip lista sa dodatim elementom

```
begin
    x := S → header;
    for i := S → level downto 1
        while (x → forward[i] → key < k) do
            x := x → forward[i];
        update[i] := x; /* pamtimo prethodni element nivoa i */
    x := x → forward[1];
    if (x → key == k) then x → value := v;
    else
        |v| :=randomLevel();
        if |v| > S → level then
            for i := S → level + 1 to |v| do
                update[i] := S → header;
            S → level := |v|;
        x :=makeNode(|v|, k, v);
        for i := 1 to level do
            x → forward[i] := update[i] → forward[i];
            update[i] → forward[i] := x;
end
```

Algoritam Delete(S, k)

Ulaz S (data skip lista), k (ključ elementa koji brišemo)

Izlaz skip lista sa obrisanim elementom

```
begin
    x := S → header;
    for i := S → level downto 1
        while (x → forward[i] → key < k) do
            x := x → forward[i];
        update[i] := x;
    x := x → forward[1];
```

```
    if (x → key == k) then
        for i := 1 to S → level do
            /* ako je i preskočilo nivo datog čvora,
            izlazimo iz petlje */
            if update[i] → forward[i] ≠ x then break;
            update[i] → forward[i] := x → forward[i];
        free x;
        /* ako je to bio čvor najvišeg nivoa
        dekrementiramo nivo liste */
        while (S → level > 1) and
            (S → header → forward[S → level] == NIL) do
                S → level := S → level − 1;
end
```

2. Napisati algoritam Select(S, k) za odredjivanje k-tog najvećeg elementa skip liste S koja ima n elemenata. Dozvoljeno je dodati polja svakom od elemenata liste S. Složenost algoritma treba da bude $O(\log n)$.

Algoritam Select(S, k)

Ulaz S (data skip lista), k (rang elementa koji tražimo)

Izlaz k-ti najveći element

```
begin
    p := S → header;
    pos := 0;
    for i := S → level downto 1
        while (pos + d(p, i) ≤ k)
            pos := pos + d(p, i);
            p := p → forward[i];
        if (pos == k)
            return p;
end
```

3. Napisati algoritam koji konstruiše skip listu S na osnovu zadatog binarnog stabla pretrage T koje ima n elemenata, tako da vremenska složenost svake od operacija za S u najgorem slučaju bude $O(\log n)$. Stablo T može biti nebalansirano. Vremenska složenost algoritma treba da bude $O(n)$.

Algoritam Build(T, S)

Ulaz T `(dato stablo)`

Izlaz S `(skip lista)`

```
begin
    S_1 := INORDER(T);
    i := 1;
    while (i < log n)
        for j := 1 to |S_i|
            if (mod(j,2) == 0)
                S_{i+1}.add(S_i[j]);
                S_{i+1}[|S_{i+1}| - 1].setChildPtr(S_i[j]);
        i++;
end
```

6 Sortiranja

1. Sortirati RADIX SORT-om brojeve: 329, 457, 657, 839, 436, 720, 355.

2. Dato je k listi i svaka od njih sadrži n elemenata. Ključevi elemenata su celi brojevi iz opsega $[1, m]$. Pokazati kako se mogu sortirati sve liste tako da vremenska složenost u najgorem slučaju bude $O(kn + m)$.

3. Dat je niz od n prirodnih brojeva sa više ponavaljanja elemenata tako da je broj različitih elemenata u nizu $O(\log n)$.

   (a) Konstruisati algoritam za sortiranje ovakvih nizova u kome se izvršava najviše $O(n \log \log n)$ uporedjivanja brojeva.

   (b) Zbog čega je složenost ovog algoritma manja od donje granice $\Omega(n \log n)$ za sortiranje niza od n brojeva?

4. Pokazati kako je moguće sortirati n celih brojeva iz opsega $[0, n^2 - 1]$ u vremenu $O(n)$.