Borna Gajić

Zadatak 1.
Napišite pseudokod za rekurzivnu proceduru OS-Key-Rank(T, k) kojoj se kao ulaz prosljeđuje redna statistika T_i ključ k, a kao izlaz vraća rang ključa k u dinamičkom skupu koji predstavlja T. Pretpostavite da su svi ključevi u T međusobno različiti.

Rješenje

```
def OS-KEY-RANK(T, k):
    if k < T.key
        return OS-KEY-RANK(T.left, k)
    else if k > T.key
        return T.left.size + 1 + OS-KEY-RANK(T.right, k)
    else
        return 1 + T.left.size
    end if
```

Zadatak 2.
Uočite da se u OS-Select i OS-Rank size atribut čvora koristi samo kako bi se izračunao rang. Pretpostavimo da u čvor umjesto atributa size spremamo njegov rang u podstablu u kojem je taj čvor korijen. Pokažite kako bi se informacija o rangu održavala prilikom izvršavanja insert i delete operacija na stablu. Pri tome imajte na umu da ove dvije operacije mogu uzrokovati rotacije.

Rješenje
Prilikom izvršavanja procedure $INSERT$ morali bi smo povećavati rank za jedan svim čvorovima na putu od T.root do zadnjeg čvora prije ubacivanja novog elementa, ukoliko je novi element lijevo dijete svog roditelja. U suprotnom, rank čvorova na putu do novog elementa ostaje nepromijenjen. Kod procedure DELETE oduzimamo jedan, ako je element bio lijevo dijete svog roditelja.
Ukoliko su potrebne rotacije nakon operacija $INSERT$ i $DELETE$
(x je element na kojem se vrši rotacija):
Kod lijeve roatcije rank čvora x ostaje nepromijenjen, dok rank desnog djeteta od x (koje postaje roditelj od x) postaje jednak $y.rank = y.rank + x.rank$
Postupak kod desne rotacije se vrši analogno.

Zadatak 3.
Napišite pseudokod za lijevu rotaciju koja djeluje na čvorovima intervalnog stabla i mijenja max atribute čvorova u $O(1)$ vremenu.

Rješenje

```
def LEFT-ROTATE(T, x):
    y = x.right
    x.right = y.left
    if x.left != T.NIL
        y.left.parent = x
        y.parent = x.parent
        if x.parent == T.NIL
            T.root = y
        else if x == x.parent.left
            x.parent.left = y
            x.max = Max{y.left.max, y.right.max, y.int.high}
        else
            x.parent.right = y
            y.left = x
            y.max = Max{y.left.max, y.right.max, y.int.high}
        end if
    end if
```

Zadatak 4. Napišite pseudokod za Interval-Search tako da radi ispravno i kada su svi intervali otvoreni.

Rješenje

```
def INTERVAL-SEARCH(T, i):
    x = T.root
    while x != T.NIL and "i ne sjece x.int" do
        if x.left != T.NIL and x.left.max > i.low
            x = x.left
        else
            x = x.right
        end if
    end while
    return x
```

Zadatak 5. Koliko iznosi maksimalan broj ključeva (iskazan u terminima minimalnog stupnja t) koji se može spremiti u B-stablo visine h?

Rješenje

Maksimalan broj ključeva koji se mogu spremiti u B-stablo visine h i minimalnog stupnja t iznosi suma

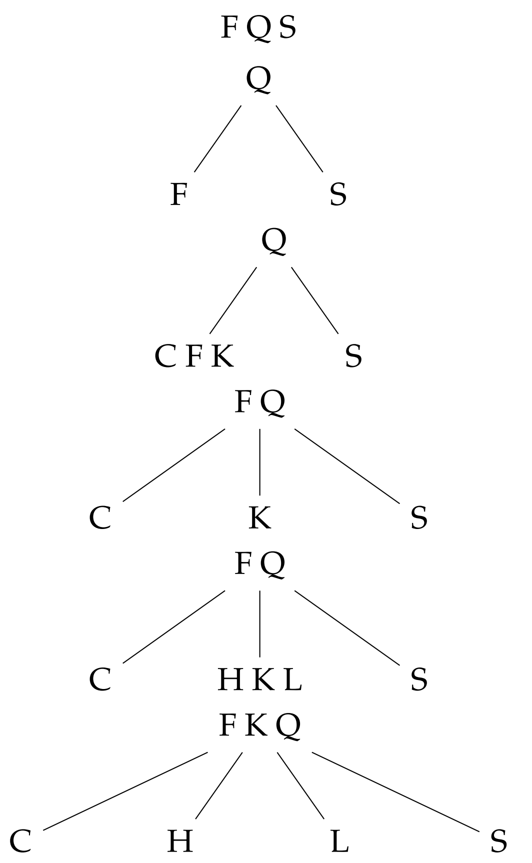
$$n = \sum_{i=0}^{h} (2t-1) \cdot (2t)^i = (2t)^{h+1} - 1$$

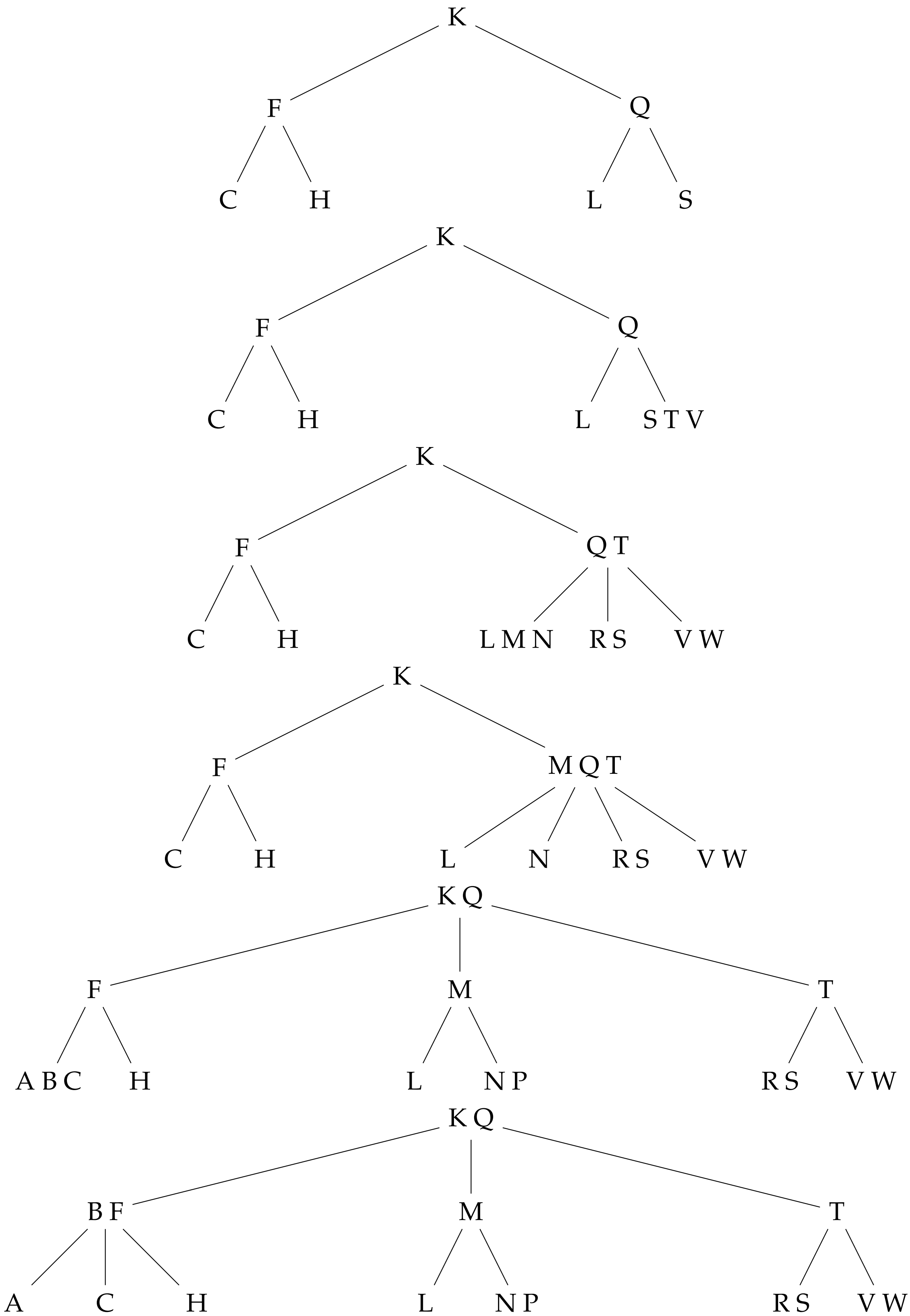
Gdje izraz $2t-1$ predstavlja maksimalni broj ključeva koji se mogu spremiti u čvor, a $2t$ maksimalni broj djece koje taj čvor može imati. Stoga suma računa broj ključeva koji se mogu spremiti u čvorove na svakoj razini i, od nulte do h-te i ona iznosi $(2t)^{h+1} - 1$

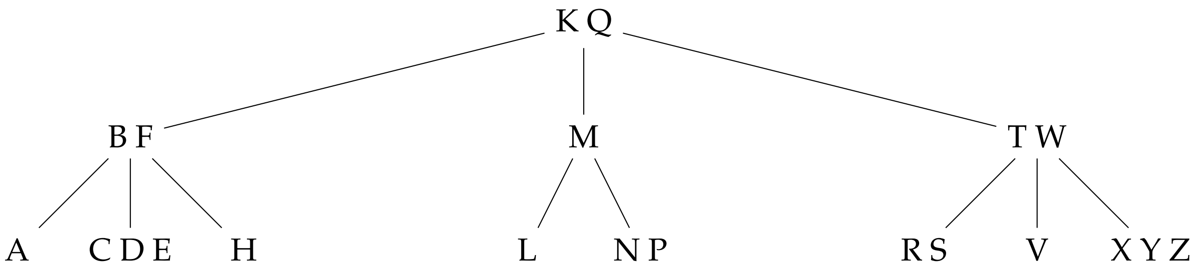
Zadatak 6.

Prikažite postupak umetanja ključeva F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E redom u B-stablo s minimalnim stupnjem 2. Dovoljno je nacrtati samo one konfiguracije stabla koje se događaju neposredno prije nego se neki čvor treba razdvojiti i konfiguraciju nakon tog razdvajanja. Nacrtajte i rezultirajuće stablo nakon svih umetanja.

Rješenje




K
F
Q
C
H
L
S
K
F
Q
C
H
L
S T V
K
F
Q T
C
H
L M N
R S
V W
K
F
M Q T
C
H
L
N
R S
V W
K Q
F
M
T
A B C
H
L
N P
R S
V W
K Q
B F
M
T
A
C
H
L
N P
R S
V W



Zadatak 7.
Pretpostavimo da je B-Tree-Search implementiran tako da se umjesto linearnog pretraživanja u čvoru koristi binarno pretraživanje. Pokažite da se ovom promjenom CPU vrijeme mijenja u O(lg n), neovisno o izboru parametra t kao funkcije od n.

Rješenje
Vrijeme izvršavanja $BINARY-SEARCH$ algoritma je $O(\log n)$, no u našem slučaju bi bilo $O(\log(2t-1))$, gdje t predstavlja minimalni stupanj B-stabla, jer je $2t-1$ najveći broj ključeva koje možemo staviti u B-stablo. Gornju granicu možemo zapisati i kao $O(\log(2t-1)) = O(\log t)$ pošto je razlika u vremenu izvršavanja zanemariva; odnosno razlika je do na konstantu. $B-TREE-SEARCH$ ima linearno vrijeme izvršavanja od $O(h)$ što znamo da je ekvivalentno $O(\log_t n)$. Iz toga imamo:

$$O(h \cdot \log t) = O(\log_t n \cdot \log t) = O(\frac{\log n}{\log t} \cdot \log t) = O(\log n)$$