Rječnik (Dictionary)

Rječnik je skup kod kojeg imamo samo operacije `MakeNull`, `Insert`, `Delete` i `Member`.

Zbog manjeg broja operacija moguće su implementacije efikasnije od implementacija općenitog skupa. Rječnik se često implementira pomoću sortirane liste: lista je prikazana pomoću polja, jedna ćelija je jedan element rječnika, `Member` se implementira algoritmom binarnog traženja (složenost $O(\log n)$), `Insert` i `Delete` su spori!

a.t.p. Dictionary

```
elementtype     . . .
Dictionary      . . .
DiMakeNull(&A)  . . .
DiInsert(x,&A)  . . .
DiDelete(x,&A)  . . .
DiMember(x,A)   . . .
```

Zadatak 3.

Nacrtajte prikaz rječnika $A = \{a, d, f, g, l, m, o, p, s, t, v, z\}$ u sortiranom polju. Opišite rad funkcije `DiMember(x, A)` za `x` $= v$ i `x` $= c$. Koliko čitanja je potrebno?

Rješenje.

```
typedef struct {
    int last;
    elementtype elements[];
} Dictionary;

int DiMember(elementtype x, Dictionary A) {
    int f = 0, l = A.last, m;
    while (f <= l) {
        m = (f + l) / 2;
        if (A.elements[m] == x)
            return m;
        else if (A.elements[m] < x)
            f = m + 1;
        else
            l = m - 1;
    }
    return -1;
}
```

Uočimo, funkcija nije potpuno u skladu s definicijom u apstraktnom tipu podataka – ako je element u skupu, ona umjesto `1` vraća indeks tog elementa, a ako nije, umjesto `0` vraća `-1`.

Zadatak 4.

Neka je n broj elemenata u rječniku koji je prikazan sortiranim poljem. Pokažite da je vrijeme izvršavanja funkcije `DiMember` uvijek $O(\log n)$.

Rješenje.

Napomena. Ako ima puno `Insert` / `Delete` operacija, često se koriste tzv. hash tablice.

Zadatak 5.

Nacrtajte dijagram na kojem se vidi prikaz rječnika {BARBARA, CECILIA, EMMA, LISA, MARY, PAMELA, RUTH, TINA, VILMA, XENIA, YVONNE} pomoću otvorene hash tablice s $B = 7$ pretinaca. Hash funkcija je

$$h(\text{ime}) = \left\lfloor \frac{\text{redni broj prvog slova imena u eng. abecedi}}{4} \right\rfloor .$$

(Pretpostavite da su imena ubacivana redoslijedom iz gornjeg popisa i da novi zapis ide na kraj liste.)

Rješenje.

Zadatak 6.

Nacrtajte dijagram rječnika iz prethodnog zadatka s linearnim hashiranjem s $B = 14$ pretinaca i hash funkcijom

$$h(\text{ime}) = \left\lfloor \frac{\text{redni broj prvog slova imena u eng. abecedi}}{2} \right\rfloor .$$

Rješenje.

Ovdje je došlo do tipične pojave za ovaj tip hashiranja: stvorile su se nakupine (eng. clusters).

Napomena. Za funkcioniranje hash tablice važno je da hash funkcija h uniformno distribuira elemente rječnika u pretince, tj. da je vjerojatnost da slučajno odabran element rječnika upadne u pretinac podjednaka za sve pretince.

Navodimo sada primjere često korištenih dobrih hash funkcija.

Otvoreno hashiranje

Primjer. sredina kvadrata

Primjer. modulo funkcija

Primjer. preklapanje

Zatvoreno hashiranje

Kod zatvorenog hashiranja osim $h(a)$ trebaju nam alternativni pretinci $h_1(a)$, $h_2(a)$, ... za slučaj preklapanja – element se tada sprema u prvi pretinac koji je prazan.

Najjednostavnije linearno hashiranje $h_i(a) = (h(a) + i) \,\%\, B$ nije jako dobro jer često dolazi do stvaranja nakupina popunjenih pretinaca – to vodi do sporog nalaženja elementa a ako se element a morao smjestiti daleko od idealnog pretinca $h(a)$. Evo nekih alternativa.

Primjer. $h_i(x) = (h(x) + c \cdot i) \,\%\, B$ za neki $c > 1$

Primjer. $h_i(x) = (h(x) + i^2) \,\%\, B$

Primjer. $h_i(a) = (h(a) + d_i) \,\%\, B$, gdje je d_1, d_2, ..., d_{B-1} slučajna permutacija od $\{1, 2, \ldots, B-1\}$

Primjer. double hashing $h_i(a) = (h(a) + i \cdot \tilde{h}(a)) \,\%\, B$

Sada ćemo se baviti još jednom dobrom implementacijom rječnika: binarnim stablom traženja.

Binarno stablo traženja

Definicija. Binarno stablo T je binarno stablo traženja (BST) ako vrijedi

(1) korisne informacije (oznake) pohranjene u T imaju definiran totalni uređaj $\leq$

(2) ako je oznaka l spremljena u nekom čvoru v stabla T, onda su oznake spremljene u čvorovima lijevog podstabla od v manje od l, a oznake spremljene u čvorovima desnog podstabla veće ili jednake od l.

Napomena. Svakom elementu rječnika odgovara točno jedan čvor stabla i obratno. Elementi rječnika su oznake (može i imena) čvorova.

Zadatak 7.

Nacrtajte dijagram na kojem se vidi prikaz rječnika $A = \{m, a, r, f, d, s, v, t, g, z, o, l\}$ pomoću BST. Stablo je sagrađeno od praznog stabla uzastopnim ubacivanjem elemenata. Pretpostavite da je redoslijed ubacivanja: (a) kakav je naveden kod definicije, (b) po abecedi uzlazno.

Rješenje.

Napomena. Uočimo da za `Member` i `Insert` treba maksimalno onoliko koraka kolika je visina stabla. Zato je bolje da je stablo sličnije potpunom stablu ($O(\log n)$) nego putu ($O(n)$). U običnoj se implementaciji ne možemo boriti protiv toga. Postoje bolje implementacije koje rade balansiranje stabla (red-black trees).

Zadatak 8.

Napišite funkcije kojima se implementiraju operacije `Min` i `Max` pod pretpostavkom da je skup (rječnik) prikazan s BST. Samo binarno stablo je implementirano pomoću pointera.

Rješenje. `Min` ide lijevo sve dok može, `Max` ide desno sve dok može.

```
elementtype Min(celltype* A) {
    if (A->leftchild == NULL)
        return A->label;
    else
        return Min(A->leftchild);
}
```

Napomena. Pomoću BST može se napraviti lijep algoritam za sortiranje liste podataka:

1. Listu $L = (a_1, a_2, ..., a_n)$ shvatimo kao skup uređenih parova $A = \{(a_1, 1), (a_2, 2), ..., (a_n, n)\}$ s leksikografskim uređajem.
2. Postupnim ubacivanjem sagradimo BST za A.
3. Napravimo INORDER obilazak stabla – to je sortirana lista.

Zadatak 9.

Sortirajte listu $L = (8, 3, 5, 9, 8, 2, 1, 10, 3, 7, 4, 10)$ pomoću BST.

Rješenje.

Uočimo da se redoslijed duplikata u sortiranoj listi podudara s njihovim redoslijedom u polaznoj listi. To može biti korisno ako sortiramo naprimjer listu zapisa po više elemenata zapisa (npr. studente po ocjeni, pa po prezimenu). Algoritme za sortiranje koji imaju ovo svojstvo zovemo *stabilnima* (eng. stable algorithm).

Napomena. Komplikaciju s čuvanjem početne pozicije u A uvodimo samo ako želimo nakon sortiranja moći dohvatiti koje je mjesto imao neki element u polaznoj listi. Ako nam to nije važno, možemo umjesto (a_i, i) u stablu čuvati samo a_i.

Zadatak 10.

Dokažite da INORDER zaista obilazi elemente binarnog stabla traženja u sortiranom redoslijedu.

Rješenje.