Zadaci

1. Implementirati *hash* tabelu zasnovanu na rešavanju konflikata upotrebom pristupa ulančavanja (*chaining*). Na slici 1 dati su pseudokodovi funkcija za rukovanje *hash* tabelom. Elemente tabele (podatke) predstaviti kao objekat klase *Data* koja ima bar dva atributa: *key* i *literal*. Atribut *key* predstavlja celobrojnu vrednost koja se koristi kao argument u *hash* funkciji, dok atribut *literal* predstavlja znakovnu vrednost atributa *key*. Za pravljenje *hash* funkcije (h), odnosno za rešavanje koflikata, koristiti sledeće pristupe:

   a. Metod deljenja (*The division method*)

      $$h(k) = k \bmod m$$

      k – vrednost ključa
      m – broj mesta (slotova) za smeštanje elemenata

   b. Metod množenja (*The multiplication method*)

      $$h(k) = \text{floor}(m\ (kA \bmod 1)); \quad A \sim (\sqrt{5} - 1)/2 = 0.6180339887...$$

      A – Optimalna konstana (Knuth)

   c. Univerzalno heširanje (*Universal hashing*)

      $$h_{ab}(k) = ((ak + b) \bmod p) \bmod m$$

      p – prost broj dovoljno velik da obuhvata sve vrednosti ključeva k; $k \in Z_p$;
      a, b – slučajno izabrani brojevi na početku izvršavanja; $a \in Z_p^*$; $b \in Z_p$;
      $Z_p = \{0, 1, 2, ..., p-1\}$;
      $Z_p^* = \{1, 2, ..., p-1\}$;

```
CHAINED-HASH-INSERT(T, x)
1  insert x at the head of list T[h(x.key)]

CHAINED-HASH-SEARCH(T, k)
1  search for an element with key k in list T[h(k)]

CHAINED-HASH-DELETE(T, x)
1  delete x from the list T[h(x.key)]
```

Slika 1 – Pseudokod funkcija za rukovanje *hash* tabelom (*chaining*)

Izmeriti (i) vreme formiranja (punjenja) *hash* tabele i (ii) vreme pretrage slučajno odabranog broja iz ulaznog skupa. Za eksperiment koristiti sledeće kombinacije parametara:

- n = 10, 50 i 100 hiljada slučajno generisanih ulaznih brojeva (*key* vrednosti) u opsegu [0, p-1],
- za n = 10, 50 i 100 hiljada ulaznih elemenata koristiti proste brojeve p = 23, 9973 i 99991, respektivno,
- koristiti različite veličine *hash* tabele, $m = p$, $p/2$ i $p/4$.

  Napomena: Prilikom formiranja *hash* tabele voditi računa o tome da se već postojeći podaci u tabeli ažuriraju, odnosno, pre ubacivanja podatka sa ključem *key* treba proveriti da li taj ključ već postoji u *hash* tabeli.

2. Implementirati *hash* tabelu zasnovanu na rešavanju konflikata upotrebom pristupa otvorenog adresiranja (*open addressing*). Pseudokodovi funkcija za rukovanje *hash* tabelom dati su na slici 2. Elemente hash *tabele* predstaviti kao u prethodnom zadatku. Za pravljenje *hash* funkcije (h) koristiti sledeće pristupe provere:

   a. Linearna provera (*Linear probing*)

   $$h(k, i) = (h'(k) + i) \bmod m$$

   h' – pomoćna *hash* funkcija
   k – vrednost ključa
   m – broj mesta (slotova) za smeštanje elemenata
   i – pokušaj provere; $i \in \{0, 1, 2, \ldots, m-1\}$

   b. Kvadratna provera (*Quadratic probing*)

   $$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

   c_1 i c_2 – konstante

   c. Dvostruko heširanje (*Double hashing*)

   $$h(k, i) = (h_1(k) + h_2(k)) \bmod m$$

   h_1 i h_2 – pomoćne *hash* funkcije

   Izmeriti (i) vreme formiranja (punjenja) *hash* tabele i (ii) vreme pretrage slučajno odabranog broja iz ulaznog skupa. Za eksperiment koristiti sledeće kombinacije parametara:

   - n = 10, 50 i 100 hiljada slučajno generisanih ulaznih brojeva (*key* vrednosti),
   - konstante $c_1 = c_2 = 1/2$,
   - pomoćne funkcije h_1 i h_2:

- $h_1(k) = k \bmod m$
- $h_2(k) = 1 + (k \bmod m')$; $m' <= m-1$

- koristiti različite veličine *hash* tabele, $m = n$, $n/2$ i $n/4$.

Napomena: Prilikom formiranja *hash* tabele voditi računa o tome da se već postojeći podaci u tabeli ažuriraju, odnosno pre ubacivanja podatka sa ključem *key* treba proveriti da li taj ključ već postoji u *hash* tabeli.

```
HASH-INSERT(T, k)
1  i = 0
2  repeat
3      j = h(k, i)
4      if T[j] == NIL
5          T[j] = k
6          return j
7      else i = i + 1
8  until i == m
9  error "hash table overflow"
```

```
HASH-SEARCH(T, k)
1  i = 0
2  repeat
3      j = h(k, i)
4      if T[j] == k
5          return j
6      i = i + 1
7  until T[j] == NIL or i == m
8  return NIL
```

Slika 2 – Pseudokod funkcija za rukovanje *hash* tabelom (*open addressing*)