Haszowanie

Niniejsze zajęcia poświęcone są implementacji wybranych metod haszowania. Dane jest uniwersum U, w którym nie określono porządku liniowego. Zadaniem jest reprezentacja zbiorów $A \subseteq U$ wraz z operacją $Search(A, x)$ – wyszukania x w zbiorze A. Jeśli $x \in A$, to otrzymujemy odpowiedź TRUE, w przeciwnym przypadku FALSE i dodatkowo element jest wstawiany do zbioru A.

Zakładamy, że dana jest liczba $M \in \mathbb{N}$ oraz funkcja haszująca $h : U \to \{0, \ldots, M-1\}$. Elementy zbioru A reprezentować będziemy tablicy $H[0..M-1]$. Dla $x, y \in U$, jeśli $h(x) = h(y)$, to mamy do czynienia z *kolizją*.

1. Hashowanie łańcuchowe

Elementami tablicy H są głowy list elementów kolidujących. Można reprezentować zbiory o liczności większej niż M.

Algorytm 4.1: *ChainSearch*

```
ChainSearch(x : U) : Bool;
var i : int; p,t : list;
begin
   i := h(x);
   t := H[i]; p := NULL;
   while t ≠ NULL do
      if t↑.elem = x then return(true) fi;
      p := t; t := t↑.next;
   od;
   new(t); t↑.elem := x; t↑.next := NULL;
   if p = NULL then H_i := t else p↑.next := t  fi;
   return(False)
end ChainSearch;
```

2. Hashowanie rozproszone

Elementy zbioru A wraz z listą elementów kolidujących są wstawiane do tablicy H. Ściślej: $H[i]$ jest parą (el, idx), gdzie wartością pola el są elementy zbiory A, zaś idx jest indeksem w tablicy H gdzie umieszczono następny element z listy kolizji.

Algorytm 4.1: ***ScatteringSearch***

```
ScatteringSearch(x : U) : Bool;
var i : int;
begin
  i := h(x);
  while H[i].elem ≠ 0 do
    while i ≠ M do
      if H[i].elem = x then return(True) fi;
      i := H[i].next;
    od;
    while H_last.elem ≠ 0 do
      last := last − 1;
      if last = −1 then ERROR fi;
      od;
    H[i].next := last; i := last;
  od;
  H[i].elem := x; H[i].next := M; return(False)
end ScatteringSearch;
```

3. Hashowanie otwarte

Elementami tablicy H są jedynie elementy zbiory A, jeśli występuje kolizja, element wstawiany jest w pierwsze "od dołu" (w kierunku malejących indeksów) wolne pole tablicy H.

Algorytm 4.1: *OpenSearch*

```
OpenSearch(x : U) : Bool;
var i : int;
begin
   i := h(x);
   while H[i] ≠ 0 do
     if H[i] = x then return(True) fi;
     i := i−1;
   if i<0 then i := M−1 fi;
   od;
   if n = M−1 then ERROR fi;
   n := b+1; H[i] := x; return(False)
end OpenSearch;
```

4. Zadania

1. Zaimplementować metodę adresowania łańcuchowego dla zbiorów liczb całkowitych. Jako funkcje haszujące przyjąć funkcje postaci $h(x) = x \bmod M$, gdzie M jest rozmiarem tablicy haszującej.

2. Zaimplementować metodę podwójnego adresowania otwartego dla zbiorów liczb całkowitych. Jako funkcje haszujące przyjąć funkcje postaci $h(x) = \lfloor M \cdot \mathrm{frac}(\alpha \cdot x) \rfloor$, gdzie M jest rozmiarem tablicy haszującej, $\alpha = \frac{\sqrt{5}-1}{2}$, zaś $frac(y)$ jest funkcją zwracającą część ułamkową argumentu y.

3. Zaimplementować metodę adresowania rozproszonego dla zbiorów liczb całkowitych. Jako funkcje haszujące przyjąć funkcje postaci $h(x) = (ax \bmod b) \bmod M$, gdzie $a = \frac{\sqrt{5}-1}{2}$ gdzie M jest rozmiarem tablicy haszującej, zaś a i b liczbami pierwszymi.