TABELA DE DISPERSIE

Hash Table

- Este o structură de date **eficientă** pentru implementarea dicționarelor (și nu numai).
- Compilator – **tabelă de simboluri** (cheia = șir de caractere corespunzătoare unui identificator)
- TD poate fi folosită pentru implementarea containerelor pe care operațiile specifice sunt: **adăugare** element, **căutare** element, **ștergere** element. Ex: dicționare, colecții, mulțimi
  - JAVA - HashMap, HashSet.
  - STL – unordered_set, unordered_map
- TD este o generalizare a noțiunii mai simple de **tabelă cu adresare directă**
- Notații
  - n – numărul de elemente din container
  - un element e din container este o pereche cheie (c) – valoare (v) ($TElement = TCheie \times TValoare$)
  - U – universul (domeniul) cheilor
  - K – domeniul actual al cheilor (mulțimea cheilor efectiv memorate în container)

Tabelă cu adresare directă

- Notații
  - chei numere naturale, chei distincte
  - $U = \{0,1,2...,m\text{-}1\}$ - m relativ mic
  - K – domeniul actual al cheilor (mulțimea cheilor efectiv memorate în container)
- Tabela cu adresare directă este memorată sub forma unui vector T[0..m-1]
  - Locația T[c] va corespunde cheii c
  - Dacă o cheie $c \notin K$, atunci T[c] va conține NIL (sau o valoare special care marchează locație goală)
  - T[c] poate memora un pointer spre elementul având cheia c sau elemental (cheia și valoarea asociată)

TElement
  c:TCheie
  v:TValoare

TabelaAdresareDirecta
  m:Întreg
  e:TElement[0..m-1]

CAUTĂ (T, c)
//pre: T este o tabelă cu adresare directă, c este de tip TCheie
@ returnează $T.e[c]$

ADAUGĂ (T, e)
//pre: T este o tabelă cu adresare directă, e este de tip TElement
@ $T.e[e.c] \leftarrow e$

ȘTERGE (T, c)
//pre: T este o tabelă cu adresare directă, c este de tip TCheie
@ $T.e[e.c] \leftarrow$ NIL

- Observații
  - funcționează bine dacă universul cheilor este mic
  - complexitatea timp a operațiilor este $\phi(1)$
  - spațiul de memorare este $\phi(|U|)$

Tabela de dispersie

- spațiul de memorare este $\phi(|K|)$
- complexitatea timp ***medie*** pentru toate operațiile pe TD (adăugare, căutare, ștergere) este $\phi(1)$.
  - **căutarea** unui element într-o TD poate necesita $\phi(n)$ în caz ***defavorabil*** (ca și căutarea în liste)
    - în practică, dispersia funcționează foarte bine
    - timpul ***mediu*** preconizat pentru căutare este $\phi(1)$
- funcție de dispersie (*hash function*) $d : U \rightarrow \{0,1,...m-1\}$
- **dispersia perfectă** (*perfect hashing, perfect hash function*)
- $d(c_1) = d(c_2)$ coliziune
- **adăugarea unui nou element** $e=(c, v)$
  - se calculează $i = d(c)$
  - dacă locația i este liberă, atunci se adaugă elementul
  - dacă la locația i mai e memorat un alt element $\Rightarrow$ **rezolvare coliziune**
- funcție de dispersie bună
  - este ușor de calculat (folosește operații aritmetice simple) - $\phi(1)$
  - produce cât mai puține coliziuni.

Interpretarea cheilor ca numere naturale

- Majoritatea funcțiilor de dispersie presupun universul cheilor din mulțimea numerelor naturale
- În cazul în care cheile nu sunt numere naturale, trebuie găsită o modalitate de a le interpreta ca numere naturale – o funcție care asociază fiecărei chei un număr natural (implementare - $hashCode: TCheie \rightarrow \{0,1,2...\}$)
  - Ex: identificatorul **pt** poate fi interpretat ca un număr în baza **128** – $(\mathbf{pt})_{128}=112\cdot128+116=14452$.
- Pp. în cele ce urmează că avem chei naturale.

Funcții de dispersie

- O funcție de dispersie bună satisface (aproximativ) *ipoteza dispersiei uniforme simple* (**Simple Uniform Hashing**): fiecare cheie se poate dispersa cu aceeași probabilitate în oricare din cele *m* locații.
  - $P(d(c)=j)=\frac{1}{m}, \forall j=0,...,m-1 \quad \forall c \in U$
  - Dacă această ipoteză este verificată, atunci se minimizează numărul de coliziuni
  - în practică se pot folosi tehnici euristice pentru a crea funcții de dispersie care să se comporte bine.

I. Metoda diviziunii

- Dispersia prin diviziune
- $d(c)=c \bmod m$
- Experimental: valori bune pentru *m* sunt numerele prime nu prea apropiate de puteri exacte ale lui 2 (ex: 13,...)
- m=13
  - $c=63 \Rightarrow d(c)=11$
  - $c=26 \Rightarrow d(c)=0$

II. Metoda înmulțirii

- $d(c)=[m\cdot(c\cdot A \bmod 1)]$ unde "$c\cdot A \bmod 1$" reprezintă $c\cdot A-[c\cdot A]$
- Valoarea lui *m* nu e critică (în general este o putere a lui 2)
- Knuth: valoarea optimă pentru *A* este $\frac{\sqrt{5}-1}{2} \approx 0.6180339887$ - **golden ratio**

III. Dispersia universală

- $c=<c_1, c_2, ..., c_k>$

- $d(c) = (\sum_{i=1}^{k} c_i \cdot x_i) \bmod m$ unde $< x_1, x_2, ..., x_k >$ este o secvență de numere aleatoare fixate (selectate de-a lungul inițializării funcției de dispersie)
- Proprietate (foarte bună): oricare ar fi două chei distincte *a* și *b*, probabilitatea ca o funcție de dispersie aleatoare *d* să le mapeze în aceeași locație în tabela de dispersie este $\frac{1}{m}$.

A. <u>Rezolvare coliziuni prin *liste independente* (*înlănțuire*) – SEPARATE CHAINING</u>

- Elementele care se dispersează în aceeați locație (sunt într-o coliziune), vor fi puse într-o listă înlănțuită
- Locația *j* conține un pointer către capul listei înlănțuite a elementelor care se dispersează în locația *j* (dacă această listă e vidă, se memorează NIL).
- Ex: m=10, U={c_1,…c_{10}}, K={ c_1, c_2 c_3 c_4 c_5} și d(c_1)= d(c_4)=1, d(c_2)=5, d(c_3)= d(c_5)=7

<u>Reprezentare și operații</u>

TElement	**Container**
c:TCheie	*m*:Întreg
v:TValoare	*l*:TListă[0..*m*-1]

- *d* este funcția de dispersie
- pp. cheia are o singură valoare asociată

CAUTĂ (*C*, *ch*)
// pre: *C* este un container reprezentat sub forma unei TD (coliziuni prin înlănțuire), *ch* este de tip
// TCheie
@ caută elementul cu cheia *ch* în lista *C.l*[*d*(*ch*)]

ADAUGĂ (*C*, *e*)
// pre: *C* este un container reprezentat sub forma unei TD (coliziuni prin înlănțuire), *e* este de tip
// TElement
@ se adaugă elementul *e* în capul listei înlănțuite *C.l*[*d*(*e.c*)]

ȘTERGE (*T*, *ch*)
// pre: *C* este un container reprezentat sub forma unei TD (coliziuni prin înlănțuire), *ch* este de tip
// TCheie
@ se șterge elementul cu cheia *ch* din lista înlănțuită *C.l*[*d*(*e.c*)]

<u>Observații</u>

- Este posibil ca listele independente să fie memorate ordonat după cheie sau valoare
- Funcția de dispersie este considerată *bună* dacă listele au aproximativ aceeași lungime

➢ Dacă apar multe liste de vide sau liste prea lungi $\Rightarrow$ redispersare (**rehashing**)

<u>Exemplu</u>

m=10, $d(c)$=c mod m

c	5	15	13	22	20	35	30	32	2
d(c)	5	5	3	2	0	5	0	2	2

0→30→20
1
2→2→32→22
3→13
4
5→35→15→5
6
7
8
9

<u>Iterator</u>

Telement
c:TCheie
v:TValoare

Nod
e:TElement
urm:↑Nod

Container
m:Întreg
l:↑Nod [0..m-1]

IteratorContainer
c:Container
pozCurent:Intreg
curent:↑Nod

(a se vedea implementarea operațiilor la http://www.cs.ubbcluj.ro/~gabis/sda/Cursuri/Curs10/).

<u>Timp defavorabil pentru operații</u>

Pp n este numărul elementelor din container.

- **CAUTĂ** – $O(n)$
- **ADAUGĂ -** $\phi(1)$
- **ȘTERGE –** presupune (1) căutare nod în lista înlănțuită + (2) ștergere nod $O(n)$

<u>Analiza dispersiei cu înlănțuire</u>

Notații și presupuneri

➢ $\alpha = \frac{n}{m}$ *factorul de încărcare* al tabelei (numărul mediu de elemente memorate într-o înlănțuire)

- ➢ Pp. că timpul de calcul al funcției de dispersie este $\theta(1)$ (!! la timpul de căutare se adaugă și timpul de calcul al funcției de dispersie

- La **căutare** apar 2 cazuri
  - Căutare **cu succes**
  - Căutare **fără succes**

Teorema 1. Într-o TD în care coliziunile sunt rezolvate prin înlănțuire, în *ipoteza dispersiei uniforme simple* (SUH), o căutare **fără succes**, necesită, în ***medie***, un timp $\theta(1+\alpha)$.

Teorema 2. Într-o TD în care coliziunile sunt rezolvate prin înlănțuire, în *ipoteza dispersiei uniforme simple* (SUH), o căutare **cu succes**, necesită, în ***medie***, un timp $\theta(1+\alpha)$.

CONCLUZII

- Dacă $n = O(m) \Rightarrow \alpha = \frac{O(m)}{m} = O(1) \Rightarrow$ **căutarea** necesită, în ***medie***, timp constant $\theta(1)$
- Adăugarea necesită $\theta(1)$
- Dacă listele sunt dublu înlănțuite atunci ștergerea unui nod se poate face în $\theta(1)$

$\Rightarrow$ **TOATE OPERAȚIILE (adăugare, căutare, ștergere) POT FI EXECUTATE ÎN *MEDIE* ÎN** $\theta(1)$

PROBLEME

1. Presupunem că folosim o funcție de dispersie aleatoare **d** pentru a dispersa *n* chei distincte într-o tabelă *T* de dimensiune *m*. Care este numărul mediu de coliziuni? (cardinalul probabil al mulțimii $\{(x,y) \in TCheie \times TCheie : d(x) = d(y)\}$)
2. Presupunem că folosim o TD în care coliziunile sunt rezolvate prin înlănțuire (liste independente), dar fiecare listă este ordonată după cheie. Care va fi timpul de execuție pentru **căutare** (cu succes, fără succes), **adăugare** și **ștergere**?
3. Arătați că dacă $|U| > n \cdot m$, atunci există o submulțime a lui U de mărime *n* ce conține chei care se dispersează toate în aceeași locatie, astfel încât timpul de căutare pentru dispersia cu înlănțuire, în cazul cel mai defavorabil, este $\phi(n)$.