Laborator 12 - Tabele de dispersie

Tabele de dispersie

Ne punem problema analizarii unor structuri de date din punct de vedere al operatiilor de inserare si de cautare. Elementele structurii de date sunt caracterizate prin chei (naturale sau artificiale) si valori.

Pentru inceput consideram o structura de date de tip lista inlantuita. Aceasta are avantajul de a folosi eficient memoria. De asemenea, operatia de inserare a unui element nou in lista poate fi de complexitate O(1). Insa accesul la al k-lea element al unei liste se face in timp O(k), iar cautarea unui element este de complexitate O(n), unde n reprezinta dimensiunea listei.

O alta varianta este de a folosi un vector. In cazul acestei structuri de date accesul la un element pentru care se cunoaste pozitia, numita si index este constant, O(1), la fel ca si insertia. Dar fara a avea cunostinte apriori despre datele memorate, cautarea are tot complexitate liniara.

Structură de date	**Inserţie**	**Căutare**	**Observaţii**
Listă înlănţuită	$\mathcal{O}(1)$	$\mathcal{O}(n)$	Foloseşte eficient memoria
Vector	$\mathcal{O}(1)$	$\mathcal{O}(n)$	Accesul la un element pentru care se cunoaşte poziţia are complexitate $\mathcal{O}(1)$
Vector ordonat	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	Accesul la un element pentru care se cunoaşte poziţia are complexitate $\mathcal{O}(1)$

Apare atunci intrebarea: putem construi un vector in care cheia unui element sa ne returneze pozitia acestuia? Daca acest lucru ar fi realizat, atunci cautarea unui element ar presupune calcularea indexului sau si verificarea existentei in vector pe pozitia obtinuta. In acest caz si cautarea ar avea complexitate constanta.

O idee initiala ar fi sa se construiasca un vector ai carui indecsi sa fie chiar posibilele chei ale elementelor (adresare directa). Aceasta abordare nu este insa practica intotdeauna, deoarece universul cheilor ar putea fi mult prea mare relativ la memoria care poate fi folosita.

Pentru a evita astfel de situatii se folosesc tabele de dispersie. Acestea reprezinta structuri de date care implementeaza tablouri asociative: cheii unui element i se asociaza un cod din plaja de valori ale indecsilor unui vector; insertia si cautarea se vor face la pozitia data de acest cod.

Maparea intre cheile valorilor de memorat si universul de chei ale structurii de date folosite se realizeaza cu ajutorul unor functii numite functii de dispersie (vezi http://en.wikipedia.org/wiki/Hash_function). Aceste functii transforma valori de dimensiuni variabile in coduri de dimensiune exacta ce pot fi apoi folosite cu rol de indecsi, realizand practic o criptare intr-un singur sens.

Cum in general universul cheilor este mult mai mare decat multimea de indecsi ai tabloului de dispersie, o functie de dispersie va asocia mai multor chei acelasi cod. Acest fenomen poarta numele de coliziune. O posibila rezolvare a acestei probleme se face prin inlantuire: la fiecare pozitie a tabelului se memoreaza o legatura catre o lista inlantuita de elemente. Atunci un element va fi inserat sau cautat in lista corespunzatoare codului hash al acestuia.

Eficienta acestei abordari depinde de cat de "buna" este o functie de dispersie. Aceasta trebuie sa aiba urmatoarele proprietati: sa fie usor de calculat, sa evite coliziunile si sa distribuie uniform cheile in tabel. Daca aceste conditii sunt respectate, atunci operatia de insertie are complexitate O(1), iar cea de cautare are complexitate medie O(n=m), unde n reprezinta numarul de elemente memorate in tabel, iar m reprezinta numarul de pozitii ale acestuia.

Astfel, un element cu cheia k nu va fi memorat în locatia k, ci în h(k), unde h: U -> {0, 1, ..., N-1} este o funcție aleasă aleator, dar deterministă (adica h(x) va returna mereu aceeași valoare pentru un anumit x în cursul rulării unui program). Functia hash depinde de cheia utilizata.

O aplicatie a functiilor de dispersie apare in autentificarea utilizatorilor.Pentru a asigura o buna securitate a parolelor, acestea nu sunt memorate in clar. Unei parole i se aplica o serie de functii de dispersie si ceea ce se memoreaza in baza de date este valoarea obtinuta. Atunci cand un utilizator introduce o parola pentru a se autentifica, acesteia i se aplica aceeasi serie de functii si rezultatul se compara cu valoarea memorata pentru respectivul utilizator. Daca sunt egale se considera ca autentificarea a fost realizata cu succes. Important in aceasta situatie este ca fiind dat un cod hash si cunoscandu-se functia cu care a fost obtinut sa fie dificil de calculat valoarea initiala.

Tabele de dispersie sunt folosite in implementarea tablourilor asociative, a unei memorii de tip cache etc.

Un exemplu de utilizare a unui tabel de dispersie este redat in figura de mai jos, utilizand o functie hash h(x)=x mod 11, in care coliziunile sunt rezolvate prin inlantuire (utilizand liste):

Text:	U	N		E	X	E	M	P	L	U		I	N	D	E	X	A	R	E
Hash:	**8**	**1**		**3**	**0**	**3**	**0**	**3**	**10**	**8**		**7**	**1**	**2**	**3**	**0**	**10**	**5**	**3**
ASCII	85	78		69	88	69	77	80	76	85		73	78	68	69	88	65	82	69

h(x) = x mod 11

h	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	X	N	D	E		R		I	U		L
	M	N		E					U		A
	X			P							
				E							
				E							

Cele mai cunoscute functii de dispersie:

- Message Digest (MD) - familia MD5;
- Secure Hash - SHA - mai multe info;
- RACE Integrity Primitives Evaluation Message Digest - RIPEMD;
- Whirlpool;

Aplicatii ale utilizarii functiilor de dispersie:

- Stocarea parolelor;
- verificarea integritatii datelor
- Corespondanta intre fisier si calea acestuia, la nivel de sistem de fisiere
- Cautare de secvente identice - plagiarism (algoritmul Rabin-Karp)

Mai multe exemple si exercitii gasiti aici: https://afteracademy.com/blog/applications-of-hash-table

Un exemplu de generare a valorii hash utilizand functii standard: SHA256 demo

Exercitiul 6.1.

Considerand ca se introduce urmatorul text: "TEXT PENTRU CALCULUL REZUMATULUI", determinati numarul de valori posibile generate (reprezentand indecsii unor tabele de dispersie), respectiv numarul de coliziuni, pentru urmatoarele functii de dispersie:

1. h(x) = x mod 12;

2. h(x) = x mod 8;

    a. pentru fiecare litera din text;

    b. pentru fiecare cuvant, calculand suma codului ASCII a fiecarei litere continute;

    c. Creati un program C/C++ pentru rezolvarea subpunctelor de mai sus.

Creati un program C/C++ pentru rezolvarea subpunctelor de mai jos:

3. h(x) =[M/W (x^2 mod W)], unde $W = 2^6$ si $M = 2^5$;

4. h(x) =[M/W (ax mod W)], unde $W = 2^6$, $M = 2^5$;si a = 101;

5. h(x) = [m(Ax mod 1)], unde prin (Ax mod 1) se noteaza partea fractionara a lui Ax, iar m = 1000 si A =(sqrt(5)-1)/2;

Indicatie: se va folosi codul ASCII pentru fiecare litera din text, formatul zecimal.

Exercitiul 6.2.*

Generati aleator 1000 de numere intregi cu valori cuprinse intre 0 si 9999. Aplicati functiile de dispersie de la exercitiul anterior acestor valori si, fara a construi efectiv un tabel de dispersie, pentru fiecare functie:

1. numarati coliziunile de pe fiecare cheie;

2. calculati urmatoarea suma $\sum_{i=0}^{m-1} \frac{(c_i - \frac{n}{m})^2}{\frac{n}{m}}$

unde n reprezinta numarul de elemente (n = 1000), m numarul de indecsi ai tabelului, iar c_i reprezinta numarul de coliziuni ale cheii i.

Last modified: Monday, 31 October 2022, 9:28 AM

« PREVIOUS ACTIVITY
C12- Functii si tabele de dispersie

NEXT ACTIVITY
Laborator 13 - Cod Huffman »