Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Structuri de Date si Algoritmi

- suport de curs -

Dobrovat Anca - Madalina

An universitar 2019 – 2020
Semestrul I
Seriile 21 + 25

Curs 12

17/12/2019

Curs 12 - Cuprins

6. Tabele de dispersie

Functii de dispersie.

Rezolvarea coliziunilor prin inlantuire.

Rezolvarea coliziunilor prin adresare directa.

Cautare, inserare, stergere in tabele de dispersie.

Dispersie universala.

Sursa: – materiale de curs R. Ceterchi

Tabele de dispersie (hash tables)

Functii de dispersie

Problema: mentinerea unui set finit de date, fiecare dintre ele identificata printr-o cheie unica, astfel incat operatiile de inserare, stergere, cautare sa fie eficiente.

Notatii:
K - *multimea cheilor actuale* (prin care se identifica setul de date)
U - *universul cheilor*

Pe aceasta multime dorim sa facem operatii eficiente. Ideal, cautarea sa fie in O(1) in raport cu |U|, sau amortizat constant.

Rezolvare (posibila) **: tabel cu adresare directa**

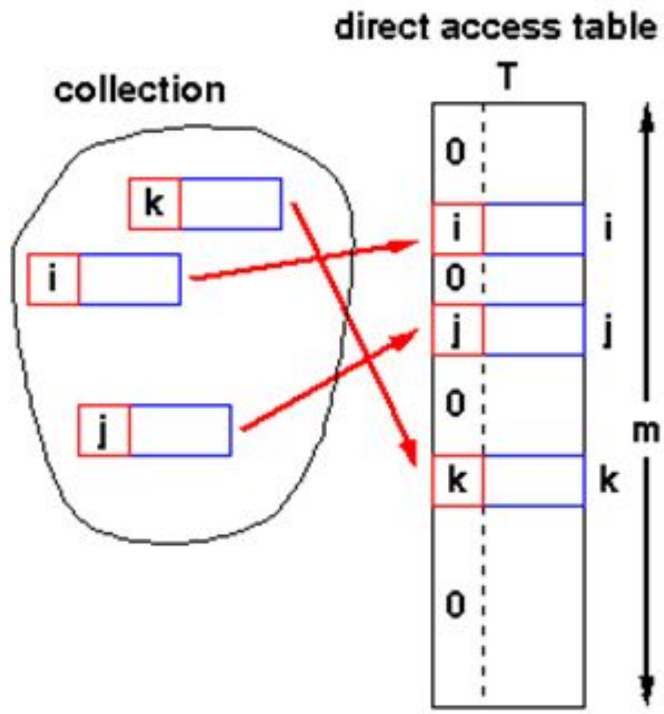
Un vector T a carui multime de indici sa fie chiar U → pentru orice cheie actuala k, in locatia T[k] am putea mentine data identificata de cheia k.

Tabele de dispersie (hash tables)

Functii de dispersie

Rezolvare (posibila) **: tabel cu adresare directa**

Un vector T a carui multime de indici sa fie chiar U → pentru orice cheie actuala k, in locatia T[k] am putea mentine data identificata de cheia k.



Tabele de dispersie (hash tables)

Functii de dispersie

Cazul: |K| << |U| → folosirea unui tabel cu adresare directa, adica a unui tabel T indexat chiar dupa U, posibila risipa de spatiu.

Exemple:

- 50 angajati ai unei firme, fiecare avand numar de identificare din 4 cifre.

|K| = 50
U = {0000, 0001, ..., 9999}
|U| = 10000.

- anuarul telefonic al unei localitati mentine informatii despre abonatii identicati cu ajutorul numelui si prenumelui, deci un sir de maximum 20 de caractere deci va avea dimensiunea |U| = 26^20.

Recomandare: gasirea unei multimi de indici T, |T| = m, m << |U|, eventual apropiat de |K|, si o metoda de a dispersa cheile din K pe multimea de indici {0, 1, …, m-1}. Cu alte cuvinte, avem nevoie de o functie, definita pe universul cheilor si cu valori in multimea indicilor T:

h : U → {0, 1, …, m-1} (functie de dispersie)

Tabele de dispersie (hash tables)

Functii de dispersie

h : U → {0, 1, …, m-1} (functie de dispersie)

Problema de rezolvat: stocarea unui set de date de dimensiune n intr-un tabel de dimensiune m (data identificata printr-o cheie k, sa se gaseasca in locatia T[h(k)] a lui T)

Definim: alpha = n / m (factor de incarcare "***load factor***") - numarul mediu de elemente stocate in tabelul T.

Cerintele indeplinite de o functie buna de dispersie:

1. Sa se poata calcula rapid. Timpul de calcul al lui h(k) ne va da timpul de acces la componenta T[h(k)].
2. Codomeniul ei sa fie cat mai mic. Este dezirabil ca m sa fie cat mai apropiat de |K|.
3. h sa fie cat mai aproape de o functie surjectiva, adica sa avem cat mai putine locatii goale.
4. h sa fie cat mai aproape de o functie injectiva pe K (pentru cheile k1 != k2, h(k1)!=h(k2).

Tabele de dispersie (hash tables)

Functii de dispersie

Ipoteza de lucru:

Ipoteza hashingului simplu uniform (HSU) :orice cheie este distribuita prin functia h cu probabilitate egala in oricare din locatiile {0, 1, …, m-1}.
Mai precis, pentru oricare k, probabilitatea ca h(k) = i este 1/m, unde i = 0, 1, …, m-1, si este independenta de restul cheilor.

In cele ce urmeaza vom interpreta cheile si numerele naturale, adica vom considera ca U < N. Atunci cand cheile sunt stringuri, metoda folosita pentru a le converti la numere naturale este urmatoarea: se inlocuieste ecare caracter cu codul sau ASCII si se calculeaza valoarea sirului rezultat ca intreg in baza 128.

Tabele de dispersie (hash tables)

Functii de dispersie

Tehnici euristice de creare a unor functii de dispersie bune

Metoda diviziunii

Se numesc functii de dispersie obtinute prin metoda diviziunii functiile de forma:

$$\mathbf{h(k) = k \bmod m}$$

Ele sunt printre cele mai comune pentru ca sunt usor de calculat.

Alegerea unei astfel de functii revine practic la alegerea lui m, dimensiunea tabelului de dispersie.

Important este ca cheile sa e bine dispersate pe multimea {0,1, … ,m-1} si sa avem cat mai putine ***coliziuni (definitia putin mai tarziu)***.

Tabele de dispersie (hash tables)

Functii de dispersie

Tehnici euristice de creare a unor functii de dispersie bune

Metoda diviziuni

De evitat, ca valori pentru m:

- puterile lui 2; daca m = 2^p, h(k) = k mod 2^p reprezinta doar ultimii p biti din scrierea binara a lui k. Am pierdut in felul acesta informatia continuta in ceilalti biti, lucru nerecomandabil: daca nu avem indicatii contrare, trebuie sa facem ca h(k) sa depinda de toti bitii lui k.
- numerele apropiate de puteri ale lui 2 sunt si ele de evitat. De exemplu: m = 2^p - 1 si cheile k sunt siruri de caractere in baza 2^p, atunci 2 siruri care difera doar printr-o transpozitie a doua caractere adiacente vor avea aceeasi valoare prin h.
- puterile lui 10 , pentru ca h(k) nu ar fi decat o parte a caracterelor ce apar in scrierea lui k in baza 10.

Valori bune pentru aceasta metoda sunt m, un numar prim si cat mai departe de puteri ale lui 2.

Tabele de dispersie (hash tables)

Functii de dispersie

Tehnici euristice de creare a unor functii de dispersie bune

Metoda multiplicarii

Fie A o constanta pozitiva subunitara, 0 < A < 1. **h(k) = [m(kA mod 1)].**
Knuth recomanda pentru A numarul de aur phi = (1+sqrt(5))/2 a carui valoare aproximata cu 7 zecimale este 1.6180339.

Metoda impachetarii
Fiecare cheie k, despre care am facut presupunerea ca este intreg, este partitionata in bucati de lungimi egale (eventual cu exceptia ultimei), k1, k2, … k*r*. Daca k este un intreg scris in baza 10 bucatile vor siruri de caractere in baza 10, de aceeasi lungime fixa, l, pe care din nou le putem considera intregi.

O functie de dispersie h : U → [0, …, 10^2 - 1], este data de formula:
h(k) = (k1 + k2 +   + kr) mod 10^l ceea ce ^nseamna ca se aduna cele r bucati, ca intregi, si din suma se pastreaza doar l cifre, cele mai putin semnicative.

Tabele de dispersie (hash tables)

Functii de dispersie

Tehnici euristice de creare a unor functii de dispersie bune

Metoda patratului

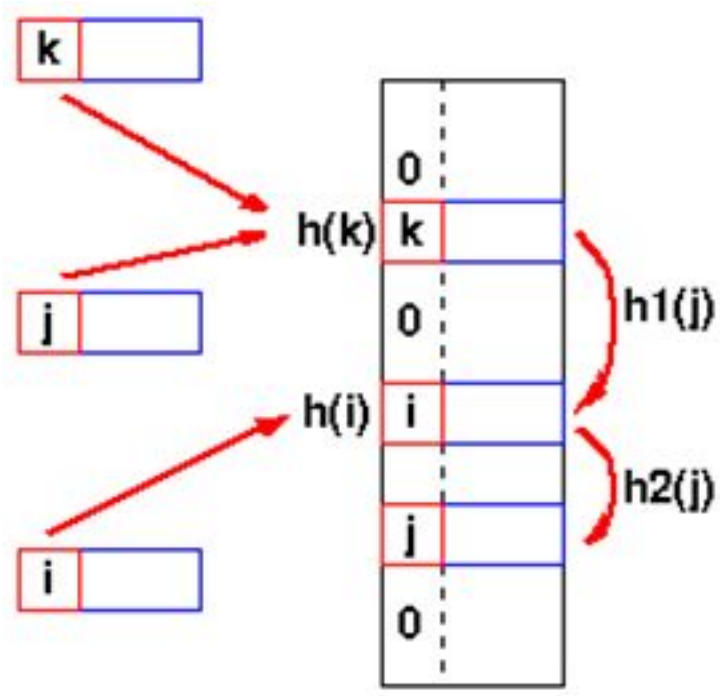
Presupunem ca l este lungimea xata a lui h(k) ca intreg in baza 10. Fie o functie de dispersie h : U → [0, …, 10^l - 1], este data de formula:
h(k) = (k^2 div 10^c1) mod 10^c2

Se ridica la patrat, iar din intregul lung astfel obtinut se elimina c1 cifre dintre cele mai putin semnicative si c2 cifre dintre cele mai semnicative, pastrandu-se exact l cifre din ,,centrul" lui k^2. Cu alte cuvinte c1 si l sunt fixate, iar c2 se obtine din formula:
l = lung(k^2) - c1 - c2.

Tabele de dispersie (hash tables)

Coliziuni

Numim coliziune a cheilor k1 si k2 situatia in care h(k1) = h(k2).

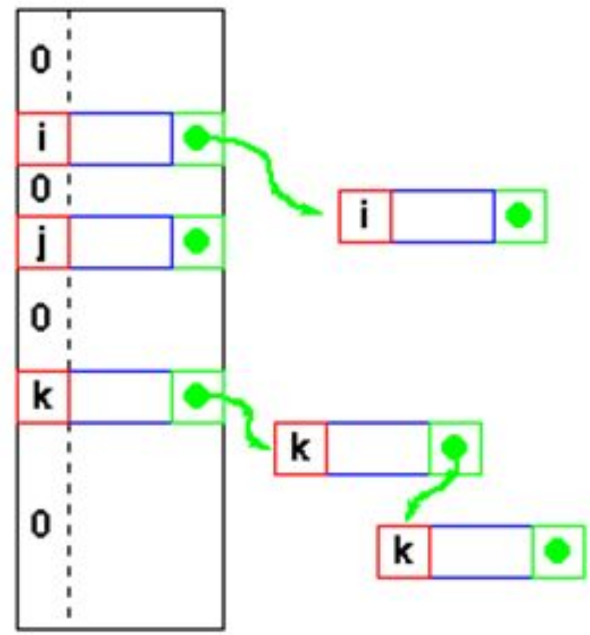


Tabele de dispersie (hash tables)

Coliziuni

Coliziunile nu pot evitate complet.
Metode de rezolvare a coliziunilor:

1) rezolvarea coliziunilor prin inlantuire (cand locatia T[h(k)] contine o lista inlantuita a tuturor inregistrarilor cu chei ce au colizionat cu k);



Inserarea si stergerea (pe o lista dublu inlantuita) O(1), dar cautarea / crearea listei O(n) + timpul de calculare al functiei de dispersie, pe cazul cel mai defavorabil.

Exemplu:
chei: 5, 28, 19, 15, 20, 33, 12, 17, 10
sloturi: 9
functia: h(k) = k mod 9.

Tabele de dispersie (hash tables)

Coliziuni

Coliziunile nu pot evitate complet.
Metode de rezolvare a coliziunilor:

2) rezolvarea coliziunilor prin adresare directa, cand se folosesc metode mai sosticate de cautare a unei locatii libere ^n T pentru o cheie care colizioneaza.

Exemple de metode:

- ***re-hashing*** presupune aplicarea în cascadă a aceleiaşi funcţii hash sau a altui model dintr-o mulţime de funcţii până când valoarea obţinută reprezintă o poziţie liberă din cadrul tabelei de dispersie;
- ***linear probing*** se bazează pe căutarea secvenţială a primei poziţii libere în care să fie inserat elementul nou, poziţie aflată la stânga sau la dreapta coliziunii; în cazul căutării, procesul presupune verificarea elementelor adiacente poziţiei indicate de valoarea hash;
- ***quadratic probing*** este o metodă de tipul linear probing care evită crearea grupurilor de coliziuni prin utilizarea unui pas de regăsire a următoarei poziţii libere diferit de 1; astfel, în caz de coliziuni au loc salturi în tabela de dispersie din două în două poziţii sau din patru în patru;

Tabele de dispersie (hash tables)

Coliziuni

- ***quadratic probing***

în general, pentru a determina următoare poziţie în care să se insereze un element nou sau în care să se caute un element existent este dat de funcţia:

$$poziţie = h(k)+c*i\hat{}2$$

unde:
poziţie – noua poziţie din tabela de dispersie în care se inserează sau se caută un element;
k – valoarea cheii asociate elementului;
h(k) – poziţia indicată de valoarea hash a elementului care se adaugă sau se caută;
c – valoare constantă definită în mulţimea {1, 2, 4};
i – numărul operaţiei de rehash sau numărul de poziţii verificate.

- ***overflow area*** presupune o abordare ce împarte tabela de dispersie în două zone, primară pentru reţinerea elementelor iniţiale şi secundară alocată elementelor ce generează coliziuni;