Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare

TABELE DE DISPERSIE

Introducere

- Tabelele de dispersie sunt structuri de date care oferă posibilitatea **inserării** și **căutării rapide**
- Indiferent de numărul de elemente, inserarea și căutarea se efectuează în timp aproape constant, adică **O(1)**

Avantaje

- Tabelele de dispersie se utilizează în programe în care este necesară căutarea într-un timp foarte scurt, prin câteva zeci de mii de elemente
- Tabelele de dispersie sunt mai rapide decât arborii, care operează într-un timp O(logN)

Dezavantaje

- Tabelele de dispersie se bazează pe tablouri, care sunt greu de extins, după ce au fost create
- În unele tipuri de tabele de dispersie, performanțele se reduc la completarea tabelei peste un anumit prag
- Programatorul trebuie să aproximeze înainte numărul elementelor pe care le va insera

Dezavantaje

- Nu există o modalitate de a vizita elementele unei tabele de dispersie într-o anumită ordine, de exemplu în ordine crescătoare
- Trebuie utilizată o altă structură pentru această facilitate

Dispersia datelor

- Dispersia datelor se referă la transformarea unui domeniu de valori ale unei anumite chei într-un domeniu de indici dintr-un tablou
- Într-o tabelă de dispersie, această operație este realizată utilizând o **funcție de dispersie** (hash function)

Observații

- Anumite tipuri de chei nu au nevoie de aplicarea acestei funcții de dispersie
- Valorile lor pot fi utilizate direct ca indici în tablou

Dicționar

- Dacă se dorește memorarea tuturor cuvintelor din limba engleză într-un dicționar, astfel încât să poată fi accesate foarte rapid, utilizarea unei tabele de dispersie este o soluție foarte bună

Compilatoare

- O aplicație a tabelelor de dispersie este în cadrul compilatoarelor limbajelor de programare, care păstrează o **tabelă de simboluri** implementată printr-o astfel de structură
- Tabela de simboluri conține toate numele variabilelor si funcțiilor definite de programator, precum și adresele la care acestea se găsesc în memorie

Exemplu

- Se dorește stocarea unui dicționar al limbii engleze, cu 50.000 de cuvinte
- Este posibil ca fiecare cuvânt să ocupe propria celulă într-un tablou cu 50.000 de elemente, sau se poate utiliza un index
- Care va fi relația dintre un cuvânt și numărul de index corespunzător ?

Transformarea cuvintelor în numere

- Este necesară o relație care să permită transformarea unui cuvânt în numărul de index corespunzător
- Codurile ASCII au valori cuprinse între 0 și 255, pentru a putea include literele mici și mari, semnele de punctuație și alte caractere

Observații

- Alfabetul limbii engleze conține 26 de litere
- Se presupune că:
  - a are asociată valoarea 1
  - b are asociată valoarea 2
  - z are asociată valoarea 26
  - spațiul are asociată valoarea 0
- Cum se combină cifrele asociate literelor într-un număr caracteristic întregului cuvânt ?

Adunarea cifrelor

- O metodă de transformare a unui cuvânt într-un număr este de a aduna codurile numerice asociate tuturor caracterelor care îl compun
- Pentru cuvântul “cast” se obține:
- c+a+s+t=3+1+19+20=43
- Cuvântul “cast” este memorat în celula cu indicele 43

Observații

- Dacă ne restrângem la cuvinte cu cel mult 10 litere, primul cuvânt din dicționar, a, este reprezentat prin valoarea:
- 0+0+0+0+0+0+0+0+0+1=1
- Ultimul cuvânt potențial din dicționar este zzzzzzzzzz
- Codul obținut prin adunarea literelor este:
- 26+26+26+26+26+26+26+26+26+26=260

Observații

- Domeniul global al codurilor este de la 1 la 260
- Deoarece există 50.000 de cuvinte în dicționar, nu dispunem de suficienți indici pentru a putea asocia fiecărui cuvânt un indice propriu
- Fiecare element al tabloului va conține aproximativ 192 de cuvinte (50.000/260)

Observații

- Se poate introduce un subtablou sau o listă înlănțuită de cuvinte în fiecare din elementele tabloului
- Această soluție duce la o reducere a vitezei de acces
- Deși avem acces imediat la orice element din tablou, căutarea cuvântului dorit prin lista de 192 de cuvinte este lentă

Forma polinomială

- Fiecare caracter dintr-un cuvânt trebuie să contribuie în mod unic la numărul rezultat în final, pentru ca fiecare cuvânt să ocupe singur o celulă din tablou
- Se poate efectua descompunerea unui cuvânt în litere, după care se convertesc literele în cifrele echivalente, se înmulțesc cu puterile corespunzătoare ale lui 27 și se adună rezultatele

Exemplu

- Prin transformarea cuvântului “cast” într-un număr, se obține:
- $3*27^3+1*27^2+19*27^1+20*27^0=60.311$
- Această metodă generează un număr unic pentru fiecare cuvânt
- Pentru cuvinte mai lungi, domeniul de indici devine prea mare

Dezavantaje

- Această metodă asociază o celulă dintr-un tablou fiecărui șir de caractere, chiar dacă acesta reprezintă sau nu un cuvânt
- Numai o mică parte din celule sunt asociate unor cuvinte reale, deci majoritatea celulelor din tablou vor fi vide

Observații

- Metoda adunării cifrelor a generat prea puțini indici
- Metoda reprezentării polinomiale a generat prea mulți indici

Dispersie

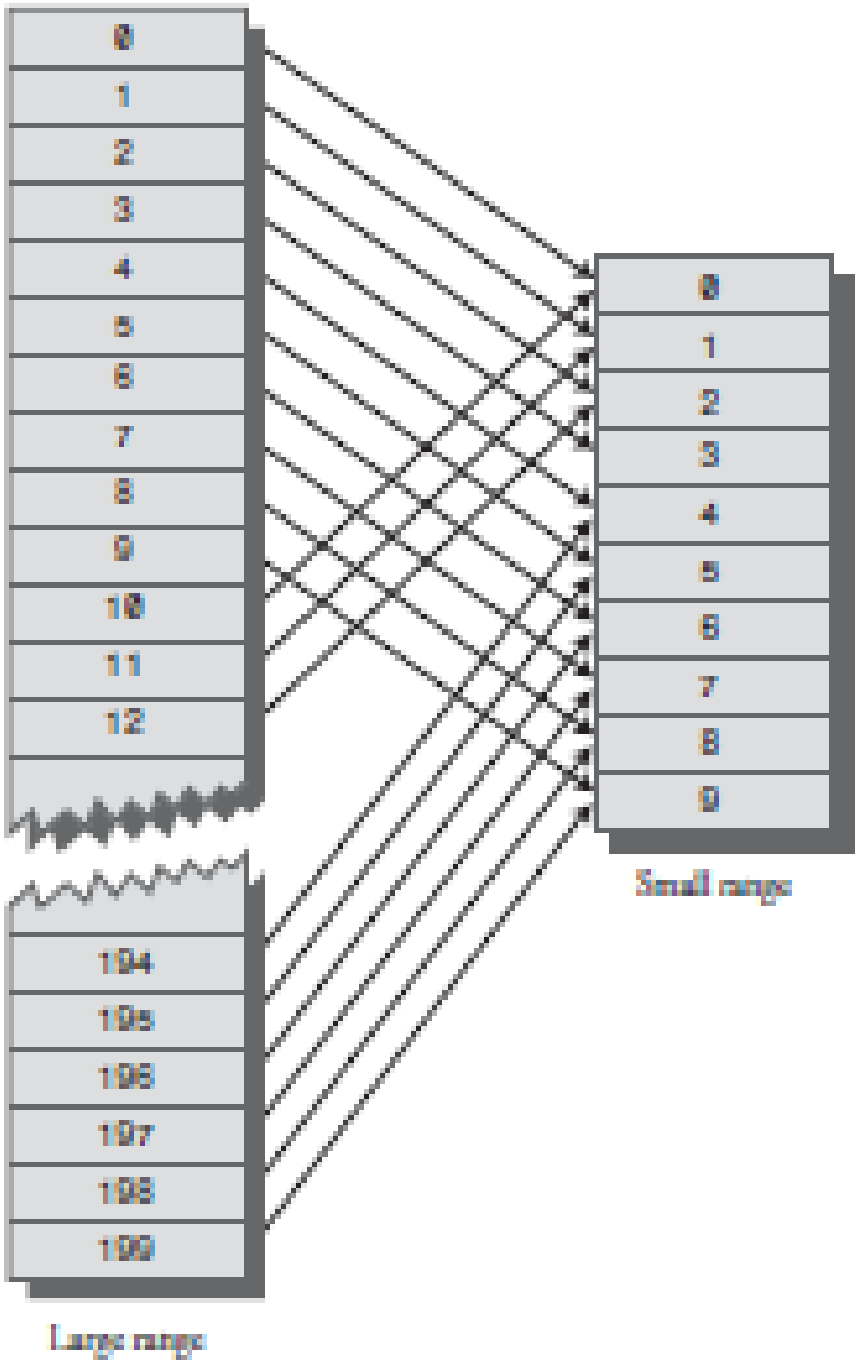
- Este necesară o modalitate de a compacta domeniul de valori obținut în urma reprezentării polinomiale, pentru a obține un domeniu care să se apropie de dimensiunile unui tablou rezonabil
- Cât de mare va fi tabloul necesar pentru exemplul dicționarului limbii engleze ?

Dispersie

- Este necesar un tablou cu dimensiunea **dublă** față de numărul de cuvinte, adică un tablou cu 100.000 de elemente
- Se poate reduce domeniul obținut în urma reprezentării polinomiale într-un domeniu cuprins între 0 și 100.000, prin utilizarea operatorului modulo (**%**)

Exemplu

- Se restrâng numerele 0-199 (desemnate prin variabila *largeNumber*) în domeniul 0-9 (desemnate prin variabila *smallNumber*)
- Există 10 valori în domeniul mai mic și se definește variabila *smallRange*=10
- *smallNumber = largeNumber % smallRange*


0
1
2
3
4
5
6
7
8
9
10
11
12
194
195
196
197
198
199
Large range
0
1
2
3
4
5
6
7
8
9
Small range

Observații

- Se compactează domeniul, cu un **raport de compresie** de 20 la 1
- Pentru a comprima numerele care reprezintă, în mod unic, fiecare cuvânt din limba engleză, în numere de index adecvate tabloului, avem:
- *arrayIndex = hugeNumber % arraySize*

Funcție de dispersie

- Funcția de dispersie convertește un număr dintr-un domeniu mai mare într-un număr dintr-un domeniu mai mic
- Domeniul mai mic corespunde cu domeniul indicilor dintr-un tablou
- Un tablou în care elementele sunt inserate utilizând o funcție de dispersie se numește **tabelă de dispersie**

Observații

- Utilizând operatorul **%**, se comprimă domeniul foarte mare într-un domeniu cu lungimea dublă față de numărul elementelor care trebuie memorate
- Exemplu de funcție de dispersie:
- *arraySize = numberWords * 2*
- *arrayIndex = hugeNumber % arraySize*

Observații

- În domeniul foarte mare, fiecare număr reprezintă un șir de caractere, dar puține dintre aceste șiruri sunt cuvinte cu sens din limba engleză
- Funcția de dispersie transformă aceste valori mari în numere de index dintr-un domeniu mult mai mic

Observații

- În acest tablou, există în medie un cuvânt la fiecare două celule
- Unele celule rămân neocupate, dar există și celule cu mai mult de un cuvânt
- O implementare directă a acestei metode duce la depășirea de către variabila *hugeNumber* a dimensiunii maxime a tipului său

Coliziuni

- Restrângerea domeniului mai mare are un dezavantaj
- Nu este sigur că, pentru două cuvinte distincte, nu se va asocia aceeași celulă din tablou (funcția de dispersie nu este injectivă)

Coliziuni

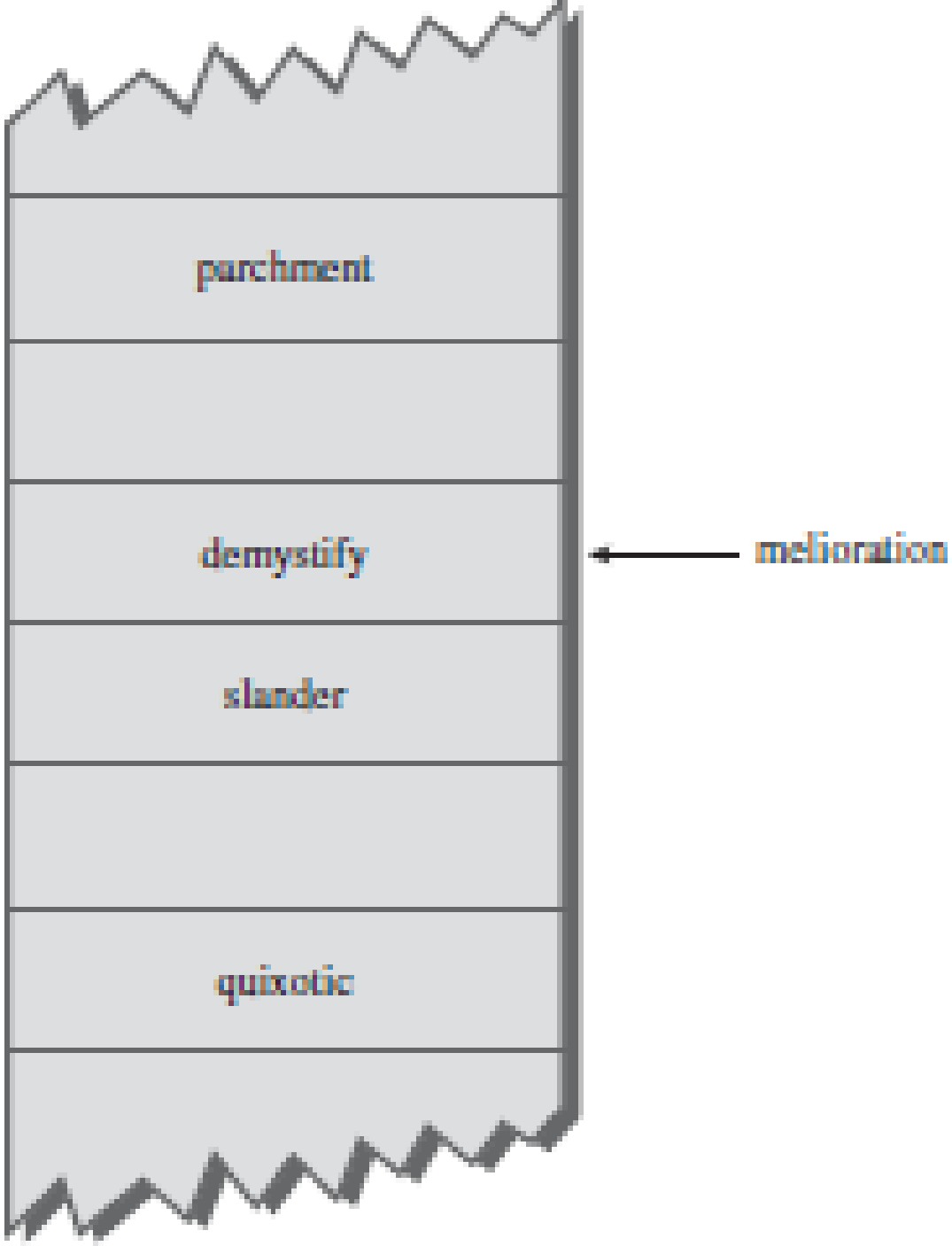
- Situația seamănă cu cazul în care se adunau codurile literelor, dar este mai bună
- Pentru cuvinte de până la 10 litere, existau numai 260 de rezultate posibile
- Numărul rezultatelor este acum de 50.000
- Este imposibil să se evite asocierea multiplă a diferitelor cuvinte în aceeași celulă din tablou

Coliziuni

- Nu este posibil ca fiecare indice să fie asociat unui cuvânt unic
- Este de dorit să nu existe foarte multe cuvinte care să fie repartizate în aceeași celulă

Exemplu

- Se presupune că se inserează cuvântul “melioration” în tablou
- Se aplică funcția de dispersie, pentru a obține numărul indicelui, dar se găsește celula ocupată de cuvântul “demystify”, acestuia fiindu-i asociat același indice
- Această situație se numește **coliziune**


parchment
demystify
melioration
slander
quixotic

Observații

- Deoarece tabloul are de două ori mai multe celule decât elemente, jumătate din celulele tabloului sunt neocupate
- O soluție este ca, la apariția unei coliziuni, să se caute în tablou (într-un mod stabilit) o celulă liberă, după care se inserează elementul nou în acea celulă, în locul celei obținute prin aplicarea funcției de dispersie

Adresare deschisă

- Această metodă se numește **adresare deschisă** (open addressing)
- Dacă cuvântului “cast” îi corespunde celula 5.421 și aceasta este deja ocupată, se inserează cuvântul în celula 5.422

Înlănțuire separată

- A doua metodă este de a crea un tablou care conține liste înlănțuite în loc de cuvinte
- La apariția unei coliziuni, elementul nou va fi inserat într-o listă de cuvinte cu același indice
- Metoda se numește **înlănțuire separată** (separate chaining)

Adresare deschisă

- În această metodă, când nu se poate insera un element la valoarea indicelui calculat de funcția de dispersie, se caută altă poziție din tablou

Tipuri de adresare deschisă

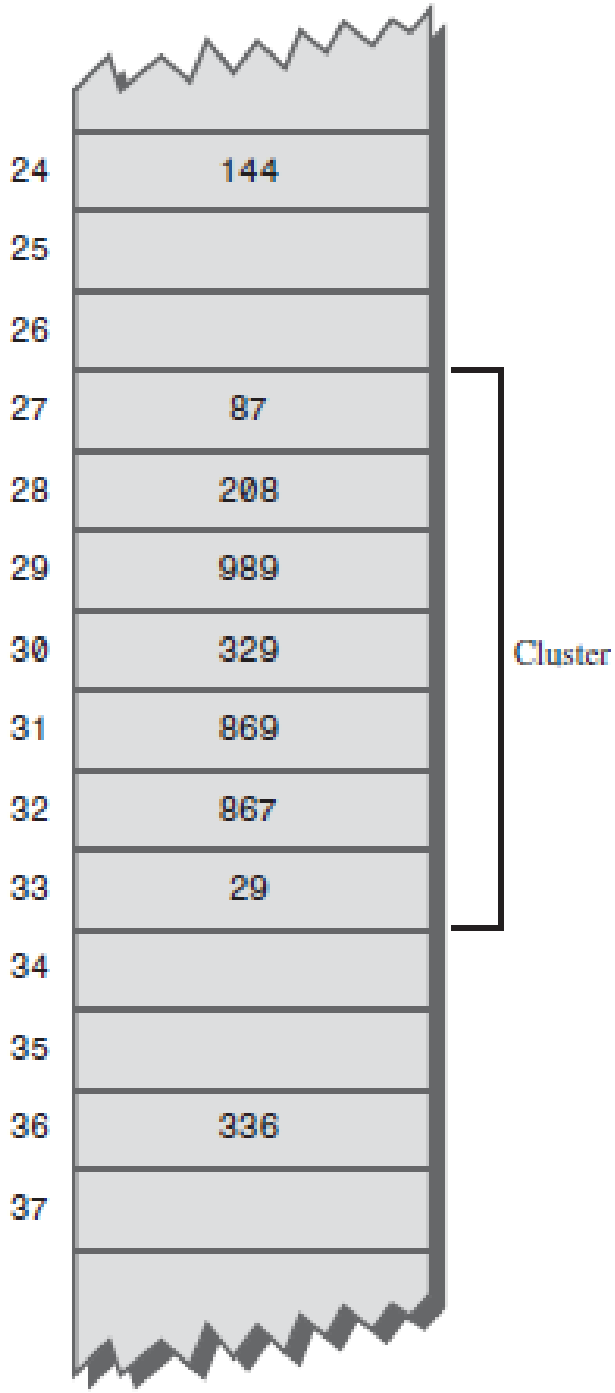
- Există trei metode de adresare deschisă, care diferă prin modul de căutare a următoarei celule libere:
  - **sondaj liniar** (linear probing)
  - **sondaj pătratic** (quadratic probing)
  - **dublă dispersie** (double hashing)

Sondaj liniar

- Metoda caută secvențial o celulă liberă
- Se incrementează indexul, până când se găsește o celulă liberă
- Metoda se numește **sondaj liniar**, deoarece parcurge secvențial mulțimea celulelor din tablou

Observații

- O secvență de celule ocupate dintr-o tabelă de dispersie se numește **secvență ocupată** (filled sequence)
- Pe măsură ce se adaugă elemente noi, secvențele ocupate devin din ce în ce mai lungi
- Acest proces se numește **creare de unități de alocare** (clustering)


24
144
25
26
27
87
28
208
29
989
30
329
31
869
32
867
33
29
34
35
36
336
37
Cluster

Observații

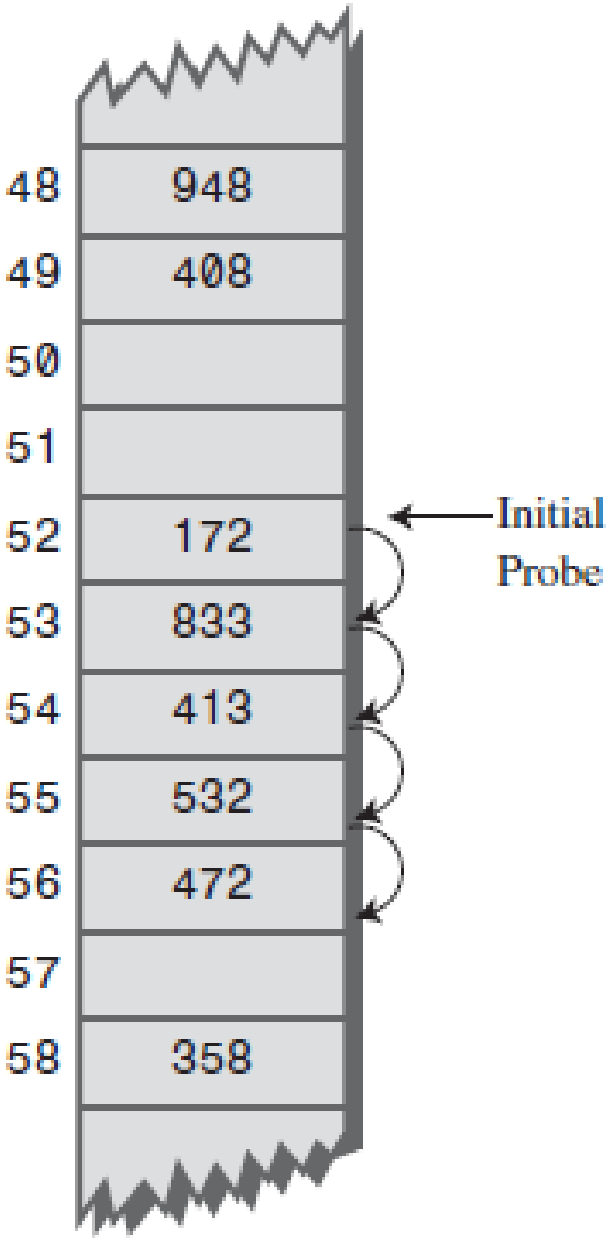
- Umplerea aproape completă a unei tabele de dispersie este ineficientă
- Când o tabelă de dispersie se umple complet, niciunul dintre algoritmi nu mai funcționează

Sondaj

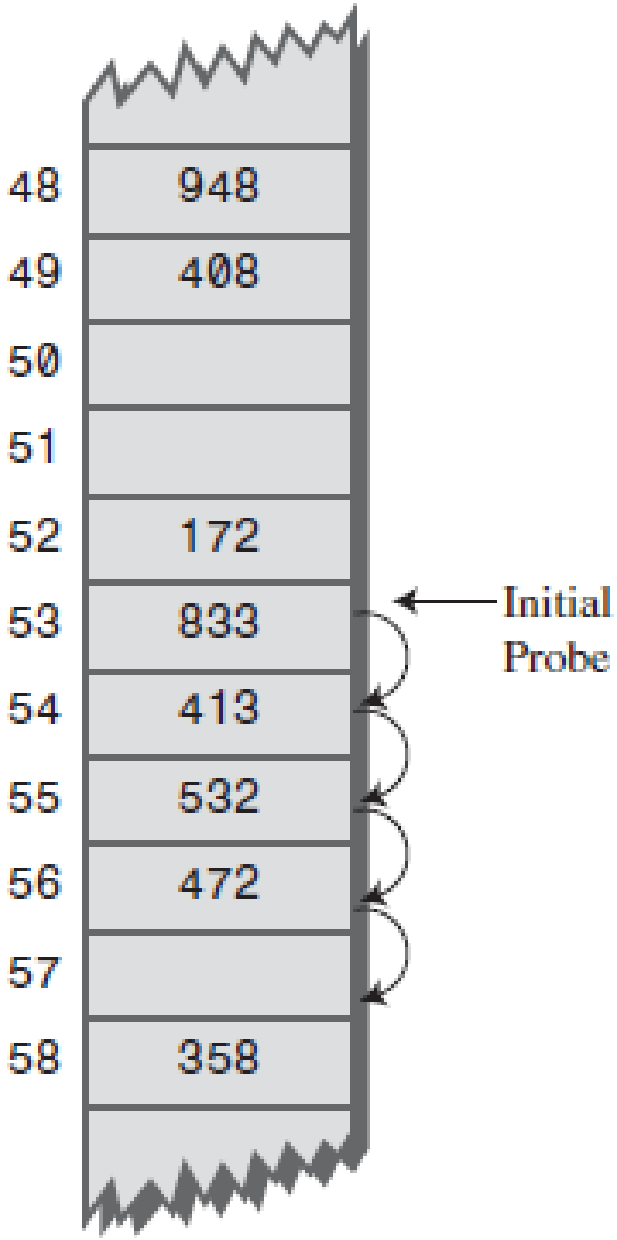
- La căutarea unei chei, se aplică funcția de dispersie și se obține un indice din tablou
- Dacă celula cu indicele respectiv este ocupată de un element cu o altă cheie, apare o coliziune
- Algoritmul va căuta cheia în următoarea celulă din tablou
- Procesul de căutare a celulei potrivite, în urma unei coliziuni, se numește **sondaj** (probing)

Observații

- După detectarea coliziunii, algoritmul va continua să examineze în ordine celulele din tablou
- Dacă se întâlnește o celulă liberă (înainte de a găsi cheia căutată), înseamnă că operația a eșuat



a) Successful search for 472



b) Unsuccessful search for 893

Observații

- Numărul pașilor efectuați pentru găsirea unei celule libere se numește **lungimea sondajului** (probe length)
- În majoritatea cazurilor, lungimea sondajului este de câteva celule

Fascicule

- Pe măsură ce se umple tabela, fasciculele de elemente devin mai mari
- Crearea de fascicule lungi conduce la creșterea lungimilor de sondaj
- Accesul la elementele de la sfârșitul secvenței devine lent

Observații

- Cu cât tabela este mai plină, cu atât efectele creării de fascicule devin mai neplăcute
- Nu este nicio problemă dacă tabela este plină pe jumătate, până la o proporție de două treimi
- Peste această valoare, performanțele se reduc, iar fasciculele devin tot mai mari

Observații

- În cazul tabelelor cu **adresare deschisă**, care utilizează metoda **sondajului liniar**, pot apărea **fascicule** de elemente
- După ce se formează, fasciculele tind să crească
- Elementele noi se vor insera la sfârșitul secvenței, care va crește

Coeficient de încărcare

- Raportul dintre numărul de elemente dintr-o tabelă și dimensiunea tabelei se numește **coeficient de încărcare** (load factor)
- O tabelă cu 10.000 de celule și 6.667 de elemente are un coeficient de încărcare de 2/3
- *loadFactor = noItems / arraySize*

Observații

- Fasciculele se pot forma chiar atunci când coeficientul de încărcare are valori scăzute
- O parte din tabelă poate conține fascicule lungi, cealaltă parte rămânând aproape neocupată
- Prezența fasciculelor conduce la scăderea performanțelor tabelei

Exemplu

- Se inserează cheile 21, 32 și 31
- Presupunem *tableSize = 5*
- hashIndex pt 21 = 21 % 5 = 1
- hashIndex pt 32 = 32 % 5 = 2
- hashIndex pt 31 = 31 % 5 = 1
- Avem coliziune pentru 31 și 21

Exemplu

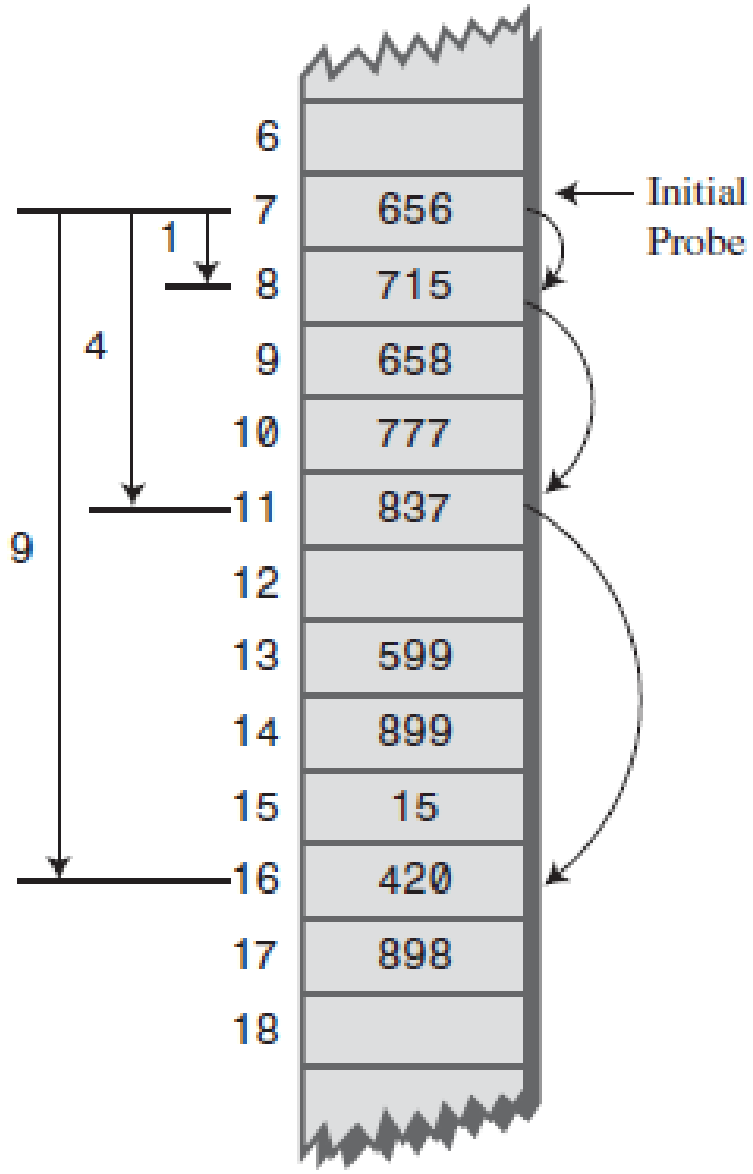
- Se caută următoarea celulă liberă
- hashIndex pt 31 = (31+1) % 5 = 2
- hashIndex cu valoarea 2 este deja ocupat de cheia 32
- hashIndex pt 31 = (31+2) % 5 = 3
- hashIndex cu valoarea 3 este liber

Sondaj pătratic

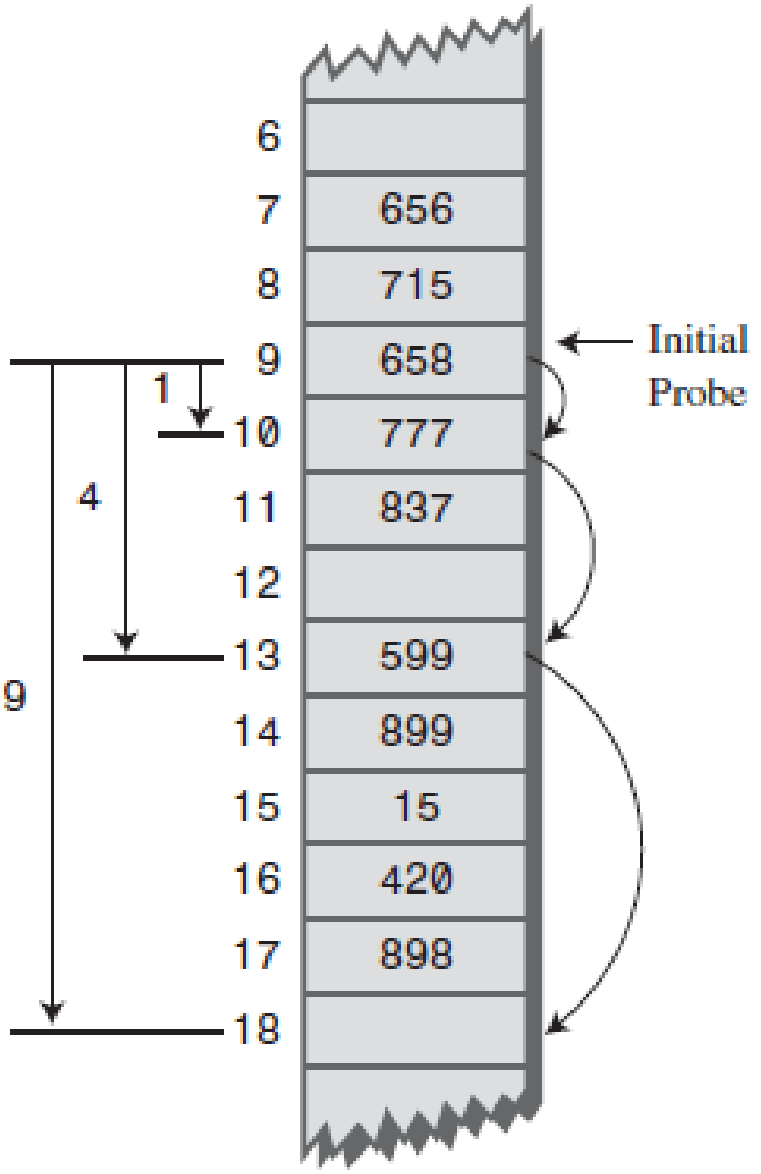
- O posibilitate de prevenire a formării fasciculelor este utilizarea **sondajului pătratic**
- Ideea este de a sonda celule distribuite pe o arie ceva mai largă, în locul celor imediat adiacente locului în care trebuie efectuată inserarea

Pasul de incrementare este pătratul numărului iterației

- Într-un sondaj liniar, dacă indicele inițial de dispersie este x, celule sondate ulterior vor fi x+1, x+2, x+3, ...
- Sondajul pătratic presupune examinarea celulelor x+1, x+4, x+9, x+16, ...
- Distanța dintre acestea și celula inițială este egală cu pătratul numărului iterației curente



a) Successful search for 420



b) Unsuccessful search for 481

Observații

- Metoda sondajului pătratic mărește pasul de căutare la fiecare iterație efectuată
- În prima iterație, algoritmul alege celula adiacentă
- Dacă aceasta este ocupată, algoritmul presupune că se găsește într-un fascicul și sare la o distanță de 4 celule de locul inițial

Observații

- Dacă se găsește o celulă ocupată, se presupune că fasciculul în care se află este ceva mai mare și se încearcă la o distanță de 9 celule
- Dacă și această celulă este ocupată, algoritmul se deplasează la o distanță de 16 celule
- Algoritmul caută un loc liber, pentru a insera elementul

Observații

- Este bine ca dimensiunea tabelei să fie un număr prim
- Dacă dimensiunea tabelei nu este număr prim, este posibilă apariția unei secvențe infinite de iterații în algoritmul de sondaj

Dezavantajul sondajului pătratic

- Sondajul pătratic rezolvă problema fasciculelor, care apărea în cazul sondajului liniar
- Această problemă se numește **acumulare primară** (primary clustering)
- Sondajele pătratice produc un tip diferit (și mult mai greu de detectat) de acumulări

Observații

- Aceste acumulări apar datorită faptului că, pentru toate cheile asociate unei anumite celule, se parcurge aceeași secvență, în căutarea unui loc liber
- Se presupune că valorile 184, 302, 420 și 544 au toate asociate celula 7 și sunt inserate în această ordine

Exemplu

- Elementul 302 va necesita un sondaj cu o singură iterație, pentru 420 vor fi necesare două iterații, iar pentru 544, trei iterații
- Fiecare element nou asociat celulei 7 va determina creșterea numărului de iterații în sondaj
- Acest fenomen se numește **acumulare secundară** (secondary clustering)

Dubla dispersie

- Pentru a elimina acumularea primară, precum și cea secundară, se utilizează **dubla dispersie**
- Este necesară o modalitate de a genera secvențe de sondaj, care depind de fiecare cheie în parte, în locul celor invariante față de cheie
- Valorile diferite asociate aceluiași indice vor utiliza secvențe de sondaj diferite

Dubla dispersie

- Soluția este de a trece cheia prin funcția de dispersie pentru a doua oară, cu o altă funcție de dispersie, utilizând rezultatul obținut ca pas de deplasare
- Pentru o aceeași cheie, pasul va rămâne constant pe parcursul unui sondaj, modificându-se însă pentru o altă cheie

Dubla dispersie

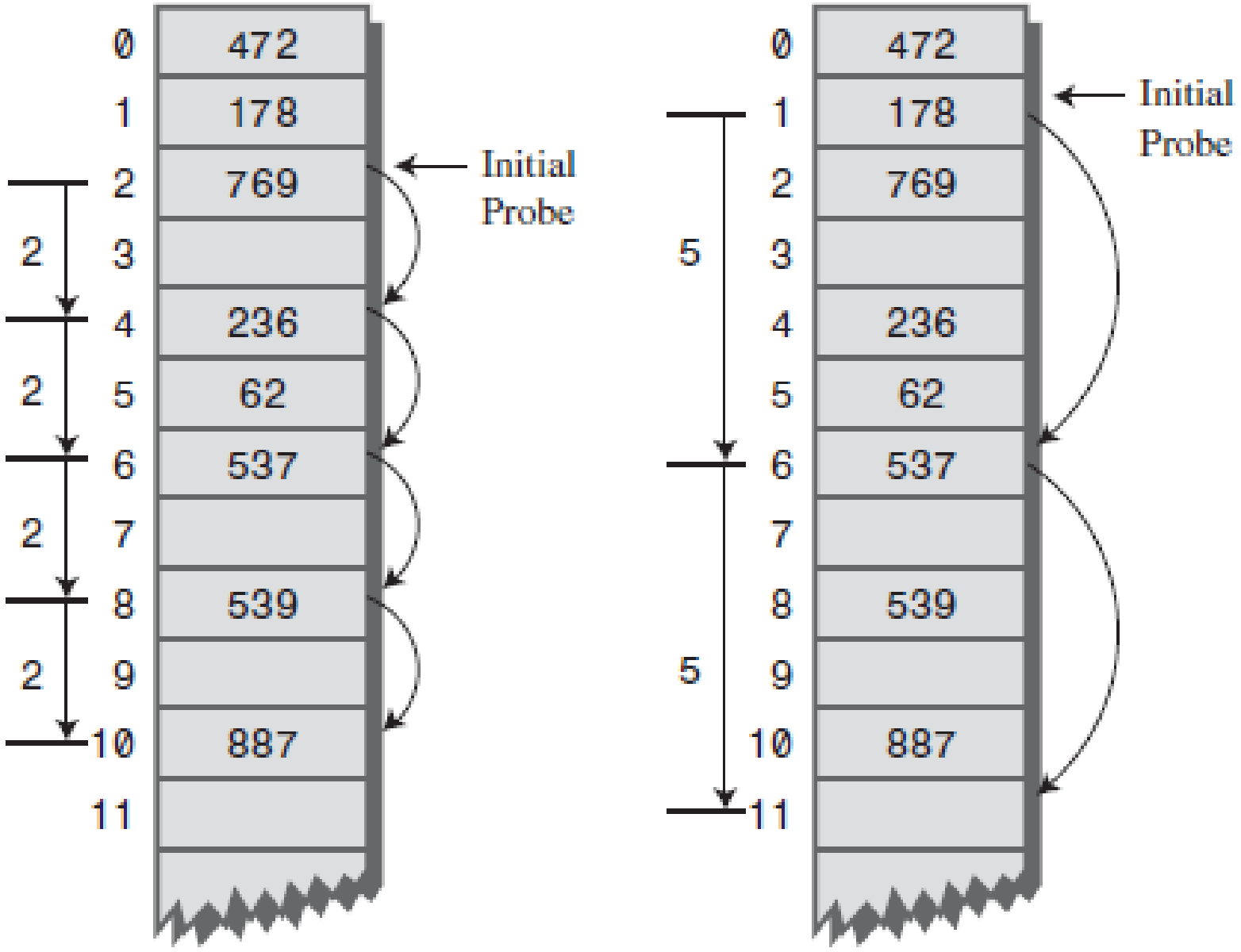
- Proprietățile funcțiilor secundare de dispersie sunt:
- **1**. Funcția secundară nu trebuie să coincidă cu cea primară
- **2**. Rezultatul funcției nu trebuie să fie niciodată 0 (care corespunde unei deplasări nule – toate sondajele ar staționa pe aceeași celulă, iar algoritmul ar intra în buclă infinită)

Dubla dispersie

- Funcțiile de forma următoare dau rezultate bune:
- *stepSize = constant – (key % constant)*
- unde valoarea *constant* este primă și mai mică decât dimensiunea tabloului
- De exemplu:
- *stepSize = 5 – (key % 5)*

Observații

- Pentru o cheie dată, toate iterațiile vor determina o deplasare cu un pas constant, dar o altă cheie va modifica acest pas
- În cazul funcției prezentate, pașii de deplasare sunt cuprinși între 1 și 5



a) Successful search for 887

b) Unsuccessful search for 709

Observații

- Chiar la valori mari ale coeficientului de încărcare, majoritatea elementelor vor fi găsite în pozițiile inițiale, utilizând numai prima funcție de dispersie
- Puține elemente vor necesita secvențe mari de sondaj

Dimensiunea tabelei trebuie să fie un număr prim

- Metoda dublei dispersii cere ca dimensiunea tabelei să fie un număr prim
- Pentru a justifica alegerea, să considerăm o situație în care această condiție nu este îndeplinită
- Să presupunem că dimensiunea tabelei este 15 (cu indici de la 0 la 14)

Exemplu

- Să presupunem că unei anumite chei îi corespunde indicele inițial 0 și pasul de deplasare 5
- Secvența de sondaj va fi:
- 0, 5, 10, 0, 5, 10, ..., repetându-se la infinit
- Singurele celule examinate sunt cele cu indicii 0, 5 și 10

Exemplu

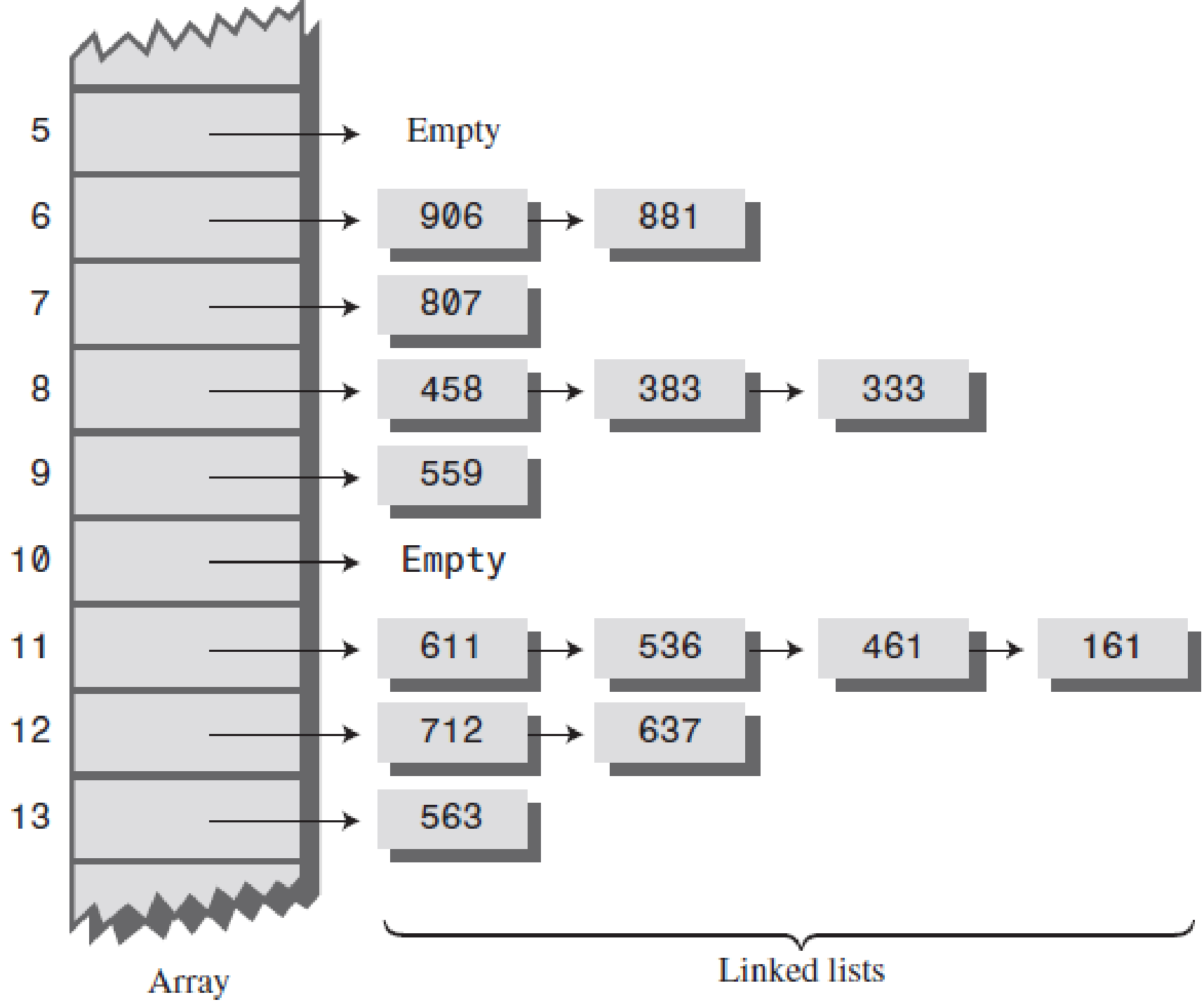
- Algoritmul nu va detecta niciodată eventualele celule libere, cu indicii 1, 2, 3, ...
- Algoritmul intră într-o buclă infinită
- Utilizând un număr prim ca dimensiune a tabelei, ne asigurăm că aceasta nu este un multiplu al pasului de deplasare, deci secvența de sondaj va vizita toate celulele din tabelă

Înlănţuire separată

- În metoda adresării deschise, coliziunile sunt rezolvate prin căutarea unei celule libere în tabela de dispersie
- O abordare diferită este ca fiecare celulă din tabelă să conţină o listă înlănţuită
- Cheia unui anumit element este transformată într-un indice, prin aplicarea funcţiei de dispersie, elementul fiind apoi inserat în lista înlănţuită din celula cu indicele calculat anterior

Înlănţuire separată

- Celelalte elemente, cărora le va corespunde acelaşi indice, vor fi inserate în aceeaşi listă
- Nu va fi nevoie de căutarea unei celule libere în tabelă
- Înlănţuirea separată este mai simplă, din punct de vedere conceptual, decât adresarea deschisă


5
Empty
6
906
881
7
807
8
458
383
333
9
559
10
Empty
11
611
536
461
161
12
712
637
13
563
Array
Linked lists

Coeficient de încărcare

- Coeficientul de încărcare (raportul dintre numărul de elemente dintr-o tabelă de dispersie şi dimensiunea maximă a tabelei) are valori diferite faţă de cazul adresării deschise
- În metoda înlănţuirii separate, este normal ca o tabelă cu *N* celule să conţină cel puţin *N* elemente

Coeficient de încărcare

- Coeficientul de încărcare va avea valori supraunitare
- Această încărcare nu reprezintă o problemă
- Vor exista anumite celule cu cel puţin două elemente în listele corespunzătoare

Observaţii

- În cazul în care listele conţin multe elemente, timpul de acces creşte, din cauză că accesul la un anumit element presupune parcurgerea, în medie, a jumătate din elementele listei

Observaţii

- Găsirea celulei iniţiale se efectuează într-un timp scurt O(1), dar căutarea într-o listă necesită un timp proporţional cu numărul de elemente din listă, O(M)
- Este bine ca listele să nu fie foarte încărcate

Observaţii

- În metoda adresării deschise, performanţele se reduc când coeficientul de încărcare depăşeşte valoarea 2/3
- Înlănţuirea separată permite o creştere a acestui factor la valori supraunitare, fără a afecta prea mult performanţele tabelei de dispersie

Valori multiple

- Valorile multiple sunt permise şi pot fi generate în procesul de umplere a tabelei
- Toate elementele cu o aceeaşi cheie vor fi inserate în aceeaşi listă
- Pentru a le găsi, este necesar să parcurgem toată lista şi în cazul în care operaţia reuşeşte
- Aceasta conduce la o scădere a performanţelor

Dimensiunea tabelei

- Dimensiunea tabelei nu mai trebuie să fie în acest caz un număr prim, cum era în cazul sondajului pătratic şi dublei dispersii
- Nemaiexistând sondaje, a dispărut pericolul ca un sondaj să intre într-o buclă infinită, din cauză că dimensiunea tabelei este divizibilă cu pasul de deplasare

Funcţii de dispersie

- O funcţie de dispersie trebuie să fie simplă, pentru a fi calculată rapid
- Avantajul tabelelor de dispersie îl reprezintă viteza de acces
- Dacă funcţia de dispersie este lentă, perfomanţele se reduc

Funcţii de dispersie

- Scopul unei funcţii de dispersie este de a transforma valorile unor chei dintr-un domeniu precizat în valori ale unor indici dintr-o tabelă, distribuite aleator şi cât mai uniform
- Cheile pot fi, la rândul lor, mai mult sau mai puţin aleatoare

Chei aleatoare

- O funcţie perfectă de dispersie asociază fiecărei chei un indice diferit în tabelă
- Acest lucru este posibil numai pentru secvenţe de chei cuprinse într-un domeniu suficient de mic, pentru a putea fi utilizate direct ca indici într-un tablou
- În majoritatea cazurilor, aceste condiţii nu sunt îndeplinite, funcţiile de dispersie având rolul de a comprima un domeniu mai mare de chei într-unul mai mic de indici

Chei aleatoare

- Distribuţia cheilor determină sarcinile pe care funcţia de dispersie trebuie să le îndeplinească
- Se presupune că datele sunt aleator distribuite în întreg domeniul
- În această situaţie, funcţia:
- *index = key % arraySize*
- este suficient de bună

Eficiența tabelelor de dispersie

- Inserarea și căutarea într-o tabelă de dispersie se pot efectua într-un timp apropiat de cel constant, O(1)
- Dacă nu apar coliziuni, inserarea unui element nou sau căutarea unuia existent se reduc la un apel al funcției de dispersie și la un acces în tabelă
- Acesta este timpul de acces minim

Eficiența tabelelor de dispersie

- În cazul apariției coliziunilor, timpul de acces depinde de lungimea secvențelor de sondaj rezultate
- Fiecare acces la o celulă, în procesul de sondaj, crește timpul de căutare pentru o celulă liberă (în cazul inserării) sau pentru o celulă existentă în tabelă

Eficiența tabelelor de dispersie

- Un acces presupune detectarea celulelor libere, iar în cazul căutării sau ștergerii, comparația dintre valoarea din celula curentă și valoarea dorită
- Timpul operației de căutare sau de inserare este direct proporțional cu lungimea sondajului efectuat
- Acesta se adună la timpul constant, necesar calculului funcției de dispersie

Eficiența tabelelor de dispersie

- Lungimea medie a sondajului (timpul mediu de acces) depinde de coeficientul de încărcare (raportul dintre numărul de elemente din tabelă și dimensiunea tabelei)
- Pe măsură ce coeficientul de încărcare crește, secvențele de sondaj devin din ce în ce mai lungi

Adresare deschisă

- Pierderea de eficiență, la valori mari ale coeficienților de încărcare, este mult mai mare pentru diferitele metode de adresare deschisă, decât în cazul înlănțuirii separate
- În adresarea deschisă, căutările nereușite durează mai mult decât cele reușite

Adresare deschisă

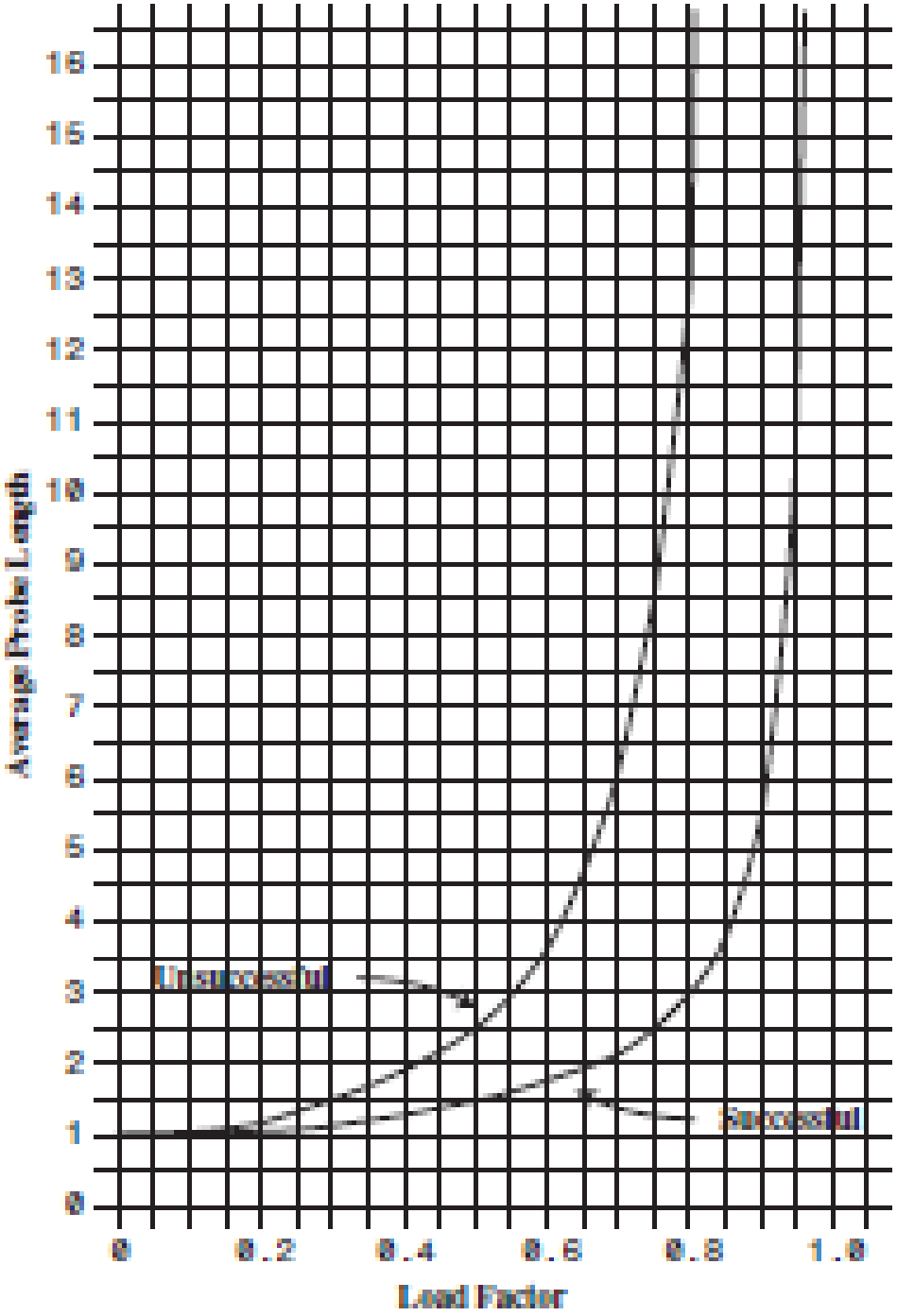
- Pe parcursul unei secvențe de sondaj, algoritmul se oprește, imediat ce găsește elementul dorit, ceea ce se întâmplă, în medie, la jumătate din lungimea totală a secvenței
- Într-o căutare nereușită, algoritmul trebuie să parcurgă toată secvența, nefiind sigur dacă va găsi sau nu elementul

Sondaj liniar

- Relația dintre lungimea sondajului (*P*) și coeficientul de încărcare (*L*), în cazul sondajului liniar, este:
- $P = (1 + 1 / (1 – L)) / 2$
- Pentru o operație nereușită, avem:
- $P = (1 + 1 / (1 – L)^2) / 2$

Observații

- La un coeficient de încărcare de 1/2, o căutare reușită presupune efectuarea a 1,5 comparații, în timp ce una nereușită necesită 2,5 comparații
- La un coeficient de încărcare de 2/3, o căutare reușită presupune efectuarea a 2 comparații, în timp ce una nereușită necesită 5 comparații


16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Average Probe Length
Unsuccessful
Successful
0
0.2
0.4
0.6
0.8
1.0
Load Factor

Observații

- Coeficientul de încărcare nu trebuie să depășească 2/3 sau, dacă este posibil, 1/2
- Cu cât coeficientul de încărcare este mai mic, cu atât se consumă mai multă memorie, pentru a stoca un anumit volum de date
- Valoarea optimă a coeficientului de încărcare depinde de compromisul dintre eficiența utilizării memoriei, care scade la valori mici ale coeficientului, și viteza, care crește

Sondaj pătratic și dubla dispersie

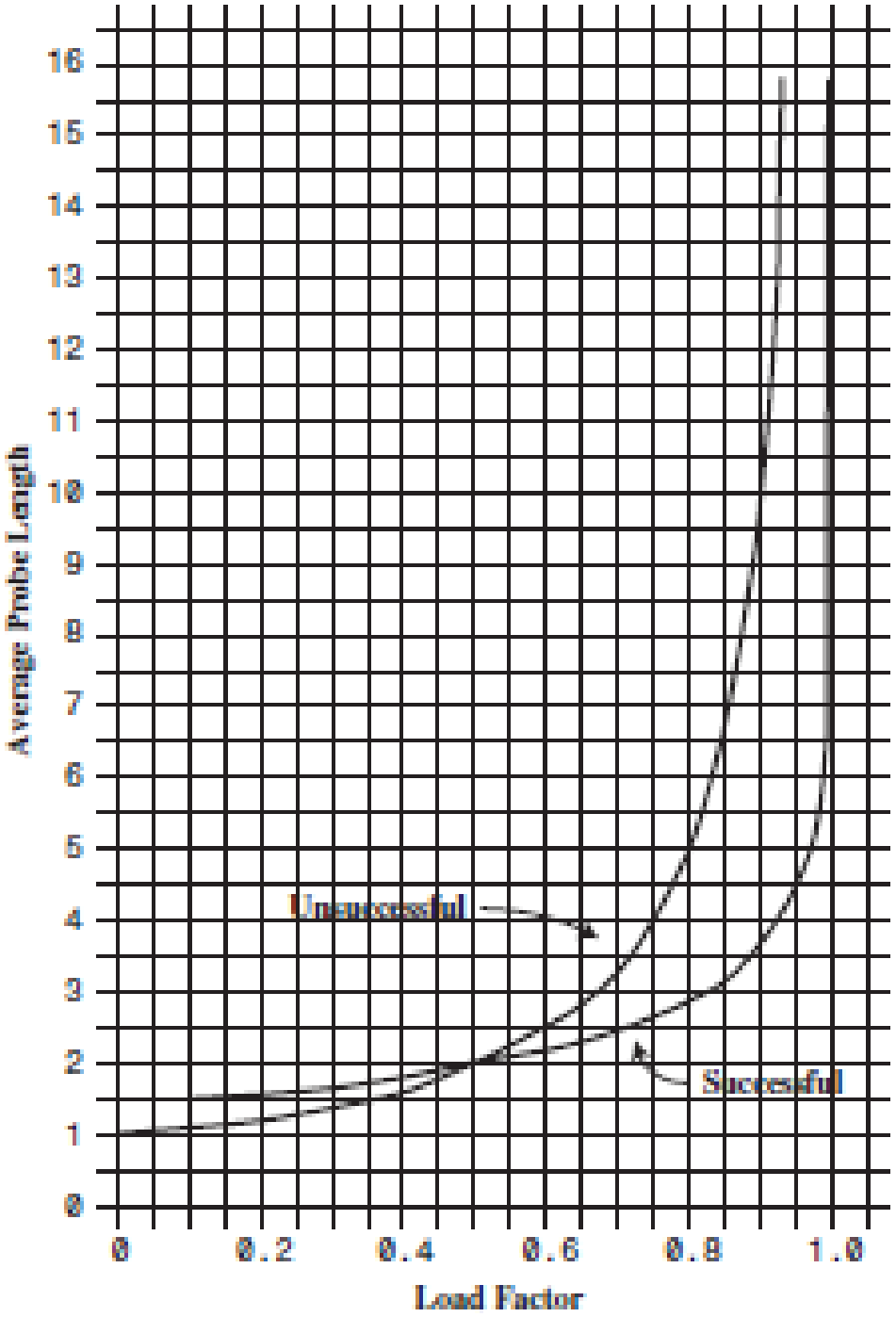
- Performanțele pentru metodele de sondaj pătratic și dubla dispersie sunt descrise prin ecuații comune
- Acestea indică o superioritate ușoară față de sondajul liniar

Sondaj pătratic și dubla dispersie

- În cazul unei căutări reușite, avem:
- $P = -log_2(1 - L) / L$
- În cazul unei căutări nereușite, avem:
- $P = 1 / (1 - L)$

Observații

- La un coeficient de încărcare de 1/2, atât o căutare reușită, cât și una nereușită, necesită, în medie, două sondaje
- La un coeficient de încărcare de 2/3, valorile sunt 2,37 și 3
- La un coeficient de încărcare de 4/5, valorile sunt 2,9 și 5


16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Average Probe Length
Unsuccessful
Successful
0
0.2
0.4
0.6
0.8
1.0
Load Factor

Observații

- Sondajul pătratic și dubla dispersie permit valori ceva mai mari ale coeficientului de încărcare, fără o scădere a performanțelor

Inlănțuire separată

- Vrem să aflăm cât durează inserarea unui element într-o tabelă de dispersie cu înlănțuire separată
- Presupunem că operația cea mai consumatoare de timp este comparația cheii elementului cu celelalte chei din listă

Inlănțuire separată

- Presupunem că timpul necesar pentru determinarea listei potrivite și pentru detectarea sfârșitului unei liste este egal cu timpul necesar unei comparații
- Durata totală a operației este:
- *1 + noComps*
- unde *noComps* reprezintă numărul comparațiilor de chei

Observații

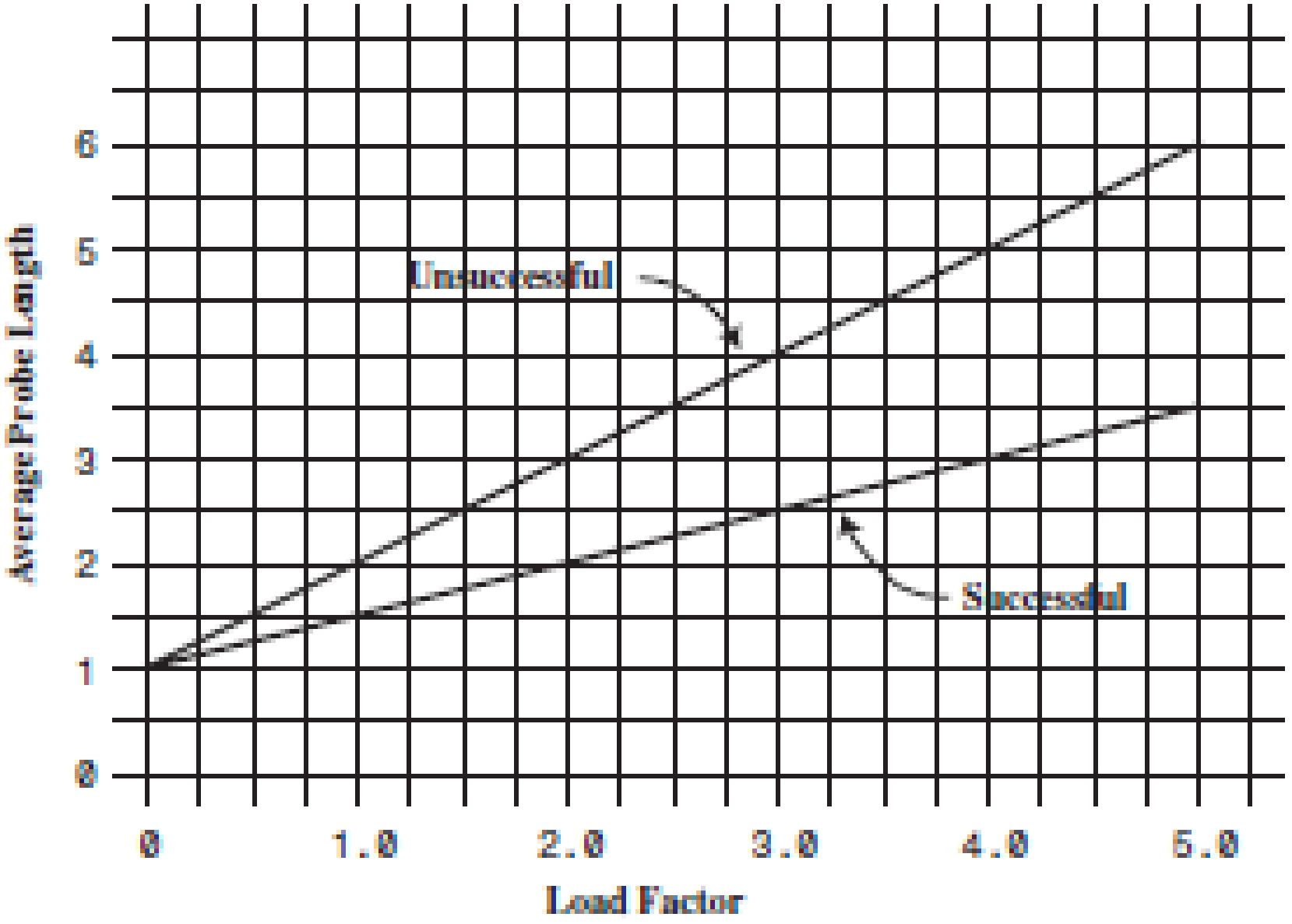
- Presupunem că tabela are *arraySize* elemente, fiecare dintre acestea conținând o listă, iar numărul elementelor inserate în tabelă este *N*
- *Lungime medie a listei = N / arraySize*
- *L = N / arraySize*
- Lungimea medie a unei liste este egală cu coeficientul de încărcare

Căutarea

- Într-o operație de căutare reușită, algoritmul determină lista potrivită și apoi caută elementul dorit în aceasta
- În medie, trebuie examinate jumătate din elemente, înainte de a-l găsi pe cel corect
- Timpul de căutare mediu este:
- *1 + L / 2*

Observații

- Această relație este valabilă, indiferent dacă listele sunt sau nu sortate
- Într-o căutare nereușită, dacă listele sunt neordonate, trebuie parcurse toate elementele, deci timpul este:
- $1 + L$


Average Probe Length
0
1
2
3
4
5
6
Unsuccessful
Successful
0
1.0
2.0
3.0
4.0
5.0
Load Factor

Observații

- În cazul listelor ordonate, o căutare nereușită va examina, în medie, jumătate dintre elemente, timpul fiind același ca și pentru o operație reușită
- În metoda înlănțuirii separate se utilizează un coeficient de încărcare de aproximativ *1* (numărul de elemente egalează dimensiunea tabelei)

Observații

- Valorile mai mici ale factorului de încărcare nu determină îmbunătățiri semnificative ale performanțelor
- Timpul necesar tuturor operațiilor crește liniar, în raport cu coeficientul de încărcare, deci depășirea valorii *2* nu este recomandată

Inserarea

- Dacă listele nu sunt ordonate, inserarea este întotdeauna imediată, adică nu necesită efectuarea de comparații
- Funcția de dispersie trebuie calculată, deci timpul de inserare este de ordinul *O(1)*

Inserarea

- Pentru liste sortate, ca și în cazul căutărilor nereușite, trebuie examinate, în medie, jumătate din elementele din listă
- Timpul de inserare este:
- *1 + L / 2*

Adresare deschisă sau înlănțuire separată

- Dacă se optează pentru adresarea deschisă, dubla dispersie are performanțe ceva mai bune față de sondajul pătratic
- Excepția o constituie cazul în care dispunem de un volum mare de memorie și datele nu se mai expandează după crearea tabelei

Adresare deschisă sau înlănțuire separată

- Sondajul liniar este ceva mai ușor de implementat și, dacă coeficientul de încărcare nu depășește *1/2*, funcționează cu performanțe destul de bune
- Dacă numărul de elemente inserate nu este cunoscut dinainte, este de preferat să utilizăm înlănțuirea separată

Adresare deschisă sau înlănțuire separată

- Creșterea coeficientului de încărcare determină scăderi ale performanțelor în metoda adresării deschise
- Pentru înlănțuirea separată, scăderea performanțelor este un proces liniar

Concluzii

- Tabelele de dispersie au ca structuri de bază tablourile
- Domeniul de valori ale cheilor, este, de regulă, mai mare decât dimensiunea tabelei
- Valoarea unei chei este asociată unui indice dintr-un tablou, utilizând o funcție de dispersie

Concluzii

- Un dicționar poate fi implementat eficient printr-o tabelă de dispersie
- Dispersia unei chei într-o celulă deja ocupată se numește coliziune
- Coliziunile pot fi tratate în două moduri:
  - Prin adresare deschisă
  - Prin înlănțuire separată

Concluzii

- În adresarea deschisă, elementele asociate unei celule deja ocupate sunt plasate în alte celule din tabelă
- În metoda înlănțuirii separate, fiecare celulă din tabelă este o listă înlănțuită
- Toate elementele asociate acelei celule sunt inserate în acea listă

Concluzii

- Există trei metode distincte de adresare deschisă:
  - Sondajul liniar
  - Sondajul pătratic
  - Dubla dispersie

Concluzii

- În metoda sondajului liniar, pasul de deplasare este *1*, deci, dacă *x* este indicele calculat de funcția de dispersie, secvența de sondaj este:
- *x, x+1, x+2, x+3, ...*
- Numărul pașilor necesari pentru a găsi un anumit element se numește lungimea sondajului

Concluzii

- Sondajul liniar determină apariția secvențelor continue de celule ocupate
- Acestea se numesc fascicule primare și determină o scădere a performanțelor
- În sondajul pătratic, deplasarea față de celula inițială *x* este pătratul numărului iterației, deci secvența de sondaj este:
- *x, x+1, x+4, x+9, x+16, ...*

Concluzii

- Sondajul pătratic elimină fasciculele primare, dar determină apariția problemei, mai puțin importante, a fasciculelor secundare
- Fasciculele secundare apar datorită faptului că toate cheile care au asociată aceeași valoare vor utiliza aceeași secvență de sondaj

Concluzii

- Toate cheile asociate aceleiași valori utilizează aceeași secvență de sondaj, din cauză că pasul de deplasare nu depinde de cheie, ci numai de numărul iterației
- În metoda dublei dispersii, pasul de deplasare depinde de cheie, fiind obținut prin utilizarea unei funcții secundare de dispersie

Concluzii

- Dacă funcția secundară de dispersie întoarce valoarea *s*, secvența de sondaj este:
- *x, x+s, x+2s, x+3s, x+4s, ...*
- unde valoarea *s* depinde de cheie, dar rămâne constantă pe parcursul sondajului

Concluzii

- Coeficientul de încărcare este raportul dintre numărul de elemente dintr-o tabelă de dispersie și dimensiunea maximă a tabelei
- Valoarea maximă a coeficientului de încărcare, în cazul adresării deschise, trebuie să fie în jur de *0,5*

Concluzii

- În metoda dublei dispersii, la un coeficient de încărcare de *0,5*, lungimea medie a unei secvențe de sondaj este *2*
- În cazul adresării deschise, timpii de căutare tind spre infinit, pe măsură ce coeficientul de încărcare se apropie de valoarea *1*

Concluzii

- Este important ca tabelele cu adresare deschisă să nu fie foarte pline
- Înlănțuirea separată permite utilizarea unui coeficient de încărcare de *1*
- La această valoare, secvența de sondaj pentru o căutare reușită are lungimea medie de *1,5*, iar pentru o operație nereușită, de *2*

Concluzii

- Lungimea sondajului crește liniar, în raport cu coeficientul de încărcare, în cazul înlănțuirii separate
- Dimensiunea tabelei de dispersie trebuie să fie, în general, un număr prim
- Această condiție trebuie respectată în special în cazul sondajului pătratic și al dublei dispersii