Data Structures and Algorithms (10)

Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Chapter 10. Search

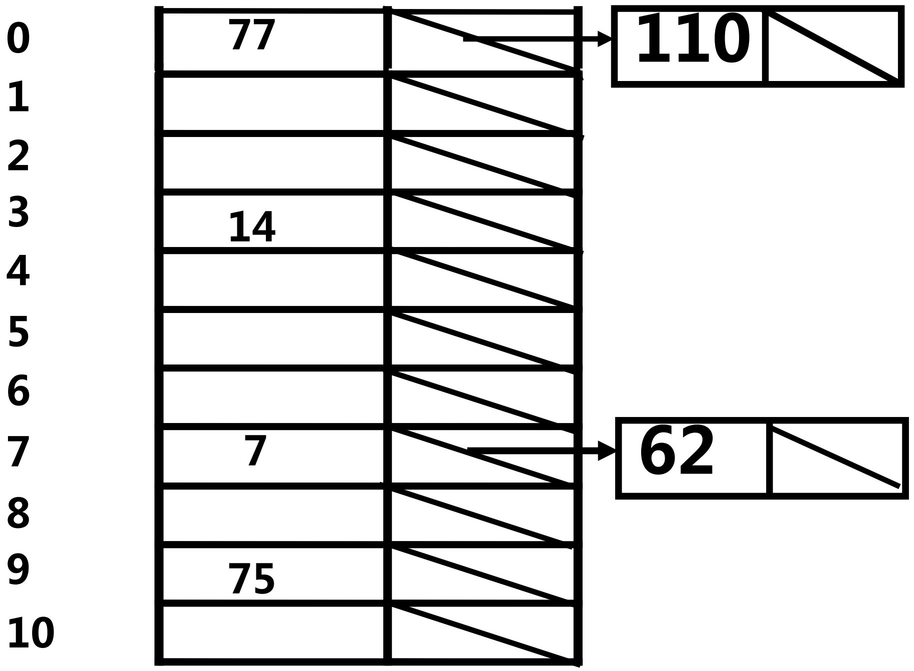
Search in a Hash Table

- 10.3.0 Basic problems in hash tables
- 10.3.1 Collisions resolution
- 10.3.2 Open hashing
- 10.3.3 Closed hashing
- 10.3.4 Implementation of closed hashing
- 10.3.5 Efficiency analysis of hash methods

Open Hashing

{77、14、75、 7、110、62 、95 }

■h(Key) = Key % 11



- The empty cells in the table should be marked by special values
  - like -1 or INFINITY
  - Or make the contents of hash table to be pointers, and the contents of empty cells are null pointers

Performance Analysis of Chaining Method

- Give you a table of size M which contains n records. The hash function (in the best case) put records evenly into the M positions of the table which makes each chain contains n/M records on the average
  - When M>n, the average cost of hash method is $\Theta(1)$

10.3.3 Closed Hashing

- d_0=h(K) is called the base address of K.
- When a collision occurs, use some method to generate a sequence of hash addresses for key K

  d_1, d_2, … d_i , …, d_{m-1}
  - All the d_i (0<i<m) are the successive hash addresses
- With different way of probing, we get different ways to resolve collisions.
- Insertion and search function both assume that the probing sequence for each key has at least one empty cell
  - Otherwise it may get into a endless loop
- We can also limit the length of probing sequence

Problem may Arise - Clustering

- Clustering
  - Nodes with different hash addresses compete for the same successive hash address
  - Small clustering may merge into large clustering
  - Which leads to a very long probing sequence

Several General Closed Hashing Methods

- 1. Linear probing
- 2. Quadratic probing
- 3. Pseudo-random probing
- 4. Double hashing

1. Linear probing

- Basic idea:
  - If the base address of a record is occupied, check the next address until an empty cell is found
    - Probe the following cells in turn: d+1, d+2, ……, M-1, 0, 1, ……, d-1
  - A simple function used for the linear probing:

    p(K,i) = I
- Advantages:
  - All the cell of the table can be candidate cells for the new record inserted

Instance of Hash Table

- M = 15, h(key) = key%13
- In the ideal case, all the empty cells in the table should have a chance to accept the record to be inserted
  - The probability of the next record to be inserted at the 11th cell is 2/15
  - The probability to be inserted at the 7th cell is 11/15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
26	25	41	15	68	44	6				36		38	12	51

Enhanced Linear Probing

- Every time skip constant c cells rather than 1
  - The ith cell of probing sequence is $(\mathrm{h}(K) + ic) \bmod M$
  - Records with adjacent base address would not get the same probing sequence
- Probing function is $\mathrm{p}(K,i) = i*c$
  - *Constant c and M must be co-prime*

Example: Enhance Linear Probing

- For instance, c = 2, The keys to be inserted, k_1 and k_2. $h(k_1) = 3$, $h(k_2) = 5$
- Probing sequences
  - The probing sequence of k_1: 3, 5, 7, 9, ...
  - The probing sequence of k_2: 5, 7, 9, ...
- The probing sequences of k_1 and k_2 are still intertwine with each other, which leads to clustering.

2. Quadratic probing

- Probing increment sequence: 1^2, -1^2, 2^2, -2^2, ..., The address formula is

  $d_{2i-1} = (d + i^2) \% M$

  $d_{2i} = (d - i^2) \% M$

- A function for simple linear probing :

  $p(K, 2i-1) = i*i$

  $p(K, 2i) = - i*i$

Example: Quadratic Probing

- Example: use a table of size M = 13
  - Assume for k_1 and k_2, $h(k_1)=3$, $h(k_2)=2$
- Probing sequences
  - The probing sequence of k_1: 3, 4, 2, 7, ...
  - The probing sequence of k_2: 2, 3, 1, 6, ...
- Although k_2 would take the base address of k_1 as the second address to probe, but their probing sequence will separate from each other just after then

3. Pseudo-Random Probing

- Probing function p(K,i) = perm[i - 1]
  - here perm is an array of length M – 1
  - It contains a random permutation of numbers between 1 and M

```
// generate a pseudo-random permutation of n numbers
void permute(int *array, int n) {
        for (int i = 1;  i <= n;  i ++)
                swap(array[i-1], array[Random(i)]);
}
```

Example: Pseudo-Random Probing

- Example: consider a table of size M = 13, perm[0] = 2, perm[1] = 3, perm[2] = 7.
  - Assume 2 keys k_1 and k_2, $h(k_1)=4$, $h(k_2)=2$
- Probing sequences
  - The probing sequence of k_1: 4, 6, 7, 11, ...
  - The probing sequence of k_2: 2, 4, 5, 9, ...
- Although k_2 would take the base address of k_1 as the second address to probe, but their probing sequence will separate from each other just after then

Secondary Clustering

- Eliminate the primary clustering
  - Probing sequences of keys with different base address overlap
  - Pseudo-random probing and quadratic probing can eliminate it
- Secondary clustering
  - The clustering is caused by two keys which are hashed to one base address, and have the same probing sequence
  - Because the probing sequence is merely a function that depends on the base address but not the original key.
  - Example: pseudo-random probing and quadratic probing

4. Double Probing

- Avoid secondary clustering
  - The probing sequence is a function that depends on the original key
  - Not only depends on the base address
- Double probing
  - Use the second hash function as a constant
    - $p(K, i) = i * h_2(\text{key})$
  - Probing sequence function
    - $d = h_1(\text{key})$
    - $d_i = (d + i\, h_2(\text{key})) \% M$

Basic ideas of Double Probing

- The double probing uses two hash functions h_1 and h_2
- If collision occurs at address h_1(key) = d, then compute h_2(key), the probing sequence we get is :

  (d+h_2(key)) % M , (d+2h_2 (key)) %M , (d+3h_2 (key)) % M , …
- It would be better if h_2 (key) and M are co-prime
  - Makes synonyms that cause collision distributed evenly in the table
  - Or it may cause circulation computation of addresses of synonyms
- Advantages: hard to produce “clustering”
- Disadvantages: more computation

Method of choosing M and h2(k)

- Method1: choose a prime M, the return values of h_2 is in the range of
  $1 \le h2(K) \le M - 1$
- Method2: set $M = 2^m$,let h_2 returns an odd number between 1 and 2^m
- Method3: If M is a prime, $h_1(K) = K \bmod M$
  - $h_2(K) = K \bmod (M-2) + 1$
  - or $h_2(K) = [K / M] \bmod (M-2) + 1$
- Method4: If M is a arbitrary integer, $h_1(K) = K \bmod p$ (p is the maximum prime smaller than M)
  - $h_2(K) = K \bmod q + 1$ (q is the maximum prime smaller than p)

Thinking

- When inserting synonyms, how to organize synonyms chain?
- What kind of relationship do the function of double hashing h_2 (key) and h_1 (key) have?

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)