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. Retrieval

- 10.1 Retrieval in a linear list
- 10.2 Retrieval in a set
- 10.3 Retrieval in a hash table
- Summary

Retrieval in a Hash Table

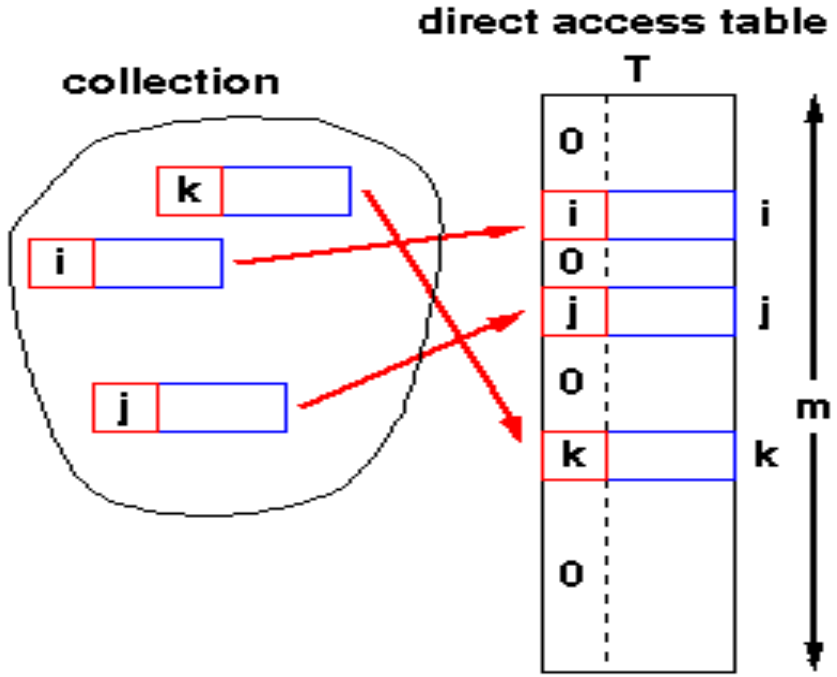
- 10.3.0 Basic problems in hash tables
- 10.3.1 Collision resolution
- 10.3.2 Open hashing
- 10.3.3 Closed hashing
- 10.3.4 Implementation of closed hashing
- 10.3.5 Efficiency analysis of hash methods

Basic problems in Hash Tables

- Retrieval based on comparison of keys
  - Sequential search: ==, !=
  - Binary search, tree based: >, == , <
- Retrieval is the operation interfaced with users
- When the problem size is large, the time efficiency of retrieval methods mentioned above may become intolerable for users
- In the best case
  - Find the storage address of the record according to the key
  - No need to compare the key with candidate records one by one.

Think of Hash from Direct Access

- For example, we can get the element in an array with a specific subscript
  - Inspired by this, computer scientists invented hash method.
- A certain function relation h()
  - Keys of nodes k are used as independent variables
  - Function value h(K) is used as the storage address of the node
- Retrieval uses this function to calculate the storage address
  - Generally, a hash table is stored in a one-dimensional array
  - The hash address is the array index



Example 1

Example 10.1: you already know the set of the key of a linear list: S = {and, array, begin, do, else, end, for, go, if, repeat, then, until, while, with}

We can let the hash table be: char HT2[26][8];

The value of hash function H(key), is the sequence number of the first letter of key in the alphabet {a, b, c, …, z}, which means H(key) = key[0] – 'a'

Example 1 (continued)

Hash address	key
0	(and, array)
1	begin
2	
3	do
4	(end, else)
5	for
6	go
7	
8	if
9	
10	
11	

Hash address	key
13	
14	
15	
16	
17	repeat
18	
19	then
20	until
21	
22	(while, with)
23	
24	

Example 2

```
// the value of hash function is the average of the sequence numbers of
the first and the last letters of key in the alphabet. Which means:
int H3(char key[])
{
    int i = 0;
    while ((i<8) && (key[i]!='\0'))  i++;
    return((key[0] + key(i-1) – 2*'a') /2 )
}
```

Example 2 (continued)

Hash address	key
0	
1	and
2	
3	end
4	else
5	
6	if
7	begin
8	do
9	
10	go
11	for

Hash address	key
13	while
14	with
15	until
16	then
17	
18	repeat
19	
20	
21	
22	
23	
24	

Several Important Concepts

- The load factor $\alpha = N/\mathrm{M}$
  - M is the size of the hash table
  - N is the number of the elements in the table
- Collision
  - Some hash function return the same value for 2 or more distinct keys
  - In practical application, there are hardly any hash functions without collision
- Synonym
  - The two keys that collides with each other

Hash Function

- Hash function: the function mapping keys to storage addresses, generally denoted by h
- $Address = Hash(key)$
- Principles to select hash functions
  - Be easy to compute
  - The range of the function must be inside the range of the hash table
  - Try to map two distinct keys to different addresses as good as possible.

Various Factors Needed to be Consider

- Lengths of keys
- Size of hash tables
- Distribution of keys
- Frequency rate of searching for records
- …

Commonly-Used Hash Functions

- 1. Division method
- 2. Multiplication method
- 3. Middle square method
- 4. Digit analysis method
- 5. Radix conversion method
- 6. Folding method
- 7. ELF hash function

1. Division method

- Division method: divide M by key x, and take the remainder as the hash address, the hash function is:

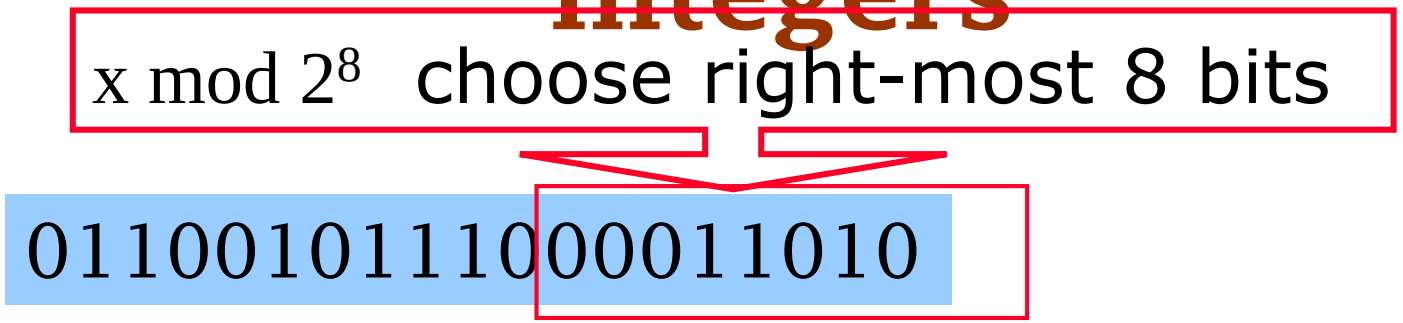
$$h(x) = x \bmod M$$

- Usually choose a prime as M
  - The value of function relies on all the bits of independent variable x, not only right-most k bits.
  - Increase the probability of evenly distribution
  - For example, 4093

Why isn't M an even integer?

- If set M as an even integer?
  - If x is an even integer, h(x) is even too.
  - If x is an odd integer, h(x) is odd too;
- Disadvantages: unevenly distribution
  - If even integers occur more often than odd integers, the function values would not be evenly distributed
  - Vice versa

M shouldn't be a Power of Integers



- If set M as a power of 2
  - Then, $h(x) = x \bmod 2^k$ is merely right-most k bits of x (represented in binary form)
- If set M to a power of 10
  - Then, $h(x) = x \bmod 10^k$ is merely right-most k bits of x (represented in decimal)
- Disadvantages: hashed values don't rely on the total bits of x

Problems of Division Method

- The potential disadvantages of division method
  - Map contiguous keys to contiguous values
- Although ensure no collision between contiguous keys
- Also means they must occupy contiguous cells
- May decrease the performance of hash table

2. Multiplication method

- Firstly multiply *key* by a constant A (0 < A < 1), extract the fraction part
- Then multiply it by an integer n, then round it down, and take it as the hash address
- The hash function is:
  - $hash(key) = \lfloor n * (A * key \% 1) \rfloor$
  - *"A * key % 1" denotes extracting the fraction part of A * key*
  - $A * key \% 1 = A * key - \lfloor A * key \rfloor$

Example

- let key = 123456, n = 10000 and let A =
  = 0.6180339 ,
- Therefore,

$$hash(123456) = (\sqrt{5}-1)/2$$
$$= \lfloor 10000*(0.6180339*123456 \ \% \ 1) \rfloor =$$
$$= \lfloor 10000 * (76300.0041151... \ \% \ 1) \rfloor =$$
$$= \lfloor 10000 * 0.0041151... \rfloor = 41$$

Consideration about the Parameter Chosen in Multiplication Method

- If the size of the address space is p-digit then choose $n = 2^p$
  - The hash address is exactly the left-most p bits of the computed value
  - A * key % 1 = A * key - ⌊ A * key ⌋
  - Advantages: not related to choose of n
- Knuth thinks: A can be any value, it's related to the features of data waited to be sort. Usually golden section is the best

3. Middle Square Method

- Can use middle square method this moment: firstly amplify the distinction by squaring keys, then choose several bits or their combination as hash addresses.
- For example
  - A group of binary key: (00000100 , 00000110 , 000001010 , 000001001 , 000000111)
  - Result of squaring: (00010000 , 00100100 , 01100010 , 01010001 , 00110001)
  - If the size of the table is 4-digit binary number, we can choose the middle 4 bits as hash addresses: (0100, 1001, 1000, 0100, 1100)

4. Digit Analysis Method

- If there are *n* numbers, each with *d* digits and each digit can be one of *r* different symbols
- The occurring probabilities of these *r* symbols may are different
  - Distribution on some digits may be the same for the probabilities of all the symbols
  - Uneven on some digits, only some symbols occur frequently.
- Based on the size of the hash table, pick evenly distributed digits to form a hash address

Digit Analysis Method (2/4)

- The evenness of distribution of each digit λ_k

$$\lambda_k = \sum_{i=1}^{r} (\alpha_i^k - n/r)^2$$

  - α_i^k denotes the occurring number of ith symbols
  - n/r denotes expected value of all the symbols occurring on n digits evenly
- The smaller λ_k get, the more even the distribution of symbols on this digit is

Digit Analysis Method (3/4)

- If the range of hash table address is 3 digits, then pick the ④ ⑤ ⑥ digits of each key to form the hash address of the record
- We can add ①, ②, ③ digits to ⑤ digit, get rid of the carry digit, to become a 1-digit number. Then combine it with ④, ⑥ digits, to form a hash address. Some other methods also work

9	9	2	1	4	8
9	9	1	2	6	9
9	9	0	5	2	7
9	9	1	6	3	0
9	9	1	8	0	5
9	9	1	5	5	8
9	9	2	0	4	7
9	9	0	0	0	1
①	②	③	④	⑤	⑥

①digit, $\lambda_1 = 57.60$
②digit, $\lambda_2 = 57.60$
③digit, $\lambda_3 = 17.60$
④digit, $\lambda_4 = 5.60$
⑤digit, $\lambda_5 = 5.60$
⑥digit, $\lambda_6 = 5.60$

Digit Analysis Method (4/4)

- Digit analysis method is only applied to the situation that you know the distribution of digits on each key previously
  - It totally relies on the set of keys
- If the set of keys changes, we need to choose again

5. Radix Conversion Method

- Regard keys as numbers using another radix.
- Then convert it to the number using the original radix
- Pick some digits of it as a hash address
- Usually choose a bigger radix as converted radix, and ensure that they are inter-prime.

Example: Radix Conversion Method

- For instance, give you a key $(210485)_{10}$ in base-10 system, treat it as a number in base-13 system, then convert it back into base-10 system
- $(210485)_{13}$

  $= 2\times13^5 + 1\times13^4 + 4\times13^2 + 8\times13 + 5$

  $= (771932)_{10}$
- If the length of hash table is 10000, we can pick the lowest 4 digits 1932 as a hash address

6. Folding Method

- The computation becomes slow if we use the middle square method on a long number
- Folding method
  - Divide the key into several parts with same length (except the last part)
  - Then sum up these parts (drop the carries) to get the hash address
- Two method of folding:
  - Shift folding — add up the last digit of all the parts with alignment
  - Boundary folding — each part doesn't break off, fold to and fro along the boundary of parts, then add up these with alignment, the result is a hash address

Example: Folding Method

- [example 10.6] If the number of a book is 04-42-20586-4

5 8 6 4	0 4 4 2 2 0 5 8 6 4
4 2 2 0	0 2 2 4 4 0
+ 0 4	+ 0 4
[1] 0 0 8 8	6 0 9 2
h(key)=0088	h(key)=6092

- (a) shift holding (b) Boundary holding

7. ELF hash function

- Used in the UNIX System V4.0 “Executable and Linking Format(ELF for short)
-

```
int ELFhash(char* key) {
unsigned long h = 0;
while(*key) {
    h = (h << 4) + *key++;
    unsigned long g = h & 0xF0000000L;
    if (g) h ^= g >> 24;
    h &= ~g;
}
return h % M;
}
```

Features of ELF hash function

- Work well for both long strings and short strings
- Chars of a string have the same effect
- The distribution of positions in the hash table is even.

Application of Hash Functions

- Choose appropriate hash functions according to features of keys in practical applications
- Someone have used statistical analysis method of "roulette" to analyze them by simulation, and it turns out that the middle square is closest to "random"
  - If the key is not a integer but a string, we can convert it to a integer, then apply the middle square method

Thinking

- Consider when using hash methods:

  (1) how to construct (choose) hash functions to make nodes distributed evenly

  (2) Once collision occurs, how to solve it?

- The organization methods of the hash table itself

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)