Map and HashMap

Lan Huang

09/23/2018

Differences of Map, HashMap and HashTable

Map vs HashMap:

HashMap is a class derived from Map interface

Source:
https://way2java.com/collections/map/map-vs-hashmap/

HashMap vs HashTable:

HashMap is non synchronized.
HashMap allows null key or values whereas Hashtable doesn’t allow any null key or value.
HashMap is generally **preferred** over HashTable if thread synchronization is not needed

Source:
https://way2java.com/collections/hashtable/hashtable-vs-hashmap/
https://www.geeksforgeeks.org/differences-between-hashmap-and-hashtable-in-java/

Hashing

- Dictionary: key=>value

    Insert()

    Delete()

    Search()

    AVL_Tree O(lgn)   Better time?

- Yes Hashing O(1)

Direct Access Table

- Problems:

  1)Key may not be int

  2)Gigantic memory need

- Solution to 1) prehash

  1)map key to none neg int

  python: hash(x)

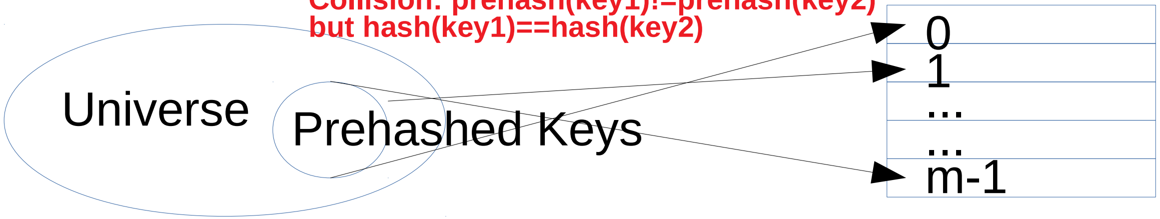
  Ideally, hash(x)==hash(y) => x==y

  But, in reality…

- Solution to 2) hashing

  2) reduce the universe of prehash values down to reasonable size m for table. Ideally; m=O(n)

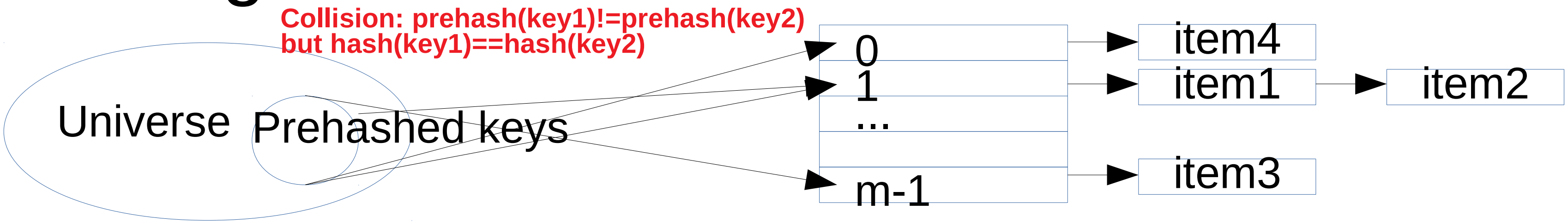
Collision: prehash(key1)!=prehash(key2)
but hash(key1)==hash(key2)



0	
1	
2	
3	item3
4	
5	
6	
7	item7
8	
9	
……	
……	
……	

How to solve collision?

- Chaining: linked list



Worst case O(n) all prehashed keys map to same slot

Assume: Simple uniform hashing

Each prehashed key map to each slot with equal probability

Analysis:

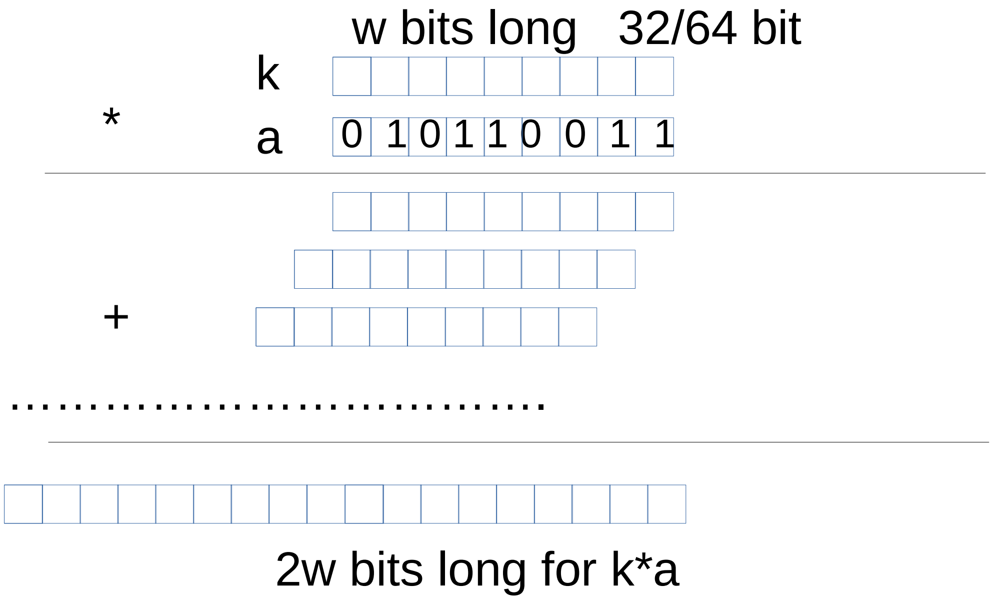
Expected length of chain: n/m =alpha (load factor) =O(1) if m=O(n)

Running time: O(1+alpha) O(1) find slot, O(alpha) find item

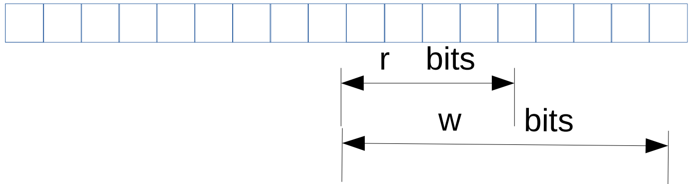
Hash functions:

1) h(k)= k mod m

2) [a*k mod 2^w] >> (w-r)



For [a*k mod 2^w] We know (x mod y) =[0,1,2,….y-2,y-1]
Therefore, [a*k mod 2^w] = [0 ,1,...2^w-1] i.e. right w bits



m=2^r

3)Universal hashing

h(k)=[(a*k+b) mod p] mod m

a/b random number [0,1,...p-1]   p=prime number

for worst case prehash(k1)!=prehash(k2) but Pr{hash(k1)==hash(k2)}=1/m

- How to choose m?

  Start at small e.g. m=8

  Grow/Shrink as needed

  If n>m **Grow**: m→m' O(m+n+m')

  make table of size m'

  build new hash h'

  rehash: for item in T:

  T'.insert(item)

- same as list in week1 m'=2m insert n items, cost O(1+2+4+8+..n)=O(n)
- TABLE DOUBLING Amortized cost O(1)

- **If n=m/2** **Shrink**: m → m/2

BAD: current #items 8 add 1item, doubling m=16

current #items 9, delete 1 item, shrinking to m=8.

Insert 1

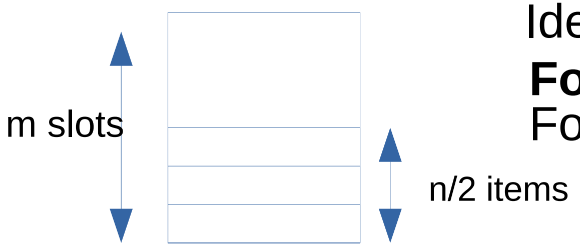
m changes: 2^K <=> 2^(K + 1)

delete 1

Lead to O(n) per insert/delete

If n=m/4 **Shrink**: m → m/2

Amortized time O(1)



Ideal case m=O(n) , For example m=n

For insertion or deletion, we always keep at least a double space.

For deletion, if m=n/2 , shirink m → m/2 then m=n , no free space.

Therefore, for deletion, only when n=m/4 we shrink, still keep at least double space.

- Open hashing: in which a single array element can store any number of elements

- Close hashing: also known as **open addressing**. Instead of storing a set at every array index, a single element is stored there. If an element is inserted in the hash table and collides with an element already stored at that index, a second possible possible location for it is computed. If that is full, the process repeats.

  There are various strategies for generating a sequence of hash values for a given element.

-

0	
1	
2	
3	item3
4	
5	
6	
7	item7
8	
9	
…...	
…...	
…...	

linear probing, quadratic probing, double hashing.

h(k,i)=h’(k)+i mod m **problem: clustering**

h(key, count)
e.g. h(k,1) h(k,2) … h(k,m-1)

insert(k,count):
Keep probing until an empty slot is found or meet deleted slot
search(k):
When key!=k, keep probing until find k or find a empty slot(continue when see a deleted flag)
delete(k):
Replace deleted item with deleted flag.

Clustering? Therefore, double hashing.

$$h(k,i)=(h1(k)+i^* h2(k)) \bmod m$$

Hashing Application:

Password in database

Save hash value. Given hash, we can not get the original value.

e.g. https://en.wikipedia.org/wiki/MD5

Normally hashing with salt.

Bloom filter (data structure):

https://en.wikipedia.org/wiki/Bloom_filter

e.g. not allowed password.

Bloom filter

- N identical balls and n bins
- Each ball thrown into a random bin
- Def load(i)=#balls assigned to bin_i
- What's the max(load(:))?

  worst case, all ball assign to bin_i. max(load)=N

  Pr{ (1/n)^n }

  What's the max(load(:)) on average?

Bloom filter

- what's the Pr that first log(n) balls assign to bin_i?
  Pr(ball 1,ball2,...ball log(n) to bin_i)=(1/n)^(log(n))
- Pr(load(bin_i) >= log(n))?
  we will prove this Pr is small
- Pr(load(bin_i) >= log(n))<= $C_n^{\log(n)}$ (1/n)^(log(n)) * (1-1/n)^(n-log(n))

The upper bound:

$$\binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

http://www.cs.technion.ac.il/~bshouty/COLT/LECTUR-NOTES/PROBABILITY/probability.pdf

So, Pr(load(bin_i)>=log(n))<= C_n^{logn}(1/n)^(log(n))= $(\frac{ne}{logn})^{logn} * (\frac{1}{n})^{logn}$

= $(e/logn)^{logn}$<= $(1/4)^{logn}$ (for n> 2^{11}) = $1/n^2$

Bloom filter

- So, Pr(load(bin_i)>=log(n))<= $1/n^2$
- Therefore, Pr(max(load(:))>=log(n))

  = Pr(at least one bin has load >=log(n))

  <= Sum_i Pr(load(bin_i)>=log(n))

  = n* $1/n^2$=1/n
- Pr(max load < log(n)) >= 1-1/n

Bloom filter

- For equal probiblity assumption:
- Maxload < log(n)   with high prob

  Can we do better?
- Yes, Assign balls sequentially.   Method: Best of 2

  For i=1$\rightarrow$n:

  choose 2 random bins, bin_j and bin_k

  if load(bin_j)<load(bin_k):

  assign ith ball to bin_j

  else: assign ith ball to bin_k
- Maxload < loglog(n) with high prob

Bloom filter

- Unacceptable passwords:

  Huge Set U= possible passwords

  S belong to U of unacceptable passwords
- Query: for x in U is x in S?

  HashTable H=[0,1,...n-1] array of linked lists

  HashFunction h: U $\rightarrow$ H

  so h(x) random in [0,1..,n-1]

  $|U|=N >> |H|=n >= |S|=m$

Bloom filter

- **Chain Hashing**

  H is array of linked lists

  H[i]=linked list of element x in S where h(x)=i
- Query time? Is x in S? Load at bin_h(x)

  |H|=n |S|=m   if m=n then maxload=O(logn)

  So, upper bond Query time O(logn)   better time?
- Get maxload=O(1) need n=O(m^2) Space cost to large.
- Solution: 2 hash functions h1 and h2

  But how to add x into S?

  Compute h1(x) and h2(x)

  add x to least loaded of h1(x) / h2(x) bin.

  How to check y in S?

  Compute h1(y) and h2(y)

  check for y in H[h1(y)] U H[h2(y)]

  what's the query time ? If m=n then query time is O(loglogn)

Bloom filter

- Unacceptable password

  chain hashing, O(logn) for 1 hashfunction

  or O(loglogn) for 2 hashfuncitons

- Better way? Goal : queries =O(1) less space.

  But **false positive** with small Pr

- False positive: x not in S, but say x in S (S is set of unacceptable password)

  Insert(x): add x in U into S

  query(x): is x in S? If x in S, we always output yes.

  If x not in S: we most time output no.

  But sometimes output yes.

Bloom filter

- H is a 0-1 array of size n

  Init: H to all 0's using random $h:U \to H$

  insert(x):set H[h(x)]=1

  Query(x): if H[h(x)]==1 then output Yes

  else No

- False positive: x not in S, y in S where h(x)=h(y)

  need to minimize false positive

Bloom filter

- K hash functions: h1, h2,...hk

  Init H to all 0's

  Insert(x): for i=1→k:

  set H[hi(x)]=1

  Query(x): if for all I, H[hi(x)]==1 then output Yes

  for some I, H[hi(x)]==0, then output No
- Still,we have false positive, when we say Yes, could be x not in S
- Reason: hi(x)=hj(z) when insert z, we using jth hashfunciton set not only for z but for x's ith hash funciton. So, to get a false positive, need all 1...k th bit for x accsidently set by other insertion.

  If k is large, too many hash functions set bit to 1. False positive rate increase.

  If k is small, checking to few bits. False positive could still be high.

Bloom filter

- False positive analysis:

  $|S|=m$ $|H|=n$ $n>=m$ let $c=n/m$ for $c>1$ (ave bit per entry)
- $|H|=cm$
- Pr(false positive for x not in S)=Pr(h1(x)=h2(x)...=hk(x)=1)
- For b in [0,1,…,n-1], Pr(H[b]==1)=1-Pr(H[b]=0)

  we have m insertion, each insertion has K hashval

  like throw mk balls into n bins: what’s Pr(H[b]==0)?

  Pr(H[b]==0)=Pr(all mk balls miss bin_b)= $(1-1/n)^{mk}$
- recall $e^{-a}=1-a$ if $a \to 0$, so if $n \to \inf$,So
- $(1-1/n)^{mk} = e^{(-1/n)mk} = e^{-k/c} =$ Pr(H[b]==0)
- So Pr(H[b]==1)=1-Pr(H[b]=0)= $1-e^{-k/c}$
- Therefore, Pr(false positive for x not in S)=Pr(h1(x)=h2(x)...=hk(x)=1)= $(1-e^{-k/c})^k$

Bloom filter

- False positive prob $=f(k)= (1-e^{-k/c})^k$
- Miminize f(k) set $f'(k)=0$ So, $k=c\ln 2$
- $f(c\ln 2)= (0.5^{\ln 2})^c=0.6185^c$

- $m=|S|$ $|H|=cm$ for $c>1$; false positive prob = 0.6185^c

e.g.	$c=10$	0.0082
	$c=100$	$1.3*10^{-21}$

- # Problem1: String matching

method1:

ABCDEFGHIJKLMNOPQRST

OPQ

   OPQ

      OPQ

         ....

                   ...

                                        OPQ

O(m*n) slow

Method2: Rolling hash   O(m+n)

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec09.pdf

- # Problem 2:Group Anagrams

Given an array of strings, group anagrams together.

Example:

Input: ["eat", "tea", "tan", "ate", "nat", "bat"],
Output:
[
  ["ate","eat","tea"],
  ["nat","tan"],
  ["bat"]
]

```python
class Solution:
    def groupAnagrams(self, strs):
        """
        :type strs: List[str]
        :rtype: List[List[str]]
        """
        mydict=dict()
        for e in strs:
            if "".join(sorted(e)) in mydict.keys():
                mydict["".join(sorted(e))].append(e)
            else:
                mydict["".join(sorted(e))]=[e]
        return [v for k,v in mydict.items()]
```

https://leetcode.com/problems/group-anagrams/description/

- # Problem3 Subarray Sum Equals K

Given an array of integers and an integer k, you need to find the total number of continuous subarrays whose sum equals to k.

Example 1:

Input:nums = [1,1,1], k = 2
Output: 2

```python
class Solution:
    def subarraySum(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: int
        """
        """
        sum  0    1              2               3
             0 sum[0]+nums[0]  sum[1]+nums[1]  sum[2]+nums[2]
        """
        count = 0;
        sum=[0 for i in range(len(nums)+1)]
        for i in range(1,len(nums)+1):
            sum[i] = sum[i - 1] + nums[i - 1];
        for start in range(0,len(nums)+1):
            for end in range(start+1,len(nums)+1):
                if sum[end] - sum[start] == k:
                    count+=1

        return count;
```

```python
class Solution:
    def subarraySum(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: int
        """
        dic = {0 : 1}
        total = 0
        cnt = 0
        for n in nums:
            total += n
            if total - k in dic:
                cnt += dic[total - k]
            if total not in dic:
                dic[total] = 1
            else:
                dic[total] += 1
        return cnt
```

https://leetcode.com/problems/subarray-sum-equals-k/solution/
https://leetcode.com/problems/subarray-sum-equals-k/description

- # Problem 4: Contiguous Array

Given a binary array, find the maximum length of a contiguous subarray with equal number of 0 and 1.

Example 1:

Input: [0,1]
Output: 2
Explanation: [0, 1] is the longest contiguous subarray with equal number of 0 and 1.

```python
class Solution(object):
    def findMaxLength(self, nums):
        count = 0
        max_length=0
        table = {0: 0}
        for index, num in enumerate(nums, 1):
            if num == 0:
                count -= 1
            else:
                count += 1

            if count in table:
                max_length = max(max_length, index - table[count])
            else:
                table[count] = index

        return max_length
```

https://leetcode.com/problems/contiguous-array/discuss/99655/Python-O(n)-Solution-with-Visual-Explanation

HW

- Copy List with Random Pointer

https://leetcode.com/problems/copy-list-with-random-pointer

- 4Sum I & II

https://leetcode.com/problems/4sum

https://leetcode.com/problems/4sum-ii

- Maximum Length of Repeated Subarray

https://leetcode.com/problems/maximum-length-of-repeated-subarray

- Replace Words

https://leetcode.com/problems/replace-words/description/