Hashing

Introduction: Suppose we have huge amount of data (can be sorted or unsorted) , if we are strong data in the array then to perform any dictionary operation(insertion , deletion , searching) it will take O(n) or O(logn) , depends on whether we are using linear search or binary search , but this is inefficient if we want to retrieve data fast , So we use another technique known as hashing , which tries to perform all the operation in constant time . Worst case time proportional to the size of the set for each operation (just like array and linked list implementation). Order of elements is irrelevant in hashing .

Basic concept: In hashing , we store key value in the table known as hash table with the help of some mapping function known as hash function . Selection of hash function is critical issue . Hash function should be good enough so that keys are mapped to the hash table uniformly. Concept of hash function selection we will discuss later in this chapter. Consider a simple example to understand the key mapping with the help of hash function .

Key values: 10, 4,6,78,45,99,47

Hash function: key value%10

Slots for key values in the table are :

10%10=0, 4%10=4 , 6%10=6, 78%10=8, 45%10=5, 99%10=9, 47%10=7

0	10
l	
2	
3	
4	4
5	45
6	6
7	47
8	78
9	99

As we saw, all the key value hashed into the unique slot in the hash table . But this is the ideal case, not practically possible . For example

key values: 1, 2,11,62,91,5,35

Hash function: key value %10 1%101, 11%10=1, 91%10=1, 2%10=2, 62%10=2, 5%10=5, 35%10=5.

There are three key values(1,11,91) are hashed(mapped) in slot number 1, two key values(2,62) are hashed in slot number 2 and two key value hashed to slot number 5. Mapping of more than one key in same slot in the hash table is known as **collision**.

Probability theory also justify the collision , there are some standard result from the probability theory about collision

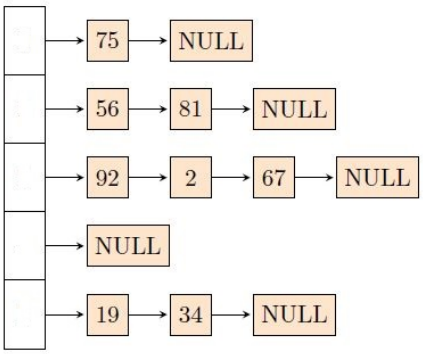
Bins and ball problem : suppose we have N bins and we tossed some ball uniformly then

1. After $\sqrt{N}/2$ tosses we can expect the two balls in the same bin.
2. After $\Theta(NlnN)$ tosses we can expect every bin has more than 1 ball.
3. After N tosses, we expect most loaded bin has Θ(log N/ log log N) balls.

Main reason of collision is bad hash function.
So in hashing we have two issues :
i) Collision resolution
ii) Good hash function calculation

1. Collision resolution : For collision resolution mainly two technique are used
a) Chaining
b) Open addressing
i) Linear probing
ii) Quadratic probing
iii) Double hashing

a) Chaining : In this technique we use the external memory in the form of linked list . Key values are stored in the linked list and hash table contain the pointer to the linked list . For example
Key values: 75,67,2,34,19,81, 56,92.
Hash function: key value%5



Let m be the size of hash table and n be number of keys so load factor (α)is defined by n/m . In case of chaining m$\leq$ n . But we should not make m$\ll$n because in that case length of linked list at every slot will be very high and searching time will approach to O(n)

Analysis of chaining :
i) Space complexity for chaining is θ(n+m) , m is the size of hash table which contain only the pointer to the linked list which store the actual value , To store n values , we require a linked list of size n .
ii) Time complexity for insertion is O(1) because it is not required to store the value at the end of linked list ,we can store at front also because ordering of elements does not

matter in hashing , Even if we are storing at the end of linked list , no need to traverse the whole linked list , we can use extra pointer which will point to the last node of the linked list.

iii) **Unsuccessful searching :**if we have good hash function then key values uniformly distribute in the hash table , probability of any key hashed to a particular slot is 1/m . Calculating slot number for any key will take O(1) time with the help of hash function .Average size of linked list at every slot of hash table will be O(n/m) , which is equal to the load factor (α) of the table .In case of unsuccessful search , we have to traverse the complete linked list at a given slot so total time is O(1+n/m) or O(1+α).

iv) **Successful search:** In successful search , on average we have to traverse the half length of linked list so time complexity is O(1+α/2) which is equivalent to O(1+α).

v) **Deletion:** for deletion also we have to search the linked list so time complexity is same as of searching .

b) Open Addressing:

In chaining none of the key values are stored in the hash table , only pointer to the linked list are stored . But in Open addressing techniques we store the key in the table itself . If collision occur then we will search for any other vacant slot in the table . This vacant slot is determined by the collision resolution technique. In open addressing ,size of table is always greater than number of key values m>n, ideally we take m=2n

i) Linear probing

ii) Quadratic probing

iii)Double hashing

Insertion :

In linear probing technique while insertion if collision occur then increase the probe number by one and check the next adjacent slot , if next slot is empty then place the key in that empty slot otherwise increment the probe number and check the next slot. This process continue until we get the empty slot in the table .

Number of collision for inserting any key is determined by the probe number. No collision means probe number zero. If we make 3 attempt to store the key then probe number is 2 (2 collision , in 3rd attempt we store the key hence no collision in last attempt)

For example : key values 10,1, 32, 11,34, 33,88

Hash function : key value%10

i)10%10 =0 mapped to slot 0 , probeNo:0

ii)1%10=1 mapped to slot 1, probeNo:0

iii)32%10=2 mapped to slot 2, probeNo:0

iv)11%10=1, collision⇒(1+1)%10=2 ,collision ⇒(2+1)%10=3 mapped to slot 3, probeNo:2.

v)34%10=4 mapped to slot no 4

vi)33%10=3, collision⇒ (3+1)%10=4,collision ⇒(4+1)=5 mapped to slot 5, probeNo=2

vii)88%10=8 mapped to slot 8

Table will looks like :

0	10
l	1
2	32
3	11
4	34
5	33
6	
7	
8	88
9	

Table1.1

Searching : Procedure for searching is same as of insertion . For searching key value , we will use the same function that we used for insertion . Suppose we want to search the key value x in any hash table ,so first we calculate the slot address with the help of hash function then compare the key value at slot address with x . if x=key value then return true otherwise false .In case of false, compare the value x with key value on next slot . Repeat this step until either we get the key value or empty slot.

Successful search : Consider the hash Table1.1 , suppose we want to search the key value=33

33%10 =3, compare 33 with 11 , false⇒(3+1)%10 =4, compare33 with 34, false⇒(4+1)%10=5, compare 33 with 33, true

Unsuccessful search : Consider the hash table 1.1, we want to search the key value =44 , 44%10=4, compare 44 with 34, false⇒(4+1)%10=5, compare 44 with 33 , false ⇒(5+1)%10=6, compare 44 with NULL ,false and terminated.

Deletion: To delete the key value , first search in the hash table then delete . After deletion we will insert some Marker in the hash table otherwise it may give wrong search results for next searches .

Lets consider the hash table 1.1

Delete the key value =34

For deletion first search 34 ,we have to calculate the slot address i.e 34%10=4 , compare 34 with 34 , true . Now vacant the memory location 4 . After deletion table will looks like :

0	10
l	1
2	32
3	11
4	
5	33
6	
7	
8	88
9	

Table1.2

Searching After Deletion:
Now suppose I want to search key value 33,
Slot address:33%10=3, false
so(3+1)%10=4, compare 33 with NULL(In hash table vacant slots are considered as NULL) , false and terminated ⇒33 not found in the hash table but in actual 33 is in the hash table . So in linear probing deletion may cause problem . How to resolve the problem ?
We will use the Marker sign(X) , which signify the key value has been deleted and slot is empty . comparison with Maker sign will always returns false (and we will check the next slot)but not false and terminated. Hash table will looks like

0	10
l	1
2	32
3	11
4	X
5	33
6	
7	
8	88
9	

Table1.3

Insertion after Deletion: After Deletion we are inserting the Marker sign , which signify that slot is empty so while insertion we face marker sign we will delete the marker sign and insert the element on the place of marker sign . For example : Consider the hash table 1.3
Key value :64, Slot address will be 64%10=4 ,now check whether slot is empty or not ,

we have marker sign on location 4 , delete the marker and insert the key value 64 .

0	10
l	1
2	32
3	11
4	64
5	33
6	
7	
8	88
9	

Table1.4

Table size :

key is a mapped into the one of m slot using the functions h(k) = k mod m, Requires only a single divisions, hence fasts. But, m should not be :

• a power of 2, since if m = 2^p, then h(k) is just the p lowest order bits of k

• a power of 10, since then the hash function does not depend on all the decimal digits of k

• 2^p - 1. If k is a character strings are interpreted in the radix 2^p, two strings that's identical except for a transposition of two adjacent character will hash to the same values.

Good values for m

• Primes not too close to exact powers of 2.

Powers of 2 are bad.

If m = 2^p for integer p, then h(k) is just the least significant p bits of k.128 = 2^7

CLRS as (67*128^3) + ($76 * 128^2$) + ($82 * 128^1$) + ($83 * 128^0$)$mod 2^7$ =

$1000011100110010100101010011_2 = 83 = 1010011_2$

ABCS as ($65 * 128^3$) + ($66 * 128^2$) + ($67 * 128^1$) + ($83 * 128^0$)$mod 2^7$ =

$1000001100001010000101010011_2 = 83 = 1010011_2$ **Drawback of linear probing :**

Primary clustering is the problem with linear probing . Primary clustering means we have long sequence of occupied slots and for free slot we have to traverse long sequence unnecessarily . for example suppose we want to insert 90 and 91 in hash table .

90%10=0 => after 6 collision it will be stored at slot address 6.

91%10=1=> after 6 collision it will be stored at slot address 7.

This problem of primary clustering is solved with the help of Quadratic Probing

Quadratic Probing :

In linear probing we have

(k+i)%table size , k is the slot address/ key value and i is the probe number . In linear probing we were incrementing the value of i linearly which cause problem of primary clus-

tering . In Quadratic Probing we increment the value of i Quadratically . In Quadratic Probing we have
(K+i^2)% table size , for all iε{0,1.table size-1} Probe function can be defined P(K,i)=i^2 , In this technique , two keys with different initial address will not cluster to same address . for example
Key values : 24, 44,84,88
Hash function: key %10
24%10=4 ,slotted to address 4
44%10=4,collision⇒4+12%10=5 , slotted to address 5
84%10=4, collision ⇒4+12%10=5, collision ⇒ 4+22%10=8, slotted to address 8
88%10=8, collision ⇒8+12%10=9 , slotted to address 9.
Disadvantage of Quadratic probing is that probe sequence does not visit all the slots of the table . Actually probe sequence also depends on the size of table . but in most of tables , Probe function will cycle through the small no of slots in the table . suppose page table size is 105 and we starts from any random address in the table then a specific key will visit maximum of 24 slots in the table after that cycling of probe sequence starts. So if these 24 slots are full but some other slots in the table are empty then it shows the message that table is full. To avoid this situation we need to select the table size wisely so that we can get maximum slots in the probe sequence. Some of good hash function for Quadratic probing are
i) If table size is prime and probe function P(k,i)=i^2 ,then at least half of slots in the table are visited.
ii) If table size is power of 2 and probe function P(k,i)=(i^2+i)/2 then almost all the slots in the table are visited.
Also in the Quadratic probing , if some keys generate same starting address then the probe sequence generated by these keys will be same .So we will get the cluster of keys ,having same starting slot address . This Problem is called secondary clustering . Reason for secondary clustering is , same hash function and same additive in case of collision . So to avoid secondary clustering we use DOUBLE HASHING .
Double Hashing :In double hashing we use the two hash function , first hash function determine the probe address and second determine the offset in case of collision . offset value is fixed in case of linear probing(offset=1) and quadratic probing (offset is determined by P(k,i)=i^2), But in Double probing offset value is varying , determined by second hash function. Probe function P(k,i) for double hashing is defined as P(k,i)=i*h'(k)
Table size in double hashing: In double hashing probe sequence constants should be relatively prime to the table size .Now how to ensure whether probe sequence constants are relatively prime to the table size? To ensure this either i) Table size should be a prime number and output of h'(k) ranges between 1 to tablesize-1.
ii) Or , table size should be power of 2 i.e tablesize=2^k for some k and output of h'(k) should be odd value between 1 to 2^k .

for Example:Suppose size of the hash table is 12.Open addressing and double hashing is used to resolve collisions.The hash function used is H(k) = k mod 12
The second hash function is: H2(k) = 7 (k mod 7)
following sequence of insertions in the hash table ?
33, 10, 9, 13, 12, 45
33%12=9
10%12=10
9%12=9,collision ⇒ (7-(9%7))=5⇒ 9+(1*5)=14%12=2
13%12=1
12%12=0
45%12=9,collision ⇒ (7-(45%7))=4⇒ 9+(1*4)=13⇒ 13%12=1,collision,9+(2*4)=17⇒ 17%12=5

Table size to avoid collision in hashing:

To minimize the collision , m=$\Omega(n^2)$ where m is the table size and n be the number of keys to be inserted into the table.
Proof:
Suppose a hash function h maps arbitrary keys to values between 0 and m - 1, inclusive. We hash n keys, k_1, k_2, . . . , k_n. If h(k_i) = h(k_j), we say that k_i and k_j collide. Any two keys collide with probability 1/m. There are $\binom{n}{2}$ pairs of elements, so the expected number of collisions is O(n^2/m). Thus, m=$\Omega(n^2)$. ★If we maintain m =$\Theta(n)$, then search and insert will take O(n) time .

GATE PROBLEM:

1. Consider the hash table with 9 slots. The hash function is h(K) = k mod 9. The collision are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively are
(a) 3, 0 and 1 (b) 3, 3 and 3 (c) 4, 0 and 1
(d) 3, 0 and 2

GATE-2014

2. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first three insertions?
(a) $(97 \times 97 \times 97)/100^3$ (b) $(97 \times 98 \times 97)/100^3$
(c) $(97 \times 96 \times 95)/100^3$ (d) $(97 \times 96 \times 95)/(3! \times 100^3)$

GATE-2014

Statements for Linked Answer Questions 3 and 4

A hash table of length 10 uses open addressing with hash function h(k) = k mod 10, linear probing. After inserting 6 values into an empty hash table , the table is a shown below.

0	
l	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

3. Which one of the following choices gives a possible order in which the key values have been inserted in the table?

(a) 46, 42, 34, 52, 23, 33 (b) 34, 42, 23, 52, 33, 46

(c) 46, 34, 42, 23, 52, 33 (d) 42, 46, 33, 23, 34, 52

4. How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table shown above?

(a) 10 (b) 20 (c) 30 (d) 40

GATE-2010

5. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing . What is the resultant hash table?

a)

0	
l	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

b)

0	
l	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

c)

0	
l	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

d)

0	
l	
2	12,2
3	13,3,23
4	
5	5,15
6	
7	
8	18
9	

GATE-2009

6. Consider a hash table of size 11 that uses open addressing with linear probing . Let h(k) = k mod 11 be the hash function used. A sequence of records with key 43 , 36 , 92 , 87 , 11 , 4 , 71 , 13 , 14 is inserted into an initially empty hash table, the bins of which are indexed from zero to ten. What is the index of the bin into which the last record is inserted?

(a) 3 (b) 4 (c) 6 (d) 7

GATE-IT:2008

7. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4) mod 7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing?
Note that '-' denotes an empty location in the table.
(a) 8,-,-,-,-,-10 (b) 1, 8, 10,-,-,-,3
(c) 1,-,-,-,-,-,3 (d) 1, 10, 8,-,-,-,3

GATE-2008

8. Consider a hash function that distributes keys uniformly. The hash table size is 20. After hashing of how much keys will the probability that any new key hashed collides with an existing one exceed 0.5
(a) 5 (b) 6 (c) 7 (d) 10

GATE-IT:2007

9. A hash table contains 10 buckets and uses linear probing to resolve collisions. The key values are integers and the hash function used is key % 10. If the values 43, 165, 62, 123, 142 are inserted in the table, in what location would the key value 142 be inserted.
(a) 2 (b) 3 (c) 4 (d) 6

GATE-IT:2005

SOLUTION:
1. (a)
5 Mod 9 = 5
28 Mod 9 = 1
19 Mod 9 = 1
15 Mod 9 = 6
20 Mod 9 = 2
33 Mod 9 = 6
12 Mod 9 = 3
17 Mod 9 = 8
10 Mod 9 = 1
Maximum length of chain = 3
Minimum length of chain = 0
Average length of chain = (0 + 3 + 1 + 1 + 0 + 1 + 2 + 0 + 1) / 9 = 1
Option (a) is correct.
2. (a) All the three key can be placed at any slot. Even all the key can be placed on same slot (Collision is resolved by chaining) So sample space $=100^3$
Let First three slots are not filled
No. of ways to insert 1st key on 97 slots = 97
No. of ways to insert 2nd key on 97 slots = 97
No. of ways to insert 3rd key on 97 slots = 97
Probability = $(97 \times 97 \times 97)/100^3$

Option (a) is correct.
3. (c)
⋆Key value 42, 23, 34, 46 are on their slots so their ordering does not matters.
⋆key value 52 is on slot no. 6, it means there were three collision so 52 will always be after 42, 23, 34, in the sequence.
⋆33 is slotted on slot No. 7 but it should be on slot No. 3, it means there are 4 collisions so 33 will be at Last in sequence.
4. (c)
In the given key sequence 42, 23 and 34 will always appear before 52 and 33 but ordering between 42, 23, and 34 does not matters so there are total 3! Sequence. Also 46 will appear any of the five places before 33. So total sequence are 3! $\times$ 5= 30.
5.(c)
12 %10=2
18%10=8
13%10=3
2%10=2 ,collision $\Rightarrow$ (2+1)%10=3,collision $\Rightarrow$ (3+1)%10=4.
3%10=3,collision $\Rightarrow$ (3+1)%10=4,collision $\Rightarrow$ (4+1)%10=5.
23%10=3,collision $\Rightarrow$ (3+1)%10=4,collision $\Rightarrow$ (4+1)%10=5,collision $\Rightarrow$ (5+1)%10=6.
5%10=5,collision $\Rightarrow$ (5+1)%10=6,collision $\Rightarrow$ (6+1)%10=7.
15%10=5,collision $\Rightarrow$ (5+1)%10=6,collision $\Rightarrow$ (6+1)%10=7,collision $\Rightarrow$ (7+1)%10=8,collision $\Rightarrow$ (8+1)%10=9.
6. (d)
43%11=10.
36%11=3.
92%11=4.
87%11=10,collision $\Rightarrow$ (10+1)%11=0.
11%11=0,collision $\Rightarrow$ (0+1)%11=1.
4%11=4,collision $\Rightarrow$ (4+1)%11=5.
71%11=5,collision $\Rightarrow$ (5+1)%11=6.
13%11=2.
14%11=3,collision $\Rightarrow$ (3+1)%11=4$\Rightarrow$(4+1)%11=5,collision $\Rightarrow$ (5+1)%11=6,collision $\Rightarrow$ (6+1)%11=7.
7.(a)
1:7%7=0
3:13%7=6
8:28%7=0
10:34%7=6
As we are using the closed hashing ,On collision element is not inserted into the table so option a is correct
8(a) probability of inserting a new key will depends on previous insertions. p(new key —

m keys)
We are asking to insert the new key with probablity exceeds 0.5
0.5 < probabiltity of collision when (m + 1) keys are inserted.
⇒ 0.5 < 1 - p(all are hashed to a different location)
after insertion of 5 keys
p(all are hashed to different location) = 1 * 19/20 * 18/20 * 17/18 * 16/20 = 0.581 ⇒ 1 - 0.581 = 0.418 < 0.5.after insertion of 6^{th} keys it exeedes 0.5. so, after hashing of 5 keys new key hashsed collides with an existing one exceed 0.5. 9.(d)
43 % 10=3
165 % 10=5
62 % 10=2
123 % 10=3,collision ⇒ (3+1)%10=4
142% 10=2,collision ⇒ (2+1)%10=3 ,collision ⇒ (3+1)%10=4,collision ⇒ (4+1)%10=5,collision ⇒ (5+1)%10=6

practice problems:
Common data for Questions 1-2
Suppose that we have a hash table with 2n slots, with collisions resolved by chaining, and suppose that n keys are inserted into the table. Assume simple uniform hashing, i.e., each key is equally likely to be hashed into each slot.
1. What is the expected number of elements that hash into slot i?
a)0.5 b)0.6 c)0.8 d)0.3
Solution: E[X_{ij}] = 0*P_r[j does not hash into slot i] + 1*P_r[j hashes into slot i]
The probability that an element will hash into slot i is $\frac{1}{2n}$, assuming simple uniform hashing, so E[X_{ij}] = $\frac{1}{2n}$.
Let N_i be the number of elements that hash into slot i*N_i =$\sum_{j=1}^{n} X_{ij}$, so, by linearity of expectation, E[N_i] = $\sum_{j=1}^{n} E[X_{ij}] = \sum_{j=1}^{n} \frac{1}{2n} = \frac{1}{2}$
2.What is the probability that exactly k keys hash into slot i?
a)$\binom{n}{k}\left(\frac{1}{2n}\right)^k\left(\frac{1}{n}\right)^{n-k}$ b)$\binom{n}{k}\left(\frac{1}{n}\right)^k\left(\frac{n-k}{n}\right)^{n-k}$
c)$\binom{n}{k}\left(\frac{1}{2n}\right)^k\left(\frac{2n-k}{2n}\right)^{n-k}$ d)$\binom{n}{k}\left(\frac{1}{2n}\right)^k\left(\frac{2n-1}{2n}\right)^{n-k}$
2.(d) It is the probability that some chosen k keys hash into slot i and the other n - k keys hash into any other slot.
So, probability is $\binom{n}{k}\left(\frac{1}{2n}\right)^k\left(\frac{2n-1}{2n}\right)^{n-k}$ Linked Questions 3-4
You are given a hash table with n keys and m slots, with the simple uniform hashing assumption (each key is equally likely to be hashed into each slot). Collisions are resolved by chaining.
3) What is the probability that the first slot ends up empty?
a)$\left(\frac{m}{m^2}\right)^n$ b)$\left(\frac{1}{m}\right)^n$
c)$\left(\frac{1}{2m-1}\right)^n$ d)$\left(\frac{m-1}{m}\right)^n$
3.(d) Independently, each key has a 1/m probability of hashing into the first slot, or (m

- 1)/m probability of not hashing into the first slot. Thus, the probability that no key hashes into the first slot is
$\left(\frac{m-1}{m}\right)^n$
4.What is the expected number of slots that end up not being empty?
a)$m\left(1-\left(\frac{m-1}{m}\right)^n\right)$ b)$m\left(\frac{1}{2m-1}\right)^n$
c)$\left(1-\frac{1}{m}\right)^n$ d)$m\left(\frac{1}{m}\right)^n$
4.(a) Let X_i be the event that slot i is nonempty. Since $X_i = 1$ when slot i is nonempty and is 0 otherwise, $E[X_i] = Pr(X_i = 1)$ is the probability of slot i being nonempty.
The number of nonempty slots is $\sum X_i$. By linearity of expectation, the expected number of nonempty slots is $E[\sum X_i] = \sum E[X_i]$. From Question 3, the probability that the first slot is nonempty, or $E[X_1]$, is then $\left(1-\left(\frac{m-1}{m}\right)^n\right)$, and is the same for all slots. Thus, the expected number of nonempty slots is $m\left(1-\left(\frac{m-1}{m}\right)^n\right)$
5.Suppose size of the hash table is 11. Open addressing and double hashing is used to resolve collisions.
The hash function used is H(k) = k mod 11.
The second hash function is $H_2(k) = 5$ (k mod 5)
What values will be in the hash table after the following sequence of insertions?
16, 23 9, 34, 12, 56

a)

0	
1	23
2	34
3	
4	12
5	16
6	56
7	
8	
9	9
10	

b)

0	
1	34
2	23
3	
4	12
5	56
6	16
7	
8	
9	9
10	

c)

0	
1	34
2	12
3	
4	23
5	16
6	56
7	
8	
9	9
10	

d)

0	
1	23
2	34
3	
4	12
5	9
6	16
7	
8	
9	56
10	

5.(a)
16%11=5
23%11=1
9%11=9
34%11=1,collision ⇒ (5-(34%5))=1⇒ 1+(1*1)=2
12%11=1,collision ⇒ (5-(12%5))=3⇒ 1+(1*3)=4
56%11=1,collision ⇒ (5-(56%5))=4⇒ 1+(1*4)=5⇒ 1+(2*4)=9⇒ 1+(3*4)=13%11=2⇒ 1+(4*4)=17%11=6
6.What is the suggested load factor for a hash table that uses open addressing?
a)0.3 b)0.6 c)0.8 d)0.5
6.(d)

Miscellaneous problems:

Suppose there are two hash tables T_1 and T_2, each of size m, and two different hash functions h_1 and h_2 and decides to use$T_2$to resolve collisions in T_1. Specifically, given an element x, he first calculates v(x) and tries to insert x into T_1. If there is a collision, he calculates h_2(x) and inserts x into T_2.

1.Suppose we are using following hash functions for each table:

i). h_1(x) = $(a_1)^x$ mod m for table T_1

ii). h_2(x)) = $(a_1)^x$mod m for table T_2 He first tries a_1 =a_2 and uses chaining in table T_2.

Consider the following statements

I)This is likely to result in fewer collisions than if he had just used one hash table of size m with chaining

II)This is likely to result in more collisions than if he had just used one hash table of size m with chaining

which of above statement(s) is true

a)I only b)II only c) Both d)None

1.(b)

I is false . Items that collide in T_1 also collide in T_2.

2.Assume m = 21. consider the following set of values

(I) a1 = 9, a2 = 5

(II) a1 = 5, a2 = 7

(III) a1 = 4, a2 = 16

(IV) a1 = 13, a2 = 11

Which among the above will result in the fewest collisions in T_2 .

a)I and II b) II and III c) I and IV d) III and IV

2.(c) There were two correct solutions. first is a1 = 13 and a2 = 11 because neither is a power of the other nor a1 and a2 were relatively prime to each other and both are relatively prime to 21. if we worked out the residuals modulo 21 and we find that a1 = 9, a2 = 5 had the most possible residuals.

3.Consider the case in which T_2 uses linear probing to deal with collisions. Also, assume that h_1 satisfies the simple uniform hashing assumption and h_2 satisfies the uniform hashing assumption. We insert 4 elements. What is the probability that while inserting the fourth element there are at least two collisions?

a)$\frac{2}{m^4}$ b)$\frac{(2(m-1))}{m^4}$ c)$\frac{2m-4}{m^4}$ d)$\frac{6m-4}{m^4}$

3.(d) There must be one collision in the first hash table and (at least) one in the second. There are four ways this can occur: 1) k_1 and k_2 mapped to the same position in T_1 (1/m), k_3 did not collide in T_1 ((m - 1)/m), k_4 collided with k1 or k3 in T_1 (2/m), and k4 collided with k_2 in T2 (1/m). The probability of this case is (1/m)((m - 1)/m)(2/m)(1/m)

=(2(m - 1))/m^4.
2) k_1 and k_3 collide in T_1, while k_2 does not. Has the same probability as the first case (just different order), (2(m - 1))/m^4.
3) k_2 and k_3 collide in T_1. Again the same probability of (2(m - 1))/$m^4$4) k_1,k_2k_2, and k_3 mapped to the same position in T_1 ($1/m^2$), k_4 collides with k_1 in T_1 (1/m), and with k_2 or k_3 in T_2 (2/m). The probability is ($1/m^2$)(1/m)(2/m) =$2/m^4$
The total probability is therefore$\frac{6m-4}{m^4}$