LectureMaterial#2

Information Security

Dr. Abu Nowshed Chy

Department of Computer Science and Engineering

University of Chittagong

Faculty Profile

Hashing

Hash tables are used for keeping values with a key.

Just imagine a locker. You can only open them if you have the keys.

Hashing

Assume you have a hash table named **Users** wherein username is used as key and the value is the name. It will be like this:

1st record in Hash Table
Key: jsmith
Value: John Smith

2nd record in Hash Table
Key: jdoe
Value: Jane Doe

Properties of Hash Function

Property 1: Deterministic

No matter how many times you parse through a particular input through a hash function you will always get the same result.

Property 2: Quick Computation

The hash function should be capable of returning the hash of an input quickly. If the process isn’t fast enough then the system simply won’t be efficient.

Properties of Hash Function

Property 3: Pre-Image Resistance

What pre-image resistance states is that given H(A) it is infeasible to determine A, where A is the input and H(A) is the output hash.

Property 4: Small Changes In Input Changes the Hash

Even if you make a small change in your input, the changes that will be reflected in the hash will be huge.

Properties of Hash Function

Property 5: Collision Resistant

Given two different inputs A and B where H(A) and H(B) are their respective hashes, it is infeasible for H(A) to be equal to H(B).

Property 6: Puzzle Friendly

It should be difficult to select an input that provides a pre-defined output. Thus, the input should be selected from a distribution that's as wide as possible.

Hashing

Consider inserting the keys

10, 22, 31, 4, 15, 28, 17, 88, and 59

into a hash table of length m=11 using open addressing with the primary hash function h(k) = k mod m. Illustrate the result of inserting these keys using collision avoidance through linear probing. What is the resultant hash table?

Hashing

0	22
1	88
2	
3	
4	4
5	15
6	28
7	17
8	59
9	31
10	10

Hashing (Linear Probing)

Consider inserting the keys

12, 18, 13, 2, 3, 23, 5 and 15

into a hash table of length m=10 using open addressing with the primary hash function h(k) = k mod m. Illustrate the result of inserting these keys using collision avoidance through linear probing.

Hashing (Linear Probing)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

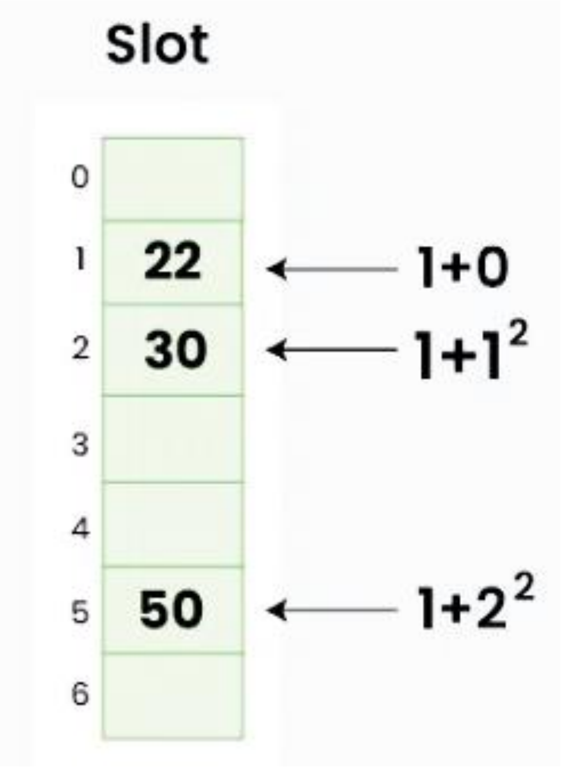
Hashing (Quadratic Probing)

let hash(x) be the slot index computed using hash function.

*If slot hash(x) % S is full, then we try (hash(x) + 1*1) % S*
*If (hash(x) + 1*1) % S is also full, then we try (hash(x) + 2*2) % S*
*If (hash(x) + 2*2) % S is also full, then we try (hash(x) + 3*3) % S*

Hashing (Quadratic Probing)

Example: Let us consider table Size = 7, hash function as Hash(x) = x % 7 and collision resolution strategy to be f(i) = i^2 . Insert = 22, 30, and 50.



Hashing (Double Hashing)

let hash(x) be the slot index computed using hash function.

*If slot hash(x) % S is full, then we try (hash(x) + 1*hash2(x)) % S*
*If (hash(x) + 1*hash2(x)) % S is also full, then we try (hash(x) + 2*hash2(x)) % S*
*If (hash(x) + 2*hash2(x)) % S is also full, then we try (hash(x) + 3*hash2(x)) % S*

Hashing (Double Hashing)

Example: Insert the keys 27, 43, 692, 72 into the Hash Table of size 7. where first hash-function is h1(k) = k mod 7 and second hash-function is h2(k) = 1 + (k mod 5)

Slot

0	
1	**43**
2	**692**
3	
4	
5	**72**
6	**27**

The next key is **72** which is mapped to **slot 2** **(72 % 7 = 2)**, but location **2** is already occupied. Using double hashing,

hnew = [h1(72) + i * (h2(72)] % 7
= [2 + 1 * (1 + 72 % 5)] % 7
= 5 % 7
= 5,

Now, as **5** is an empty slot, so we can insert **72** into **5th slot**.


Thank You!