**Recitation 9**

**Hashing**

1. Answer the following and explain

1. What is the time complexity of insertion in a hash table?

O(1)

1. What is the time complexity of searching in a hash table?

O(1)

1. What is the time complexity of deletion in a hash table?

O(1)

1. Is chained hashing best used when there is a low load factor or high load factor?

Load fac = #elem/total cap 🡪 high load fac means good with chained hashing

1. Is linear probing best used when there is a low load factor or high load factor?

Low load fac

2. Inser​​t the following numbers into this hash table below, using the specified probing method and the given hash function.

**h(k) =(2k)mod 11**

**Values = {19, 65, 09, 17, 01, 99, 12, 4, 6, 23, 22}**

Linear Probing:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Value | 99 | 17 | 1 | 12 | 6 | 19 | 23 | 9 | 4 | 65 | 22 |

Quadratic Probing: hash(x) + i^2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Value | 99 | 17 | 1 | 12 | 22 | 19 | 23 | 9 | 4 | 65 | 6 |

3. Given a hash table with size 48 with 12 elements in it, what is the load factor? Is it possible to have more than 48 elements in this hash table?

12/48 = .25

Yes, with chained hashing.

4. Consider the following hash function:

**h(k) = k mod 4**

**Values = {33, 17, 9, 40, 44, 59, 12, 19, 38}**

Which values will have h = 0? h = 1? h = 2? h = 3?

**H(33) = 1, h(17) = 1, h(9) = 1, h(40)=0, h(44)=0, h(59)=3, h(12) =0 , h(19)=3, h(38)=2**

Is this an efficient hash function? Explain why or why not.

It’s not efficient. Mod 4 dictates general size of 4 and causes repetition/collision.

5. Consider a hash table of size 11, done via double hashing with the following hash functions:

**h1(k) = k mod 11**

**h2(k) = 5 - (k mod 5)**

**Values = {32, 43, 26, 38, 17, 3, 80, 55}**

Draw the table that results after inserting, in the given order, the values. **(10 minutes)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hash Table Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Hash Table Entries | 55 | 43 |  | 3 | 26 | 38 | 17 |  | 80 |  | 32 |

6. A hash table is defined using an array of 50 IntNode references such that the collisions can

be handled by using chaining, as follows:

public class Table {

private int manyItems;

private IntNode[] keys;

public Table() {

keys = new IntNode[50];

manyItems = 0;}

private int hash(int key) {

return key % 100; } }

Write Java code for the following hash Table methods below. You may assume that IntNode class has all of its standard methods.

1. public void put (int key) //Inserts key into the appropriate “chain” of the hash table

//You may assume that the key isn’t a duplicate

Int position = this.hash(key);

if(keys[position] == null){

IntNode newHead = new Intnode(key);

keys[posotion] = newHead;

}else{

currentNode = keys[position];

while(currentNode.next != null){

currentNode = currentNode.next();

}

IntNode newNode = new IntNode(key);

currentNode.setNext(newNode);

}

b) public boolean containsKey(int key)

//Returns true if key is in the hash table, otherwise it returns false.

int position = this.hash(key);

if(keys[position] == null) { return false;

}else{

IntNode = currentNode = keys[position];

while(currentNode != null){

if(currentNode.getData == key){

return true;

}

currentNode = currentNode.getNext();

}

return false;

}

7. Given an array of integers, find the mode of the array using a hash map ()

public int mode(int[] arr) {

Hashmap<int, int> myCount = new Hashmap<>();

Text, letter

Description automatically generated