Florida International University

COP-3530

PI# 5913959

Assignment-3 Due: Mach 27, 2017



**Problem 1:** Assume the following Binary min heap.

1. If we print the keys using preorder traversal… 🡪 1, 2 4, 5, 3
2. If we print the keys using postorder traversal… 🡪 4, 5, 2, 3, 1

If we print the keys using inorder traversal… 🡪 4, 2, 5, 1, 3

The output is not necessarily sorted for any of the traversals.

**Problem 2:**

1. Max heap insertion 11, 19, 23, 12, 13, 17, 13, 14, 18, and 33
2. Min heap insertion 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2
3. Previous min heap after 3 deleteMin

**Problem 3:**

1. public void replaceKey (Integer oldKey, Integer newKey)

{

//run the loop to get the value of i where it stops

for (int i=1; i<this.currentSize && this.array[ i ] != oldKey ; i++);

if ( i == this.currentSize) // oldKey is not in the array

{

system.out.println (“The key is not present in the array”);

return;

}

this.array[ i ] =newKey;

if (oldKey > newKey)

perlocateUp(i);

else

perlocateDown(i);

return;

}

1. The running time complexity is O (n)

**Problem 4:**

Given input {4371, 1323, 6173, 4199, 4344, 9679, 1989} and a hash function

**h(key) = key mod 10**

1. Linear Probing

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9679 | 4371 | 1989 | 1323 | 6173 | 4344 |  |  |  | 4199 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. Quadratic Probing

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9679 | 4371 |  | 1323 | 6173 | 4344 |  |  | 1989 | 4199 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. Double Hashing. **h2(key) = 7 − (key mod 7)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4371 |  | 1323 | 6173 | 9679 |  | 4344 |  | 4199 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Cannot be inserted 1989. All its possible positions are busy.

1. S = {s1, s2, ..., sm} and T = {t1, t2, ..., tn}

boolean isSubset(S, T, n, m)

{

for (i=0 to n)

hashTable.insert(T[ i ]); // insert element of T in the hash table

for (i=0 to m)

if(!hashTable.contains(S[ i ])) // if the element of S is not in the table

return false;

return true;

}