Mario Bolivar

Mjb160330

CE 3345.001

Assignment 5

1. Given input {4371, 1323, 6173, 4199, 4344, 9679, 1989} and a hash function h(x) = x mod 10, show the resulting:
   1. Separate Chaining hash table

**0**

**1** 4371

**2**

**3** 1323**->** 6173

**4** 4344

**5**

**6**

**7**

**8**

**9** 4199**->** 9679**->** 1989

* 1. Hash Table using linear probing h(x) = x mod 10 P(y) = y -> (h(x) + P(y)) mod 10 note: y is incremented by 1 every time a collision occurs until an open address is found.

**0** 9679

**1** 4371

**2** 1989

**3** 1323

**4** 6173

**5** 4344

**6**

**7**

**8**

**9** 4199

* 1. Hash table using quadratic probing h(x) = x mod 10 P(y) = y\*y -> (h(x) + P(y)) mod 10 note: y is incremented by 1 every time a collision occurs until an open address is found.

**0** 9679

**1** 4371

**2**

**3** 1323

**4** 6173

**5** 4344

**6**

**7**

**8** 1989

**9** 4199

* 1. Hash table with second hash function h(x) = x mod 10 , h2(x) = 7 – ( x mod 7)

(h(x) + I \* h2(x)) mod 10 where i is incremented by 1 every time a collision occurs/is not resolved

**0**

**1** 4371

**2**

**3** 1323

**4** 6173

**5**

**6**

**7** 4344

**8**

**9** 4199

Note: 9679 and 1989 cannot be inserted into the hash table

1. A min-max heap is a data Structure that supports both deleteMin and deleteMax in O(logN) per operation. The structure is identical to a binary heap, but the heap-order property is that for any node, X, at even depth, the element stored at X is smaller than the parent but larger than the grandparent ( where this makes sense), and for any node X at odd depth, the element stored at X is larger than the parent but smaller than grandparent. See Fig.
   1. How do we find the minimum and maximum element?

The minimum and maximum element will simply be at the root of the tree.

* 1. Give an algorithm to insert a new node into the min-max heap

public void insert(int val) {

heapSize++;

      data[heapSize - 1] = val;

      percolateUp(heapSize - 1);

     }

private void percolateUp (int nodeIndex) {

            int parentIndex, tmp;

            if (nodeIndex != 0) {

                  parentIndex = getParentIndex(nodeIndex);

                  if (data[parentIndex] > data[nodeIndex]) {

                        tmp = data[parentIndex];

                        data[parentIndex] = data[nodeIndex];

                        data[nodeIndex] = tmp;

                        siftUp(parentIndex);

                  }

            }

      }

1. Merge the two binomial queues

4

18

15

13

18

16

26

14

65

24

21

12

65

24

51

23

55

29

11

2

1. Give an algorithm to find all nodes less than some value X, in a binary heap. Your algorithm should run in O(K), where K is number of nodes output.
2. private ArrayList<T> heap; //Heap data
3. public void print(T k){
4. System.out.println(print(1, k));
5. }
7. public String print(int node, T k){
8. if(node < heap.size()){ //in range of elements in heap
9. if(heap.get(node).compareTo(k) < 0){ //go through children
10. System.out.print(heap.get(node));
11. print(2\*node, k);
12. print(2\*node+1, k);
13. }
14. }
15. }