Birla Institute of Technology & Science – Pilani, Hyderabad Campus
Second Semester 2015-2016
CS F211: Data Structures & Algorithms
Test 2

Type: Closed **Time: 60 mins** **Max Marks: 50** **Date: 12.04.2016**

1.a. What are the minimum and maximum number of elements in a heap of height h? [3 Marks]

1.b. Where in a max-heap might the smallest element reside, assuming that all elements are distinct? [3 Marks]

1.c. Consider a binary heap. Print the keys as encountered in a preorder travel. Is the output sorted? Justify your answer. Attempt the same question for inorder and postorder travel. [5 Marks]

1.d. Give an algorithm o find all nodes less than some value X in a binary heap. Analyze its complexity. [5 Marks]

2.a. Given the values {2341, 4234, 2839, 430, 22, 397, 3920}, a hash table of size 7, and hash function h(x) = x mod 7, show the resulting tables after inserting the values in the given order with each of collision strategies (chaining, linear probing, quadratic probing).

{2341, 4234, 2839, 430, 22, 397, 3920} [6 Marks]

Note: h(x) = x mod 7, 2341 % 7 = 3, 4234 % 7 = 6, 2839 % 7 = 4, 430 % 7 = 3, 22 % 7 = 1, 397 % 7 = 5, 3920 % 7 = 0

2.b. You wish to store a set of *n* numbers in either a max-heap or a sorted array. For each application below, state which data structure is better, or if it does not matter. Explain your answers. [6 Marks]

(a) Want to find the maximum element quickly.

(b) Want to be able to delete an element quickly.

(c) Want to be able to form the structure quickly.

(d) Want to find the minimum element quickly.

2.c. What is the best and worst case complexity of search operation in a binary search tree. Provide a problem instance of size 10 that achieves the worst case complexity. [4 Marks]

2.d. Show the red-black trees that result after successively inserting the keys 41, 38, 31, 12, 19, 8 into an initially empty red-black tree. [6 Marks]

[PTO]

3. [4 + 4 + 4 Marks]

Consider the problem of searching for x in an array A of n elements:

```
boolean search(n,A,x){   \\searches for x in the unsorted array A[0..n-1]
   int i=0;
   while (i <= n-1 && A[i] != x)   \\searches for x
      i++;
   if (A[i] == x) return true;
   else return false;
```

You are to give an exact answer for number of times that `A[i] != x` is executed.

Along with the arithmetic sum $\sum_{i=1}^{k} i = k(k+1)/2$, another useful summation (obtained by integrating both sides of the geometric series $\sum_{i=0}^{k} x^i = (1-x^{k+1})/(1-x)$ and then setting $x = 1/2$ is

$$\sum_{i=0}^{k} i(1/2)^i = \sum_{i=1}^{k} i(1/2)^i = 2 - \frac{1}{2^{k-1}} - \frac{k}{2^k}$$

(a) Assume that with probability 1/2, `x` is in `A[0]`, with probability 1/4, `x` is in `A[1]` and with probability 1/4, `x` is not in `A`.

(b) Assume that with probability 1/2, `x` is not in `A` and with probality $1/(2n)$, `x` is position i of the list for $i = 0, 1, \ldots, n-1$.

(c) Assume that for $i = 0, 1, \ldots, n-1$, that the probability that `x` is in position i of the array is $(1/2)^{i+1}$ and with probability $(1/2)^n$ that `x` is not in the array.