CSC263: Exam Review

Question 1

Prove or disprove: if two vertices u and v in a directed graph are in the same SCC, then there is no path between u and v that leaves the SCC.

True.

Question 2

We have a very dense undirected graph, and we'd like to find one of its MSTs. We must choose between using Kruskal's Algorithm or Prim's Algorithm. Which one do you choose, and why?

↳ Prim time.

生长的方法 prim（O（V ln V））：通过heap来找最小的weight，每次找是ln n，n个结点，所以n ln n（永远都是这个更好）一次能删除n个edge
组合的方法kruskal (O(2 ln E))：用heap挑出最小的weight，takes ln n time(edge越多越不好，每个都要check一遍，并且每次只能删除一个edge)

Question 3

Consider running Kruskal's algorithm on a graph G with n vertices. During the execution of the algorithm, the partially built MST is a subgraph of G with a number of disjoint connected components. We call a connected component a "significant component" if and only if it contains at least one edge. What is the maximum possible number of "significant components" during an execution of Kruskal's algorithm? Write down your answer in terms of n.

floor(n/2)

Question 4

You are given an ADT called BLACK-BOX with the following description.

Objects: A set S of distinct float values. Let n denote the size of S.

Operations:

- `ADD(S, x)`: Add value `x` to set `S`, if `x` already exists in `S`, do nothing.
- `CONTAINS(S, x)`: Returns whether the set `S` contains the value `x`.
- `MAX(S)`: Returns the maximum element in set `S`. Return `NIL` if the set is empty.
- `SUCCESSOR(S, x)`: return the smallest element in `S` that is larger than `x`. Return `NIL` if `x` is the largest element in `S`.
- `SELECT(S, r)`: Returns the `r`-th smallest element in `S`. You may assume that `r` is no larger than the size of `S`.

In each part below, the BLACK-BOX ADT is implemented using some mystery data structure. You're given the runtimes of some of the operations (worst-case or average-case), and your job is to make an educated guess of **which data structure is being used**. Write down your answer for each part without any justification. The answer should be a simple data structure that we learned in CSC263 rather than a composition of multiple data structures. If you think the runtimes are impossible for any implementation, write "IMPOSSIBLE" as your answer.

A. `CONTAINS` is $\Theta(1)$, `MAX` is $\Theta(n)$.

Hash.

B. `MAX` is $\Theta(\log n)$, `SUCCESSOR` is $\Theta(\log n)$ and `ADD` is $\Theta(\log n)$.

AVL

C. `MAX` is $\Theta(1)$, `CONTAINS` is $\Theta(n)$ and `ADD` is $\Theta(\log n)$.

Impossible.

D. `MAX` is $\Theta(1)$, `SUCCESSOR` is $\Theta(\log n)$ and `ADD` is $\Theta(n)$.

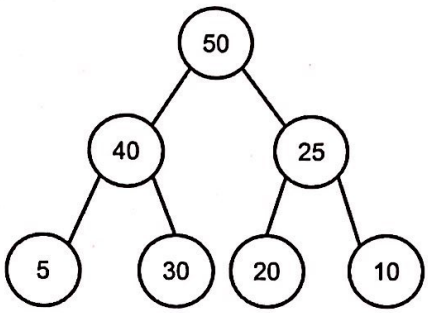
Sorted array.

E. `MAX` is $\Theta(1)$, `ADD` is $\Theta(\log n)$ and `SELECT` is $\Theta(\log n)$.

impossible

Question 5

Consider the following **binary max-heap** where each node's priority is given.



Suppose we INSERT into this heap a new node with priority chosen **uniformly at random between 1 and 100**, inclusive. We are interested in counting the number of **swaps** made by the "bubble-up" portion of the INSERT operation. Keep in mind that if a node's priority is equal to its parent's, no swap occurs.

A. What is the probability that no swap occurs in the INSERT operation?

小于等于5就不换，那就是5/100的概率

B. What is the maximum number of swaps that could occur in the INSERT operation?

0.5的概率换3次

C. What is the expected number of swaps that occur in the `INSERT` operation? Show detailed steps of your calculation. Your final result must be an exact number.

换几次*他的概率
0次*5%+1次*35%+2次*10%+3次*50%=2.05