

## 1 Graph Conceptuals

(a) Answer the following questions as either **True** or **False** and provide a brief explanation:

1. If a graph with  $n$  vertices has  $n - 1$  edges, it **must** be a tree.
2. Every edge is looked at exactly twice in each full run of DFS on a connected, undirected graph.
3. In BFS, let  $d(v)$  be the minimum number of edges between a vertex  $v$  and the start vertex. For any two vertices  $u, v$  in the fringe (recall that the fringe in BFS is a queue),  $|d(u) - d(v)|$  is **always less than 2**.

(b) Given an undirected graph, provide an algorithm that returns true if a cycle exists in the graph, and false otherwise. Also, provide a  $\Theta$  bound for the worst case runtime of your algorithm.

## 2 Fill in the Blanks

Fill in the following blanks related to min-heaps. Let  $N$  is the number of elements in the min-heap. For the entirety of this question, assume the elements in the min-heap are **distinct**.

1. `removeMin` has a best case runtime of \_\_\_\_\_ and a worst case runtime of \_\_\_\_\_.
2. `insert` has a best case runtime of \_\_\_\_\_ and a worst case runtime of \_\_\_\_\_.
3. A \_\_\_\_\_ or \_\_\_\_\_ traversal on a min-heap *may* output the elements in sorted order. Assume there are at least 3 elements in the min-heap.
4. The fourth smallest element in a min-heap with 1000 **distinct** elements can appear in \_\_\_\_\_ places in the heap. (Feel free to draw the heap in the space below.)

5. Given a min-heap with  $2^N - 1$  **distinct** elements, for an element

- to be on the second level it must be less than \_\_\_\_\_ element(s) and greater than \_\_\_\_\_ element(s).
- to be on the bottommost level it must be less than \_\_\_\_\_ element(s) and greater than \_\_\_\_\_ element(s).

*Hint:* A complete binary tree (with a full last-level) has  $2^N - 1$  elements, with  $N$  being the number of levels. (Feel free to draw the heap in the space below.)

### 3 Heap Mystery

We are given the following array representing a min-heap where each letter represents a **unique** number. Assume the root of the min-heap is at index one, i.e. A is the root. Our task is to figure out the numeric ordering of the letters. Therefore, there is **no** significance of the alphabetical ordering. i.e. just because B precedes C in the alphabet, we do not know if B is less than or greater than C.

Array: [-, A, B, C, D, E, F, G]

**Four** unknown operations are then executed on the min-heap. An operation is either a `removeMin` or an `insert`. The resulting state of the min-heap is shown below.

Array: [-, A, E, B, D, X, F, G]

- (a) Determine the operations executed and their appropriate order. The first operation has already been filled in for you!

*Hint: Which elements are gone? Which elements are newly added? Which elements are removed and then added back?*

1. `removeMin()`
2. \_\_\_\_\_
3. \_\_\_\_\_
4. \_\_\_\_\_

- (b) Fill in the following comparisons with either  $>$ ,  $<$ , or  $?$  if unknown. We recommend considering which elements were compared to reach the final array.

1. X \_\_\_\_\_ D
2. X \_\_\_\_\_ C
3. B \_\_\_\_\_ C
4. G \_\_\_\_\_ X