Quiz 2

Recall the following definitions:

- A rooted binary tree is a tree where every node (except the root) has a parent, and every node (except the leaves) has one or two children. The leaf is a node with no children, and teh root is a node with no parent.
- A complete binary tree is a binary tree where all nodes except the leaves have exactly two children. If the maximum root to leaf distance (aka the **height** of the tree) is d, then a complete binary tree has 2^i vertices at a distance of i from the root, for all $1 \leq i \leq d$.
- A heap on a set S organizes the elements of S in an "almost" complete binary tree: such a tree is a complete binary tree with one exception: there may be fewer than 2^d vertices at distance d from the root.
- A **max** heap has the property that every element is greater than or equal to its children.
- A **min** heap has the property that every element is less than or equal to its children.

---

1. **[1 mark]** Suppose we have a **min-**heap S with d layers and $2^d - 1$ elements, i.e, the structure of the heap is a complete binary tree of height d. Assume all elements are distinct.

   Which of the following elements **has** to be a leaf? Assume $d > 2$.

   (A) the second-largest element of S

   (B) the smallest element of S

   (C) the second-smallest element of S

   (D) the third-largest element of S

2. **[2 marks]** Suppose we have a **min-**heap S with d layers and $2^d - 1$ elements, i.e, the structure of the heap is a complete binary tree of height d. Assume all elements are distinct.

   Let A be an 1-indexed array that has the elements of S in sorted order. Note that $A[2^{d-1}]$ is the median element of S. Which of the following elements *cannot* be a child of the root?

   (A) $A[2^{d-1}]$ (B) $A[2^{d-1}+1]$ (C) $A[2^{d-1}-1]$ (D) $A[2^{d-1}+2]$

3. **[2 marks]** Suppose we have an empty **min-**heap and we insert the elements $1, 2, ..., n$ into it in reverse order, where $n = 2^d - 1$. That is, we first insert n, then insert $n-1$, and so on.

   After this sequence of operations is complete, how many times did we execute a swap operation between two elements? Write your answer in terms of either n or d.

   Sanity check: if $n = 7$, the number of swaps is 10:

   - inserting 7 requires no swaps
   - inserting 6 and 5 requires one swap each
   - inserting the remaining four elements requires two swaps each

   For full credit, work out a closed form expression — no summation signs :)

4. **[4 marks]** Suppose you are maintaining a min *and* max heap simultaneously for a set of elements S using the algorithms discussed in class. Let the maximum number of swaps that you have to perform in a single heap be M.

   Assume that k elements have been inserted into both heaps so far, and elements are not repeated. Suppose a new element e is inserted into both the heaps, and this causes you to perform a swaps in the min heap and b swaps in the max heap. Justify your answers to the following questions with examples or an argument.
   Assume that e is distinct from all the k elements already present in the heap.

   (A) Is it possible that $a + b = 2M$?

   (B) Is it possible that $a + b = M$?

   (C) Is it possible that $a = b = 1$?

   (D) Is it possible that $a = b = 0$?

5. **[2 marks]** Build a min-heap out of the following sequence and report the final heap as an array, and the number of swaps.

   408, 248, 308, 399, 484, 32, 439, 403, 87, 10

6. **[2 marks]** A **vertex cover** of a simple undirected graph G is a subset of vertices S such that for every edge $e = (u, v)$ has at least one of its endpoints in S, i.e, either $u \in S$ or $v \in S$ (or both).

   Which of the following is a vertex cover of a graph G?

   (A) the leaves of a DFS tree of G

   (B) all vertices *except* the leaves of a DFS tree of G

   (C) the leaves of a BFS tree of G

   (D) all vertices *except* the leaves of a BFS tree of G

7. **[2 marks]** Consider the slightly buggy implementation of BFS below:

```
current <- {start_vertex}
while current is not the set of all the vertices
    found <- {}
    for v in current:
        for each w adjacent to v:
            add w to found
    current <- found
```

   Some undirected graphs are described in terms of their edge lists below. For each, determine if the program above will terminate or not. Answer YES or NO.

   (a)

```
(1 2), (1 3), (2 3)
```

   (b)

```
(2, 1), (3, 5), (1, 6), (2, 5), (3, 1), (8, 4), (2, 7), (7, 1), (7, 4)
```