CSE202. Theory Problem 8.1. Priority Queues & Disjoint Sets.

Moyuan Huang. A53212613.

Computing k-th smallest element in heap.

Input: A[1...n]. (heap)

Output: The k-th smallest elements in A.

Running Time: O(klogk).

1. Algorithms:

```
1. Create a new heap pq2 with its root equals to the original heap's root.
2. ~~pq2.p~~ current_root ← pq2.top();
3. pq2.pop();
4. pq2.push(children_of(current_root));
5. repeat 2-4 for k times.
6. return pq2.top();
```

2. Explanation: Each time we repeat line 2-4 we a) delete one node, b) add two nodes. The deleted node is the smallest node so far (since we are using small-heap). After adding two new nodes to the pq2, the next smaller value is then stored in the now root of it.

The reason we choose to add the two children of the removed node, is that they are the only two nodes that could potentially be the pq2's root in the original heap. (the descendants of the two children are all bigger than them).

After repeating 2-4 k times, we return the root of pq2, which is the k-th smallest element of the original heap.

3. running time: ∆. Line 3 ~~took~~ takes O(logk) time at most. (delete a node in a heap which has height logk).

∆. Line 4 takes O(2logk) time (insert 2 new value to a heap which has height logk).

∆. Line 2-5 thus takes O(3logk)·k = Θ(klogk) time.

∆. The algorithm has time complexity of O(klgk).