CSCI/CMPE 3333 Assignment Three
Instructor: Zhixiang Chen

In this homework assignment, I would like you to do two parts: The first is to carry out some experimental study of the selection problem. The second is to implement a min-max heap.

- **Part 1 (100 points). The Selection Problem**: Assume a list of *N* elements is given. For any $k, 1 \le k \le N,$ find the *k*-th smallest elements in the list.

  Note 1: the *1*-th smallest element is the smallest, the 2-th smallest is the second smallest, and so on, and the *N*-th smallest is the largest element.

  Note 2: For the purpose of this homework, we may focus on integer type elements only. We further assume that all elements may not be distinct.

  Solution 1: One direct solution is to first build a binary min-heap for the list of elements (here, the root has the smallest element), then perform *k* *deleteMin( )* operations. The last one from the k-th *deletionMin* operation is the answer. *Hint: you shall use the clever O(N) time algorithm to build the heap.*

  Solution 2: This solution replies on the idea of *median3* quick sort. First, choose the median of the first, middle and last elements as a pivot *p*. Second, split the list into two sublists *A* and *B* such that, every element in *A* is less than *p* and every element in *B* is greater than *p*. Third, we consider the following three cases:

  - Case 1: If $|A| = k-1,$ then *p* is the answer. (Recall that $|A|$ denotes the size of *A*.)
  - Case 2: If $|A| \ge k,$ then the *k*-th smallest element is in *A*, so that we recursively solve the problem for *A* and *k* using the same idea of median3 quick sort.
  - Case 3: If $|A| < k-1,$ then the *k*-th smallest element is the $(k-|A|-1)$ *-th* smallest element in B, so that we recursively solve the problem for B and $k' = k-|A|-1$ using the same idea of median3 quick sort.

  Your Work: I'd like you to do:

  - Implement Solution 1 and Solution 2.
  - Randomly generate 10 lists so that each may have 100,000 integers.
  - For each list, randomly generate 5 values for *k* and then run Solution 1and Solution 2 for each k and record the respective time. Calculate the average time for each solution.
  - Use a bar chart to report the average times of the two solutions for the 10 lists.
  - Turn in your implementation of Solution 1 and Solution 2 and the bar chart report.

- **Part 2 (100 points). Implement a min-max binary heap:** Recall a typical binary min-heap (or max-heap) will always save the minimum element (or maximum element) at the root, so that finding the minimum (or the maximum) takes constant time, and *deleteMin* (or *deleteMax*) takes O(log N) time. However, a min-heap (or max-heap) cannot help you to easily find the maximum (or the minimum) unless you search through all elements at leaves, hence O(N) time is needed for such operation. On the contrary, however, a binary min-max-heap guarantees that both *deleteMin* and *deleteMax* can be done in O(log N) time.

  A binary min-max heap is identical to a *binary min-heap* (or *max-heap*) in structure, but its order property is different. For any node X at even depth, the element stored at X is smaller than the parent but larger than the grandparent (where this makes sense). For any node X at odd depth, the element stored at X is larger than the parent but smaller than the grandparent. Below is an example of *a binary min-max-heap*.

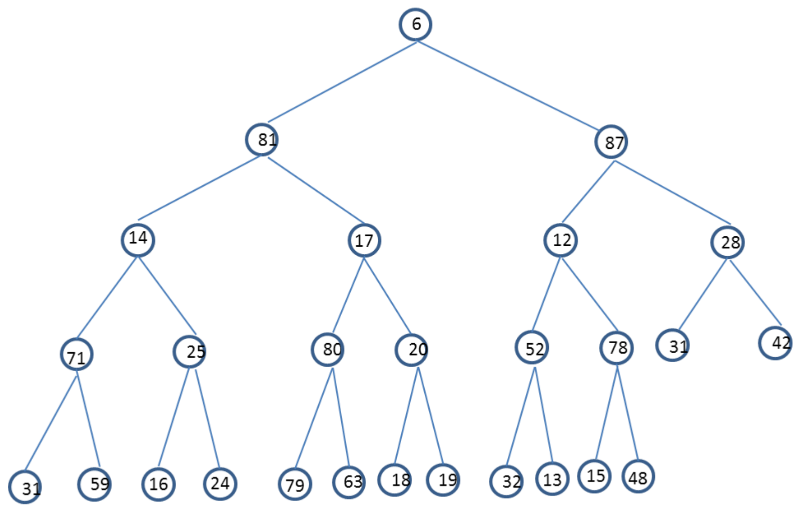

Figure 1: A binary min-max-heap

  Your work: I would like you to extend *the binary min-heap* (or *max-heap*) class template to *the binary min-max-heap*. You need to define a new class template with both a pair of getMin() and getMax() and a pair of *deleteMin( )* and *DeleteMax( )* method. You also have to refine methods for insertion, deletion, and search operations. Test your program with the following numbers to see if it builds *a binary min-max-heap* right and can do *deleteMin( )* and *deleteMax( )* successfully.

  Tests:
  (1) Save the following list of integers into a text file and use your program to build a binary min-max-heap:

  48, 63, 31, 42, 14, 81, 17, 6, 59, 52, 28, 87, 80, 12, 32, 71, 18, 25, 13, 16, 15, 20, 24, 78, 19, 79

  (2) Perform four operations: deleteMin( ), DeleteMax( ), deleteMin( ), deleteMax( ).

Caution: DO NOT PRINT OUT THE DATA FILES!
Due Date: The due date will be given via Blackboard.
Warning: Any submission one week after the due date will not be accepted.
How to submit your work?
Please upload your source program files and your test results to Blackboard.