CMPSC 465 Data Structures & Algorithms
Fall 2025 Ke Chen and Yana Safonova

HW 3

1. (20 pts.) **Heap Basics.** The array $A := [23, 4, 14, 3, 9, 13, 11, 10]$.

(a) Run build-heap on A to construct a min-heap and write down the resulting array.

(b) Insert the element 5 and write the resulting array.

(c) Insert the element 1 and write the resulting array.

(d) Delete 14 and write the resulting array.

Answers:

(a) $[3, 4, 11, 10, 9, 13, 14, 23]$

(b) $[3, 4, 11, 5, 9, 13, 14, 23, 10]$

(c) $[1, 3, 11, 5, 4, 13, 14, 23, 10, 9]$

(d) $[1, 3, 9, 5, 4, 13, 11, 23, 10]$

2. (30 pts.) **Heaps and Heap Sort.**

(a) What are the minimum and maximum numbers of nodes in a heap of height h?

(b) Is the array with values $\{10, 14, 19, 35, 31, 42, 27, 44, 26, 33\}$ a Min heap?

(c) Show that in the worst-case Heapify-UP could make $\Omega(\log n)$ swaps on a heap with n elements. (Hint: Give an example heap with n node values that would cause Heapify-UP to be called recursively at every node on a simple path up to the root).

Answer:

(a) The answer is at least 2^h and at most $2^{h+1} - 1$. This is because a complete binary tree of height $h-1$ has $\sum_{i=0}^{h-1} 2^i = 2^h - 1$ elements, and the number of elements in a heap of depth h is strictly larger than the number of vertices in a complete binary tree of height $h-1$ and less than (or equal) the number of nodes in a complete binary tree of height h.

(b) The array is not a Min heap. The node containing 26 is at position 9 of the array, so its parent is at position 4, which contains 35. This violates the Min Heap Property.

(c) Consider the min heap with n vertices where the root and every other node contains the number 2. Suppose now that 1 is inserted to the first available position at the lowest level of the heap. That is, $A[i] = 2$ for $0 \leq i \leq n-1$ and $A[n] = 1$. Since 1 is the minimum element of the heap, when Heapify-UP is called from position n, the node containing 1 must be swapped through each level of the heap until it is the new root node. Since the heap has height $\lfloor \log n \rfloor$, Heapify-UP has worst-case time $\Omega(\log n)$.

3. (30 pts.) **Median of Streaming Data.** Consider an array of N numbers $[x_1, x_2, \ldots, x_N]$ that you will be receiving one-by-one in a single pass (i.e., as a stream of numbers). You are not allowed to revisit previous numbers. Design an effective data structure involving one or more heaps that, after receiving the n-th number, reports the median of the numbers $x_1, x_2, \ldots, x_N$ observed so far. The time complexity of your algorithm should be $O(\log n)$ per number received in the worst case. Write down the pseudocode of your algorithm and analyze its time complexity. For simplicity, assume there are no duplicates in the stream.

Hint: If n is odd, the median of the data stream $[x_1, x_2, \ldots, x_N]$ is the middle element of the sorted data stream, else the median is the average of the middle two elements of the sorted data stream.

Answer:

Maintain 2 heaps. Using induction, assume that there is (a) a max heap for the lower (in value) half of the elements, and (b) a min heap for the upper half of the elements. Given a new element, check the root of the larger (in number of elements) heap.

1. If the larger heap is the max heap, then, if the element is smaller, insert the element into the max heap, delete the root, insert the root into the min heap. If the element is bigger, insert it into the min heap. After 1, (a) and (b) are still satisfied.

2. If the larger heap is the min heap, then, if the element is bigger, insert the element into the min heap, delete the root, insert the root into the max heap. If the element is smaller, insert it into the max heap. After 2, (a) and (b) are still satisfied.

3. If both heaps are the same size, choose the root of the min heap arbitrarily. If the element is bigger, insert the element into the min heap. If the element is smaller, insert it into the max heap. After 3, (a) and (b) are still satisfied.

4. If both heaps are the same size, the median is the average of the roots. If one heap is larger, the median is the root of the larger heap.

Following 1, 2, 3, and the properties of heaps, (a) and (b) will always be satisfied, so 4 always returns the median.

Note that in the base case of the induction of two empty heaps, the element can be inserted into a heap arbitrarily.

There are a constant number of operations taking $O(\log n)$ for each number, so the final runtime is as such.

4. (20 pts.) **Proofs.** Show that an n-element binary heap has height $\lfloor \log_2 n \rfloor$.

Answer:

A binary heap is by definition a full binary tree, meaning every level can only contain nodes if every above level is already full. Therefore, the number of nodes in a tree of height h is greater than or equal to 2^h. For example, A heap of height 2 must have at least 4 nodes (including 1 on level 2). We can similarly show that the number of nodes is less than or equal to $2^{h+1} - 1$. This is because a heap of height h cannot have nodes on level $h+1$. For example, a heap of height 2 can have at most 7 nodes. Adding an eighth node would increase the height to 3. Therefore, $2^h \leq n \leq 2^{h+1} - 1 < 2^{h+1}$. Taking the logarithm gives $h \leq \log_2 n < h+1$. Since h is an integer, $h = \lfloor \log_2 n \rfloor$.

Rubric:

Problem 1, ? pts

?

Problem 2, ? pts

?

Problem 3, ? pts

?

Problem 4, ? pts

?

Problem 5, ? pts

?