
You're expected to work on the problems before coming to the lab. Discussion session is not meant to be a lecture. TAs will guide the discussion and correct your solutions if needed. We may not release solutions for all problems. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor. The instructor will factor in participation in final grade.

1. (Basic) You're given an array $A[1 \cdots 8] = \langle 3, 1, 6, 8, 4, 2, 5, 7 \rangle$. Is this a max-heap? If not, make it a max-heap by iteratively applying the Max-Heapify function. In other words, illustrate the operation of Build-Max-Heap on the array. What is $A[1 \cdots 8]$ at the end?

2. (Basic) Illustrate the operation of Max-Heap-Insert(A, 9) on the heap you obtained in problem 1.

3. (Basic) Illustrate the operation of Heap-Sort on the max-heap you've obtained in problem 2. Note that you don't need to execute line 1 of Heap-Sort, as you already have a max-heap A.

4. (Basic) You're given an array $A[1 \cdots 8] = \langle 3, 1, 6, 8, 4, 2, 5, 7 \rangle$. Is this a min-heap? If not, make it a min-heap by iteratively applying the Min-Heapify function.

5. (Basic) Is an array that is sorted in increasing order a min-heap?

6. (Basic) Is Heap-Sort an in-place sorting algorithm? We say that a sorting algorithm is in-place if only $O(1)$ elements are stored outside the input array at any time – in other words, at most $O(1)$ extra memory is used in the whole process. Do you see why Insertion-Sort is in-place but Merge-Sort is not?

7. (Intermediate) Give a pseudo-code to test if a given binary heap is a max-heap or not.

8. (Intermediate) We want to show that it takes $O(n)$ time to build a max-heap. Towards this end, we will take the following steps:

   (a) Suppose the binary heap has height H. Explain that $2^H \leq n < 2^{H+1}$.
   (b) Explain that there are at most 2^{H-h} nodes of height h in the heap, for all $0 \leq h \leq H$.
   (c) Show that there are $O(n/2^h)$ nodes of height h.
   (d) What is the running time of Max-Heapify on a node of height h?
   (e) From above, the running time for building a max-heap is at most $\sum_{h=1}^{H} O(h) \cdot O(n/2^h)$.
   (f) Show that $\sum_{h=1}^{\infty} h/2^h = O(1)$.

9. (Intermediate) You want to add a new operation to the max-priority-queue, namely Heap-Second-Maximum(A). The function returns the second largest value (key) stored in the heap A. Give a pseudo-code of this operation. What's the running time of your algorithm? The faster, the better.

10. (Advanced) The operation Heap-Delete(A, i) deletes (the item in) node i from heap A. Give a pseudocode of Heap-Delete that runs in $O(\lg n)$ time for an n-element max-heap.

11. (Advanced) Give an $O(n \lg k)$-time algorithm to merge k sorted lists into an sorted list, where n is the total number of elements in all the input lists. Hint: Use a min-heap. Explain the running time.