Week 3
Priority Queue, Heapsort

Preliminary: Lecture on heap data structure

Workshop

PROBLEM A:

Write a program that utilizes heap data structure to sort the list of input numbers

INPUT: a sequence of n numbers. Consecutive numbers are separated by a space

OUTPUT: the list of the n numbers, sorted into *monotonically increasing order*.

Materials:

- SortingTest.zip : the test cases for measuring running time (from Week 2)
- selectionSort.py : a program that performs selection sort
- Heap.py : the heap class

1) Study how the provided selection sort program works. Verify the correctness with a few simple inputs.

2) Add running time recording code (just like what has been done in the past two weeks). Then, measure the running time of your program on the provided test cases.
   - Project the running time increase with the input size, n. What do you conclude as the upperbound of the selection sort running time ?

$$T(n) = \mathrm{O}(\_\_\_\_)$$

3) For this step, Computer Science students are recommended to develop your own heap data structure, but this is not required. If needed, study more about heap data structure from any resource (the recommended textbook or online).

   Because the selection sort always selects the maximum value in each iteration, this operation should be faster with the application of max-heap. The modified algorithm is a variance of *heapsort*.

   Test the resulted program with the provided test cases to see if it really is significantly faster than the original selection sort.

4) Given that each operation of heap data structure takes O(lgn), what would then be the upperbound on the running time of heapsort?

$$T(n) = \mathrm{O}(\_\_\_\_)$$

PROBLEM B:

Test cases : ropesTestCases.zip

There are given n ropes of different lengths, we need to connect these ropes into one rope. The cost to connect two ropes is equal to the sum of their lengths. We need to connect the ropes with minimum cost.

For example, if we are given 4 ropes of lengths 4, 3, 2, and 6. We can connect the ropes in the following ways.

Connect	Cost	Total cost	Ropes left
2 and 3	2+3 = 5	5	4, 5, 6
4 and 5	4+5 = 9	5 + 9 = 14	6, 9
6 and 9	6+9 = 15	14 + 15 = 29	15

INPUT: a sequence of n numbers, each represent the length of the corresponding rope

OUTPUT: the minimum total cost of connecting all ropes into one.

Test your program with the provided test cases. An efficient Python 3 program should not take more than 1.5 second for the largest test case.