Lab 5

September 8, 2021

For each question in this lab, please use the concept of quicksort done in the class.

1. Partition

   You are given an array. You have to partition the array using the first element as the pivot. To this end, use the one pass partition function done in the class. You have to output the number of swaps done during the partition.

   Input: The first input line contain an integer $n(1 \leq n \leq 1000)$. In the second line, n numbers are separated by a space (all numbers are ≤ 1000).

   Output: The number of swaps during the partition.

   Input : 5
   3 4 1 2 5
   Output : 2

   Explaination: The pair (4,2) and (3,1) is swapped.

   Input : 5
   1 2 3 4 5
   Output : 0

   Explaination: No swap is required.

2. QuickSort

   Using the above partition function, implement the quicksort procedure that we learned in the class.

   Input: The first input line contains integer $n(1 \leq n \leq 1000)$. The second line n numbers seperated by a space (all numbers are ≤ 1000).

   Output: n numbers written in increasing order separated by a space.

   Input : 5
   1 2 5 3 4
   Output : 1 2 3 4 5

3. Tweak-Quicksort

   It so happens that you can tweak quicksort and find the k-th smallest element in the array. That is, there is no need to sort the array and then find the element at index k. Can you tweak the quicksort algorithm?

   You are given an array of size n .

   You must tweak quicksort to find the k-th smallest element.

   Input: The first line contains an integer n and k seperated by a space. The second line contains n space-separated integers.

   Output: Print the k-th smallest element in the array.

   Constraints: $n < 100$. All numbers are ≤ 1000.

   Sample Input

   Input : 5 3
   10 21 5 1 3
   Output : 5

4. Decrement The Array

You are given an array in which you can decrement any element any number of times. A number is beautiful if it is a power of 2, e.g. $1, 2, 4, 8, \ldots$. The aim of the decrement operation to make beautiful numbers. A beautiful number is said to be missing in the array if it is not present after you are done with your sequence of decrements. Your job is to find the maximum missing beautiful number after all possible decrements of numbers.

There are many ways in which you can solve this problem. But can you solve it in $O(n)$ time. Moreover, can you solve it using only one pass over the input array.

Input: The first line of the input will be n ($n \leq 1000$). The second line contains n positive numbers seperated by a space. All numbers are ≤ 1000 and greater than 0.

Output: Maximum missing beautiful number.

Input	:	6 1 3 3 3 6 7
Output	:	8

Explanation: 1 is already present in the array.
The element $A[1]$ can be decremented one times to get a 2. The element $A[4]$ can be decremented two times to get a 4. There is no way we can use the decrement operation to make an 8. Thus, the answer is 8.