2/13/18

AIDS FOR EXAM (ON 2/15/2018)

2 SHEETS OF NOTES

~~TWO~~ FRONT + BACK

8 1/2" × 11" readable w/o mag. glass!

1) Radix Sort
2) Skip List Summary
3) Exam Review

→ O(n) sort Size of array = d
"digits" - relatively small

3267	0082	1319	1045	0082
✓0082	0362	1045	0082	0362
✓1266	2285	0362	1266	1045
✓2285	1045	1266	3267	1266
1319	1266	3267	2285	1319
✓0362	3267	2367	1319	2285
✓1045	2367	0082	0362	2367
2367	1319	2285	2367	3267
	↑ ↑	tens	hundred's digit	thousand's digit
	units			

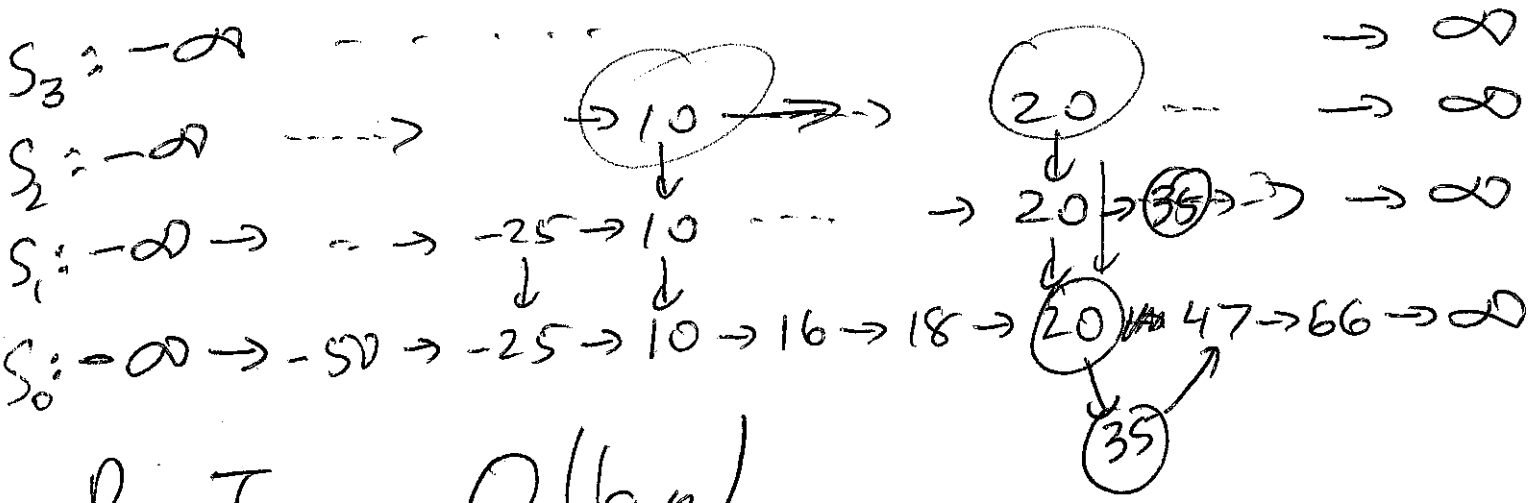
Think of the # binary

V1 - uses bits not digits, do 31 or 32 rounds.

V2 - my "digits" are (bytes) [0, 255]
ONLY DO 4 PASSES/ROUNDS OF COUNTING SORT

Insert 35

(1) Use levels of lists to "hone in on" correct location in bottom list w/o iterating through the whole thing.
(2) Insert in bottom list.
(3) 50% insert in list above, repeat if inserted



Run Time $O(\lg n)$

Memory $O(n) = n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots$

$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right)$

$= n\left(\frac{1}{1 - \frac{1}{2}}\right) = 2n$

Computer Science II Exam #1

Date: 2/14/2013

Last Name: _____, First Name: _____

1) (20 pts) You are given a composite number n, but are asked to give proof that it is composite. You have a test that you can run such that, given a composite number, it proves that it is composite 50% of the time. Thus, to achieve your goal, you'll simply keep on repeating your test until you obtain the proof that the given number is composite. Fill out the chart below indicating the probability your algorithm will run the test various number of times.

Number of Times Test is Run	Probability
1	½
2	
3	
4	
k	

Using the chart above, write an infinite summation, in summation notation equaling the expect number of times your test will have to be run to prove that a given composite number n is composite.

Solve the summation on the following page. You should get a constant value as your answer. (Use the scratch page if you need more room to complete the summation.) Also, no need to use summation notation here. Feel free to write out the pattern of the sum so that it's easier to see your work.

2) (20 pts) Consider the following program description. Complete the code on the following page so it solves the problem.

A k-gap sequence of integers $a_1, a_2, \ldots, a_n$ is such that for every i, $1 \leq i \leq n-1$, $|a_i - a_{i+1}| \geq k$. For example, the sequence 2, 8, 3, 6, 1, 7 is a 3-gap sequence but not a 4-gap sequence, since the difference the consecutive terms, 3 and 6 is 3.

Write a program that reads in a sequence of integers and an integer k and outputs the permutation of the input sequence that is first lexicographically that is a valid k-gap sequence.

Input

The first line of the input file will contain two positive integers, *n (1 ≤ n ≤ 12)*, and *k (1 ≤ k ≤ 10^6)*, where n represents the length of the input sequence for which we are looking for the ordering that is a valid k-gap sequence that is first, lexicographically. The next line will contain the *n* integers.

Output

Output the sequence specified by the problem description with each integer separated by a comma and a space.

Sample Input

```
5 10
13 19 3 2 17
```

Sample Output

```
13, 3, 19, 2, 17
```

```
import java.util.*;

public class gap {

    public static int diff;
    public static int[] values;

    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        int n = stdin.nextInt();
        diff = stdin.nextInt();
        values = new int[n];
        for (int i=0; i<n; i++)
            values[i] = stdin.nextInt();
        Arrays.sort(values);
        printSeq(new int[n], 0, new boolean[n]);
    }

    public static void print(int[] arr) {
        for (int i=0; i<arr.length-1; i++)
            System.out.print(arr[i]+", ");
        System.out.println(arr[arr.length-1]);
    }
```

```
    public static boolean printSeq(int[] cur, int k, boolean[] used) {

        if (k == cur.length) {
            print(cur);
            return true;
        }

        boolean ans = false;
        for (int i=0; i<used.length; i++) {


            /*** FILL IN ALL OF  YOUR CODE HERE ***/









        }

        return ans;
    }

}
```

3) (9 pts) In the O(n) solution to the MCSS problem, there is a single running sum that is tallied as the algorithm iterates through each number in the input sequence. Below, write that running sum ***after*** the i^{th} number has been processed.

Sequence	3	8	-5	2	-10	1	6	-8	12
Sum									

Sum 14 Q1

#times		prob
1	×	$\frac{1}{2}$
2	×	$\frac{1}{4}$
3	×	$\frac{1}{8}$
4	×	$\frac{1}{16}$
5		$\frac{1}{32}$

$$\sum_{i=1}^{\infty} i \cdot \left(\frac{1}{2}\right)^i$$

$$S = 1 \times \frac{1}{2} + 2 \times \left(\frac{1}{2}\right)^2 + 3 \times \left(\frac{1}{2}\right)^3 + \cdots$$

$$-\frac{1}{2}S = \qquad 1 \times \left(\frac{1}{2}\right)^2 + 2 \times \left(\frac{1}{2}\right)^3 + \cdots$$

$$S - \frac{1}{2}S = \frac{1}{2} + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 + \cdots$$

$$\frac{1}{2}S = \frac{\frac{1}{2}}{1 - \frac{1}{2}} \qquad \left[a_1, a_1 r, a_1 r^2, \cdots \quad \sum_{i=0}^{\infty} a_1 \cdot r^i = \frac{a_1}{1-r} \right]$$

$$\frac{1}{2}S = \frac{\frac{1}{2}}{\frac{1}{2}} \Rightarrow \frac{1}{2}S = 1, \quad \boxed{S = 2}$$

Sum 14 Q2

In for loop:

```
if (used[i]) continue;
if (k > 0 && Math.abs(values[i] - values[cur[k-1]]) < diff) continue;
used[i] = true;
cur[k] = i
boolean tmp = printSeq(cur, k+1, used);
if (tmp) return true;
used[i] = false;
```

WANT TO PLACE

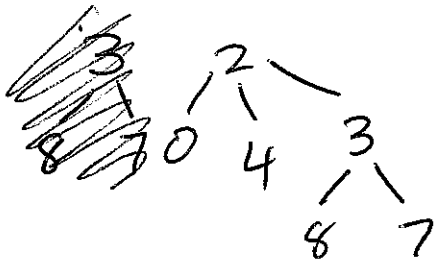
prev item in perm array

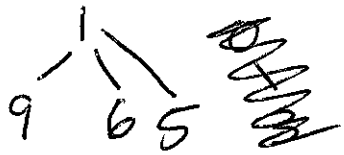
BACKTRACKING PART

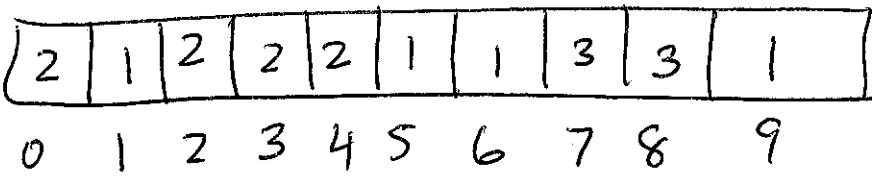
Sum 14 Q4

2/13/18 (5)

- ✓ 3, 8
- ✓ 2, 4
- ✓ 8, 7
- ✓ 4, 7
- ✓ 9, 1
- ✓ 9, 6
- ✓ 3, 0
- 5, 9







Sum 14 Q5

FOR 2-4 Trees, on node overflow, take (3rd) value to split above...