Question: https://leetcode.com/problems/permutation-sequence/

So the idea is pretty simple, i.e., we consider each point and find out it's slope will all other points which occured after that point in points array.

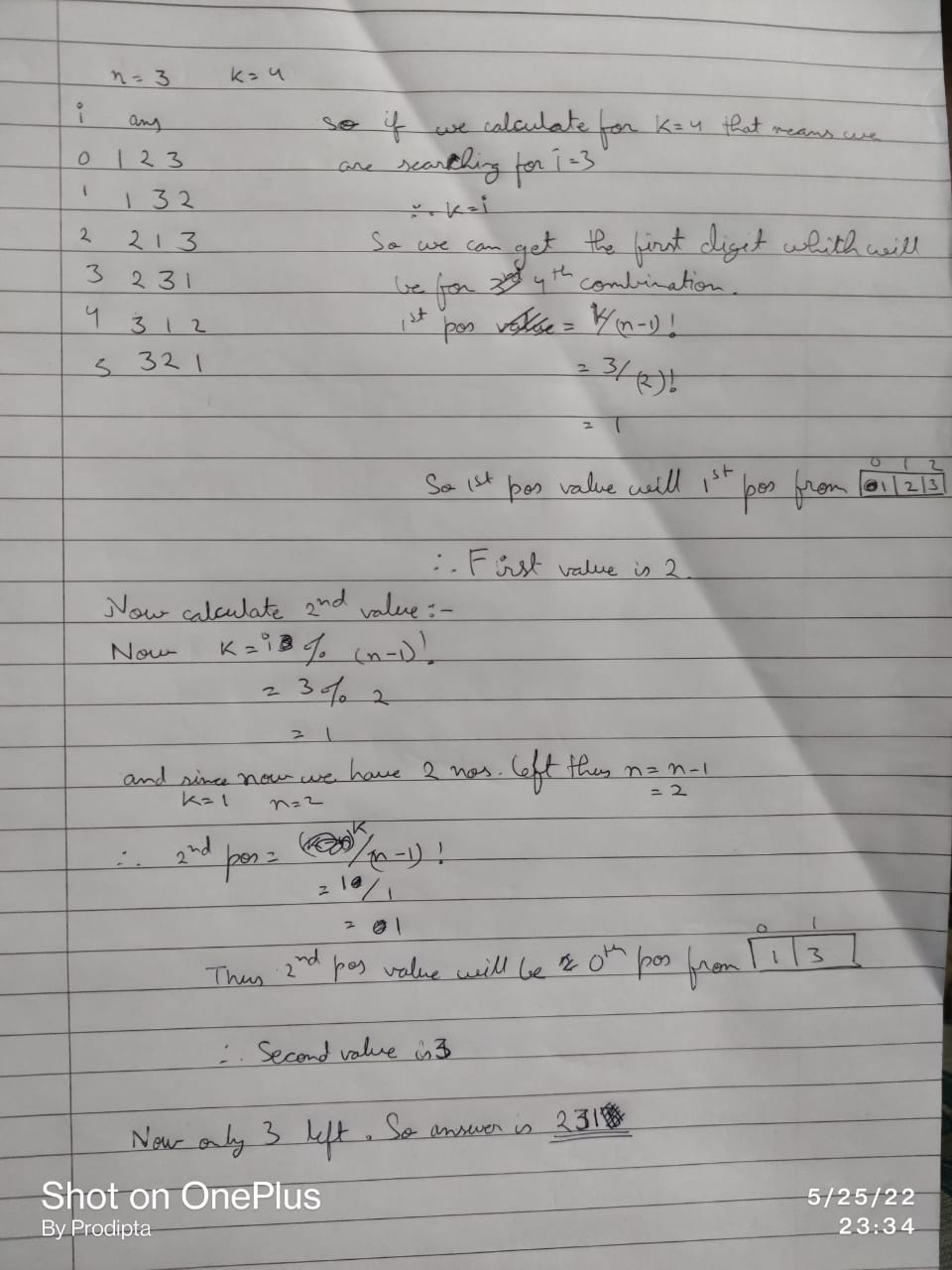
Now find max number of slope which occured for a single point.

The idea might sound to be simple but the execution isn't! As you all know there is a precision error with floating point numbers, so if we store slope as decimal values and use them as key, then there is high chance we will anomalous result.

First I tried out all combination using Backtracking, and stopped at kth solution.

But that faced TLE.

So let's now analyze the problem.



Code:

class Solution {

public String getPermutation(int n, int k) {

int[] factorials = new int[n];

factorials[0] = 1;

for (int i = 1; i < factorials.length; ++i) {

factorials[i] = i \* factorials[i - 1];

}

boolean[] vis = new boolean[n];

int remainder = k - 1;

String answer = "";

for (int i = n - 1; i >= 0; --i) {

int offset = remainder / factorials[i];

System.out.println(remainder+" "+factorials[i]);

for (int j = 0; j < vis.length; ++j) {

if (vis[j]) {

continue;

}

if (offset == 0) {

answer += j + 1;

vis[j] = true;

break;

}

--offset;

}

remainder %= factorials[i];

}

return answer;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>