CS29003 ALGORITHMS LABORATORY
ASSIGNMENT 7
Date: 5th Sep, 2019

Important Instructions

1. **List of Input Files:** input.txt
2. **Output Files to be submitted:** ROLLNO_A7_P1.c/.cpp, ROLLNO_A7_P2.c/.cpp
3. Files must be read using file handling APIs of C/C++. Reading through input redirection isn't allowed.
4. You are to **stick to the file input output formats strictly** as per the instructions.
5. Submission through **.zip files aren't allowed**.
6. Write your name and roll number at the beginning of your program.
7. Do not use any global variable unless you are explicitly instructed so.
8. Use proper indentation in your code.
9. Please follow all the guidelines. Failing to do so will cause you to lose marks.
10. **There will be part marking.**

Dynamic Programming

Problem Statement

Miracle is a pharmaceutical company which is specialized in making anti venom. Recently, their scientists have discovered a new anti venom capsule. Now, they have to test this new capsule. It is known that the capsule dissolves in venom when the venom is neutralized. Procuring venom is costly and the anti venom capsule is also very expensive. In an experiment, a drug is allowed to be tested only a certain number of times. So, Miracle needs an expert who can assist them in testing their new drug. They need to find out the maximum amount of venom in which the capsule will not dissolve. As you are an expert in algorithm, they are seeking your help.

You are given K capsules and N test tubes $(B_1, B_2, ..., B_N)$ where i^{th} test tube has i ml venom, $i = 1, ..., N$. Each capsule is identical in function i.e. the effect of same amount of venom is the same for all capsules. In each trial, you may take one capsule and put it into any test tube B_j, $1 \leqslant j \leqslant N$. If the capsule does not dissolve, it can be used again. If a capsule dissolves in x ml, $1 \leqslant x \leqslant N$ venom, it will also dissolve in a higher amount of venom. If a capsule does not dissolve in x ml venom, it will also not dissolve in a lower quantity of venom. The capsule may dissolve in 1 ml venom. Also, it is not ruled out that the capsule will not dissolve in N ml venom.

You are required to determine the minimum number of trials in the worst case using the best strategy i.e. you want to determine into which amount of venom you can drop a capsule such that it does not dissolve for a given K and N. Can you devise an algorithm to solve the problem?

Example

Assume that you have 2 capsules and 15 test tubes. Since you have only 2 capsules, you may try to solve this problem using binary search, but this is not the best strategy. In the best case you will need 5 trials. But, if the capsule dissolves into B_8, you will have to check all the remaining test tubes i.e. put the capsule into B_1, then into B_2, and so on, testing each test tube until the capsule dissolves. In the worst case number of attempts will be 8. This method will not give optimal result since you need to find the least no of trials that is guaranteed to find the maximum volume test tube where the capsule will not dissolve.

Suppose you want to restrict the number of trials to m. Then to keep this value intact as the worst case, after the first try on the m^{th} test tube, you will subsequently try $B_{m+(m-1)}, B_{m+(m-1)+(m-2)}, \ldots$. This way, if the capsule dissolves at some test tube (say B_d), you will only need to linearly check all the test tubes after the previous try (say B_p) i.e. check linearly $B_{p+1}, B_{p+2}, \ldots, B_{d-1}$. For N test tubes, you should have $\frac{m(m+1)}{2} \geqslant N$. Thus for $N = 15$, the value for m comes as 5.

- ▷ You start from B_5.
- ▷ If the capsule is dropped into B_5 and it dissolves, then you should check first, second, third and fourth test tube. So, it will take a total of 5 trials i.e. if the capsule dissolves in B_5, the trials will be: $B_5 \rightarrow B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4$.
- ▷ If the capsule is first put into B_5 and it does not dissolve, the second trial should be on the B_9, i.e. if the capsule dissolves in B_9, the trials should happen in the following sequence: $B_5 \rightarrow B_9 \rightarrow B_6 \rightarrow B_7 \rightarrow B_8$.
- ▷ Using the same reasoning, you should check B_{12}, B_{14}, and finally B_{15}.
- ▷ With this strategy you would cover all the 15 test tubes and the number of attempts would never be greater than 5, even in the worst cases.

As you can see above, the calculation for 2 capsules and 15 test tubes requires 5 trials at the worst case using this strategy. But, what if we had 3 capsules (*worst case 5 trials*), 4 capsules (*worst case 4 trials*), or more, generally K capsules. Similarly, what if you have N test tubes. You can try to calculate manually, but you will find that the calculations become perplexing. Luckily, if you observe the right way, you can identify a optimal substructure and overlapping sub-problem property for this problem and as you are taught in class, you can use the help of dynamic programming to solve this efficiently.

Part-1

Find the recurrence relation to solve the problem. Use this relation to write a ***Recursive Program*** to find minimum number of trials in the worst case. You will find that this works for smaller value of K and N, but performs miserably for larger values (try with $k = 2$ and $N = 60$, and you can see this for yourself :D – Pro. tip: do not waste your time waiting for the code to finish calculating).

Also, write the base cases and recurrence relation along with explanation in your code within comments (similar to the example shown below). Partial marks will be awarded if you can figure out the recurrence.

```
//Example code for Fibonacci Number with recurrence in comments
int Fibonacci (int n)
{
/*
Base cases :
```

```
    Fibonacci(0) = 0
    Fibonacci(1) = 1
Recurrence relation :
    Fibonacci(n) = Fibonacci(n-1)+Fibonacci(n-2)
Explanation of recurrence:
    Nth Fibonacci number is the sum of (N-1)th and (N-2)th Fibonacci numbers
 */
// Write code to compute Nth Fibonacci number
...
...
}
```

Part-2

Identify the overlapping sub-problems and optimal substructure of the problem. Write a program to solve the problem with ***Dynamic Programming*** approach.

Sample Input File

FILE: *input.txt*

```
4
2 15
2 50
4 80
10 150
```

The first line contains the number of test cases T. Each of the next T lines contains two integers K and N each.

Sample Output File

FILE: *output.txt*

```
2 15 5
2 50 10
4 80 7
10 150 8
```

Each of the lines in output file should contain the two input data followed by the output.

Make sure to read from a file named *"input.txt"*. Create a sample input file like the one shown above to test your code. At the time of evaluation, the input data might be different from the one given here.

You need not to submit *"output.txt"*, but your code should write the output in the given format in a file named *"output.txt"*