DFS LAB – ASSIGNMENT 2

MTech(CS) I year 2019–2020

Deadline: 11 October, 2019

Total: 60 marks

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs19xx-assign2-progy.c`
2. To submit a file (say `cs19xx-assign2-progy.c`), go to the directory containing the file and run the following command from your account on the server (IP address: 192.168.64.35)

   ```
   cp -p cs19xx-assign2-progy.c ~dfslab/2019/assign2/cs19xx/
   ```

   If you want to submit your files from a computer with a different IP address, run

   ```
   scp -p cs19xx-assign2-progy.c \
   mtc19xx@www.isical.ac.in:/user1/perm/pdslab/2019/assign2/cs19xx/
   ```

   (enter your password when prompted).

   To submit all `.c` and `.h` files at one go, use
   `cp -p *.c *.h ~dfslab/2019/assign2/cs19xx/` or similar.

NOTE: Unless otherwise specified, all programs should take the required input from stdin, and print the desired output to stdout.

Q1. To test the statistical significance of over-representation of successes in a sample drawn from a population, the hypergeometric p-value is calculated as the probability of randomly drawing same or more number of successes in total draws. Suppose an event is observed n number of times out of total N observations, and given the evidence that the event originally occurs e number of times out of E total cases, then the p-value is computed assuming a hypergeometric distribution as given follows.

$$p\text{-value} = \sum_{i=n}^{N} \frac{\binom{e}{i}\binom{E-e}{N-i}}{\binom{E}{N}}.$$

Note that, the cases should not exist where $(E - e) < (N - i)$. A poor p-value computed from a solution establishes the truth of not receiving the result by chance. The hypergeometric distribution is a discrete probability distribution that describes the number of successes in a sequence of n draws from a finite population without replacement.

Consider the following C program that computes the p-value of a statistical significance test based on hyergeometric distribution and reports the execution time. Write a revised program, by ONLY replacing the portion embraced within `/* START OF BLOCK */` and `/* END OF BLOCK */`, such that the revised program performs better in terms of the time taken. More credit will be given toward more efficient implementation. Note that, the following C program takes input from a separate file input.txt.

```
#include<sys/time.h>
#include<stdio.h>
#define GAP(start, end) ((end.tv_usec-start.tv_usec)+(end.tv_sec-start.tv_sec)*1000000)
void main(){
    struct timeval start_time, end_time;
    gettimeofday(&start_time, NULL);
    FILE *fptr = fopen("input.txt", "r");
    /* START OF BLOCK */
    long int n, N, e, E, e_choose_i, E_e_choose_N_i, E_choose_N;
    float p_value = 0;
    long int i, var1, var2, var3, var4; // Temporary variables
    fscanf(fptr, "%ld%ld%ld%ld", &n, &N, &e, &E);
    for(i = n; i <= N; i++){
        for(var1 = 1, var4 = 2; var4 <= e; var4++)
            var1 = var1 * var4; // Computes e!
        for(var2 = 1, var4 = 2; var4 <= i; var4++)
            var2 = var2 * var4; // Computes i!
        for(var3 = 1, var4 = 2; var4 <= e-i; var4++)
            var3 = var3 * var4; // Computes (e-i)!
        e_choose_i = var1/(var2*var3); // Computes (e choose i)
        for(var1 = 1, var4 = 2; var4 <= (E-e); var4++)
            var1 = var1 * var4; // Computes (E-e)!
        for(var2 = 1, var4 = 2; var4 <= (N-i); var4++)
            var2 = var2 * var4; // Computes (N-i)!
        for(var3 = 1, var4 = 2; var4 <= ((E-e)-(N-i)); var4++)
            var3 = var3 * var4; // Computes ((E-e)-(N-i))!
        E_e_choose_N_i = var1/(var2*var3); // Computes ((E-e) choose (N-i))
        p_value += e_choose_i*E_e_choose_N_i;
    }
    for(var1 = 1, var4 = 2; var4 <= E; var4++)
        var1 = var1 * var4; // Computes E!
    for(var2 = 1, var4 = 2; var4 <= N; var4++)
        var2 = var2 * var4; // Computes N!
    for(var3 = 1, var4 = 2; var4 <= E-N; var4++)
        var3 = var3 * var4; // Computes (E-N)!
    E_choose_N = var1/(var2*var3); // Computes (E choose N)
    p_value /= E_choose_N;
    /* END OF BLOCK */
    fclose(fptr);
    gettimeofday(&end_time, NULL);
    printf("p = %f (%ld microseconds)", p_value, (long int) GAP(start_time, end_time));
}
```

[20 marks]

Q2. Genetic Algorithms (GAs) are nature-inspired algorithms that are used for solving optimization problems. It pursues a natural selection approach that drives biological evolution process. The GA alters a population of chromosomes (denoting individual solutions) generation by generation. At each generation (derived in an iterative way), it randomly selects a pair of chromosomes (represented as strings) from the current population to be parents and uses them to produce the children for the next generation. Over successive iterations, the population evolves toward an optimal solution.

The GA employs different rules at each generation to create the next generation from the current population. Crossover is one such rule that combines two parents to form children for the next generation. The crossover depends on a `crossover probability` and a designated set of `crossover points`. In two-point crossover, one of the crossover types, characters to the right of first point in first chromosome is swapped with the characters to the right of second point in second chromosome. This results into two children, each carrying some genetic information from both parents. An illustrative example is shown below in Fig. 1.

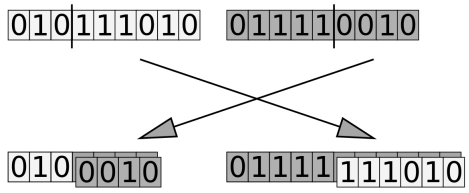

Figure 1: A two-point crossover operation.

Write a program that takes a crossover probability, a pair of parent chromosomes, and a pair of children as user inputs and return the possible pairs of two-point crossover points. Your program should be both time and space efficient.

Let the crossover points start from 0 and ends at n, the chromosome size. Note that, having a two-point crossover at points (0, 0) will simply turn the first chromosome into the second child and the second chromosome into the first child. On the other hand, having a two-point crossover at points (n, n) will simply turn the first chromosome into the first child and the second chromosome into the second child.

[20 marks]

Input Format

The first line of input is a real value (within the range [0, 1]) denoting the crossover probability. This is followed by the pairs of parent chromosomes and children, one in each line taken from the standard input.

Output Format

The output is a pair of integers representing the two crossover points in the two different parent chromosones, respectively. If the given inputs are infeasible, toward having a valid two-, it must return `INFEASIBLE`.

Sample Input 0

1
010111010
011110010
0100010
01111111010

Sample Output 0

3 5

Sample Input 1

1
110101
100001

100001110101

Sample Output 1

0 6

Sample Input 2

0
110101
100001
100001
110101

Sample Output 2

INFEASIBLE

Q3. You are given a pair of lists, say L1 and L2, each containing some positive integers. Let $P_{L1} = \prod_{a_i \in L1} a_i$ and $P_{L2} = \prod_{b_i \in L2} b_i$. Your objective is to output L1 if P_{L1} is larger, L2 otherwise. Note that, the lists may contain many numbers and their products can easily be so large that it becomes hard to manage. Write a program that can efficiently serve the said purpose.

[20 marks]

Input Format

The inputs (taken from stdin) are the elements present in the two lists L1 and L2 in succession.

Output Format

The output is either L1 or L2 denoting the list for which the product of elements is bigger. If there is a tie, output both L1 and L2.

Sample Input 0

1 2 3 4 5
1 2 3 4 5 6

Sample Output 0

L2

Sample Input 1

2 4 6 8
8 3 16

Sample Output 1

L1 L2

Sample Input 2

2 2 2 2 2 2 2 2 2
4 4 4 4

Sample Output 2

L1