Thomas Jefferson
Invitational Open in Informatics

Sample Problems (With Solutions)

Version 2.01.1

By programmers, For programmers

Preface

Welcome to the TJ IOI Sample Problems document. Here, you will find a few problems of both the theoretical and practical type. Not all categories will be covered (see the Study Materials for other categories), but we hope that these problems will give you a better idea of what to expect at the competition. Enjoy!

Feel free to contact us (`tjioiofficers@gmail.com`) if you have questions or concerns, or if you would like some more sample problems. In addition, we greatly appreciate any feedback about the difficulty level, problem types, or anything.

Programming Problems

The practical problems require programs as solutions. The format of the problems in this document will be similar to the ones on the competition. All programs will be tested on 5–10 test cases. These test cases are generally more difficult than the example cases and may be tricky (but legal), so be sure to consider cases that might mess up your programs. There are many solutions for each problem, but we have provided just one written in Java (more languages available on request). The general guideline is that your program should read the input from standard input and output to standard output, as if one were typing in the input on a console. Be sure to follow the exact input and output format specifications. During the competition, all programs will have 30 seconds to run.

Theoretical Problems

These contain several problems with a short answer format. We may decide to include additional problems without backstory, or we may decide to use similarly formatted problems during the competition. However, these problems should give a better idea of the test in terms of problem types, difficulty, and the thinking required to solve the problems.

Contents

Sample Programming Problem 1

Problem Statement: Candy Piles

William starts with three piles of candy. The first pile contains A pieces of candy, the second pile contains B pieces of candy, and the third pile contains C pieces of candy. William then buys enough candy to create the fourth pile, which contains an amount of candy equal to the first, second, and third piles combined. The fifth pile contains as much as the second, third, and fourth piles combined. William continues this pattern, with each new pile containing as many pieces of candy as the previous three piles. How many pieces of candy are in the N^{th} pile?

Input and Output Format

Input:

- Line 1: Four space-separated integers: A, B, C, and N. $0 \leq A, B, C \leq 10$, $1 \leq N \leq 30$.

Output:

- Line 1: One integer that is the number of pieces of candy in the N^{th} pile.

Examples

Input

```
1 2 3 5
```

Output

```
11
```

The fourth pile holds $1 + 2 + 3 = 6$ pieces of candy, and the fifth pile holds $2 + 3 + 6 = 11$ pieces of candy. The answer being sought is the size of the fifth pile, 11.

Input

```
5 5 5 10
```

Output

```
525
```

Sample Solution

To solve this problem, we can calculate the values of the next pile from the values of the three previous piles. The program below stores only the most recent values, but storing the values in an array works too.

```java
import java.util.*;

public class Prob1
{
    public static void main(String[] args)
    {
        Scanner console = new Scanner(System.in);
        int A = console.nextInt();
        int B = console.nextInt();
        int C = console.nextInt();
        int N = console.nextInt();

        if(N == 1)
            System.out.println(A);
        else if(N == 2)
            System.out.println(B);
        else if(N == 3)
            System.out.println(C);
        else
        {
            for(int i = 4; i <= N; i++)
            {
                int next = A + B + C;
                A = B;
                B = C;
                C = next;
            }
            System.out.println(C);
        }

        System.exit(0);
    }
}
```

Sample Programming Problem 2

Problem Statement: Choosing Cows

Farmer John wants to choose 4 cows to represent his farm at the TJ IOI. The programming level of each cow represents its ability to do well in the competition and is represented by an integer from 1 to $10,000$. Farmer John has an infinite number of cows of each programming level. However, he does not want too high of an intelligence concentration on his team or else the team will be unfair. Thus, he wants to make sure that the sum of the squares of the intelligence levels of the four cows is less than or equal to N. How many ways are there for Farmer John to choose his team? Teams are ordered and two teams differ if the programming level of a cow in one position differs from the programming level of the cow from the other team in the same position. $\{1, 4, 1, 2\}$ differs from $\{1, 2, 1, 4\}$, for example.

Input and Output Format

Input:

- Line 1: One integer: N. $4 \leq N \leq 1,000,000$.

Output:

- Line 1: One integer that is the number of ways to choose the team of cows. Watch out—this number might be too large to fit in a 32-bit integer.

Examples

Input

```
7
```

Output

```
5
```

The five possible teams are $\{1, 1, 1, 1\}$, $\{1, 1, 1, 2\}$, $\{1, 1, 2, 1\}$, $\{1, 2, 1, 1\}$, and $\{2, 1, 1, 1\}$.

Input

```
50
```

Output

```
467
```

Sample Solution

The first observation is that the programming level of any single cow is at most $\sqrt{N}$, or else the total will surely be too large. That means the maximum intelligence level we will ever have to consider is $1,000$. However, for $N = 1,000,000$, checking all combinations of 4 cows, each with up to programming level $1,000$, needs at least $1,000^4 = 1,000,000,000,000$ operations, which is far too slow. The second observation is that we do not actually need to check all the values of the fourth cow's programming level. We simply find the maximum one that works and then count for all the values below that.

```
import java.util.*;

public class Prob2
{
    public static void main(String[] args)
    {
        Scanner console = new Scanner(System.in);
        int N = console.nextInt();

        long count = 0;
        int limit = (int) Math.sqrt(N);
        for(int a = 1; a <= limit; a++)
            for(int b = 1; b <= limit; b++)
                for(int c = 1; c <= limit; c++)
                    if(a * a + b * b + c * c < N)
                        count+=(int)Math.sqrt(N - (a * a + b * b + c * c));
        System.out.println(count);

        System.exit(0);
    }
}
```

Sample Programming Problem 3

Problem Statement: Grassy Patches

James wants to plant trees on the field! However, not all patches of grass are suitable for planting. On James's field, which is a grid containing N rows of N grass patches, a grassy patch is either "good" or "bad." A bad patch is marked with a 'B' and a good patch is marked with a 'G'. Two good patches are connected if it is possible to reach one good patch from the other by traveling only up, left, right, and down through other good patches. A "good cluster" is a group of connected patches that are all good. James can only plant trees in good clusters that contain at least 3 good patches. Help James count the number of good clusters that he can plant trees in.

Input and Output Format

Input:

- Line 1: One integer: N. $1 \leq N \leq 100$.
- Line 2...$N+1$: Line $i+1$ contains N characters, all either 'B' or 'G', that describe row i of the field.

Output:

- Line 1: One integer that is the number of good clusters on the field with at least 3 good patches.

Examples

Input

```
4
GGBB
GBBB
BBBG
GBGG
```

Output

```
2
```

The upper left corner and the bottom right corner are both part of good clusters of size 3. There is also a good cluster using the bottom left good patch, but that one is not large enough to hold a tree (it has a size of 1).

Input

```
7
BGGGBBG
GGBBBGB
BGBBGBB
BBBBGBB
BGGBBBG
BGGGGBG
GBBBBBG
```

Output

```
3
```

Sample Solution

The sample below uses a recursive (depth-first search) flood fill, going to each cell and searching left, right, up, and down. During each flood fill, whenever we find a good patch that we did not already visit, we increment *size*. If this size is large enough, then we add this patch to our total count of sufficiently large good clusters.

```
import java.util.*;

public class Prob3
{
    static int N;
    static char[][] field;
    static boolean[][] visited;
    static int size;
    public static void main(String[] args)
    {
        Scanner console = new Scanner(System.in);
        N = Integer.parseInt(console.nextLine());
        field = new char[N][N];
        visited = new boolean[N][N];

        for(int i = 0; i < N; i++)
            field[i] = console.nextLine().toCharArray();

        int count = 0;
        for(int i = 0; i < N; i++)
            for(int j = 0; j < N; j++)
                if(!visited[i][j] && field[i][j] == 'G')
                {
                    size = 0;
                    flood_fill(i, j);
                    if(size >= 3)
                        count++;
                }

        System.out.println(count);

        System.exit(0);
    }

    public static void flood_fill(int row, int column)
    {
        if(row < 0 || row >= N || column < 0 || column >= N)
            return; // out of bounds
        if(visited[row][column] || field[row][column] == 'B')
            return; // no need to visit
        size++;
        visited[row][column] = true;
        flood_fill(row - 1, column); // up
        flood_fill(row + 1, column); // down
        flood_fill(row, column - 1); // left
        flood_fill(row, column + 1); // right
    }
}
```

Sample Theoretical Problem 1

Three alien races—the binsarians (who use a base-2 number system), the octonians (who use base-8), and the hexons (who use base-16)—are meeting together and wish to know what the total population amongst the three races is. Unfortunately, the three alien races each counted their populations in their own base number systms, and they need your help to find the total population.

Race	**Population**
binsarians	1011_2
octonians	201_8
hexons	$5C_{16}$

1. What is the total population in base-10?

Solution: $\boxed{232}$

In base 10, the population of the binsarians is $1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 2 + 1 = 11$.

In base 10, the population of the octonians is $2 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 = 2 \cdot 64 + 1 = 129$.

In base 10, the population of the hexons is $5 \cdot 16^1 + 12 \cdot 16^0 = 80 + 12 = 92$.

The total population in base 10 is $11 + 129 + 92 = 232$.

2. Each race would also like to know the total population in its own number system. Using the base-10 population calculated in part 1, what is the figure in base-2, base-8, and base-16?

Solution: $\boxed{11101000_2}$

$\boxed{350_8}$

$\boxed{E8_{16}}$

There are several ways to solve this problem. One way is to convert the base 10 answer into each of the bases 2, 8, and 16, and another way is to convert the original populations of each race into other bases, and add within each number system. A trick is to first convert things into base 16, and then convert the base 16 result into base 8, and then base 2.

To find 232_{10} in base 16, we note that the highest power of 16 that is less than 232 is 16^1 ($16^2 = 256 > 232$). Thus, our base 16 answer will only have two digits, for the two powers of 16 (0 and 1) that fit under 232. 16^1 goes into 232 14 times ($14 \cdot 16 = 224$). Thus, the first digit of our base 16 answer is E, or 14. There is 8 left over, so 8 is the last digit. $232_{10} = E8_{16}$. To convert into base 2, we note that each digit in base 16 maps to four binary digits. $E_{16} = 1110_2$ and $8_{16} = 1000_2$. Thus, the answer in base 2 is 11101000_2. We can group binary digits into groups of 3 to convert to base 8, so we have $000_2 = 0_8$, $101_2 = 5_8$, $11_2 = 3_8$ for an answer of 350_8.

3. There is actually a fourth alien race, the ternops. They count in base 3. The population of the ternops is equal to the total population of the binsarians and the octonians. Find the population of the ternops in base 3.

Solution: $\boxed{12012_3}$

The population of the ternops, in base 10, is $11 + 129 = 140$ (calculated from earlier). In base 3, 140 is 12012_3. We can calculate this by noting that the largest power of 3 under 140 is $3^4 = 81$, which fits once. With the $140 - 81 = 59$ remaining, the next power of 3, $3^3 = 27$, goes in twice. The next digit is 2. There is 5 remaining, which is $1 * 3^1 + 2 * 3^0 = 12_3$. The final answer is 12012_3.

Sample Theoretical Problem 2

The nefarious wizard Sreenath has devised a set of problems to ensnare the valiant knight Alex. Sreenath's twisted mind has produced several convoluted recursive riddles. The riddles will sap Alex's resolve if he cannot solve them. Help Alex defeat his foe.

1. Sreenath demands from Alex the value of $f(20)$ where

$$f(x) = \begin{cases} f\left(\left\lfloor \frac{x}{2} \right\rfloor\right) & x = 39 \\ f(x+1) & 20 \leq x < 39 \\ 1 + f(x-1) & 0 < x < 20 \\ 0 & x \leq 0 \end{cases}$$

where $\lfloor x \rfloor$ is the greatest integer less than x.

Solution: $\boxed{19}$

The function jumps up to $x = 39$ through the first case. From there we go to $x = \left\lfloor \frac{39}{2} \right\rfloor = 19$. After that, the function just adds 1 for each positive integer until we reach $x = 0$. This means $f(20) = 19$.

Note that $f(x)$ is defined only for $x < 40$.

2. One of Sreenath's minions, a Srnth, is standing 7 units away from Alex. Srnths can hop across distances of 1 or 2 units. If the Srnth moves always straight toward Alex, how many sequences of 1-hops and 2-hops can it take to reach Alex?

Solution: $\boxed{21}$

We can compute the answer using recursion. Suppose the answer for a distance of n is F_n; then $F_1 = 1$, $F_2 = 2$, and $F_n = F_{n-1} + F_{n-2}$, as a sequence can end in either in a 1-hop or 2-hop, which leave distances of $n - 1$ and $n - 2$ respectively. This is the definition of the Fibonacci numbers. Listing out the first seven terms using the recurrence: $1, 2, 3, 5, 8, 13, 21$. So our answer is $F_7 = 21$.

It is often natural to consider the "empty sequence", which makes $F_0 = 1$ here. When defining a sequence, you need to be careful about its initial conditions (such as the value for F_0). If we took $F_0 = 0$, $F_1 = 1$, we would get $F_2 = 1$, which is false.

Sample Theoretical Problem 3

Albert, Alec, and Alex are engineers. In particular, they are electrical engineers. Recently, they have been hired by Farmer Arjun to fix the lighting system in the farmer's house. Farmer Arjun has not been having a great year so far, so he has only one light bulb, which is controlled by switches conveniently labeled "A" through "F". The switches form a network that turns the light bulb on if the networks evalutes to true, and off if false. Help Albert, Alec, and Alex find which states turn the light bulb on, so that they can avoid states that result in electrocution.

For each of the following problems, assume that the boolean algebra expression given reflects the network of switches. Please leave your answers as ordered n-tuples. You may use $*$ to represent a value that can be either 0 or 1.

1. $A + B$

Solutions: $\boxed{(1,1),(1,0),(0,1)}$

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

2. ABC

Solutions: $(1,1,1)$
If any of A, B, or C are 0, then the expression ABC evaluates to 0. Thus, $(1,1,1)$ is the only possible solution.

3. $\overline{A \oplus B}$

Solutions: $(1,1),(0,0)$ (Note: this combination of NOT and XOR checks whether two variables are equivalent.)

A	B	$A \oplus B$	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

4. $A(B + AC + \overline{A})$

Solutions: $\boxed{(1,0,1,*,*),(1,1,0,*,*),(1,1,1,*,*)}$

A	B	C	AC	$\overline{A}$	$B+AC+\overline{A}$	$A(B+AC+\overline{A})$
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	1	1	0	1	1

5. $\overline{D}E + AB + \overline{D} + 0 + ABC$

Solutions: $\boxed{(*,*,*,0,*),(1,1,*,1,*)}$
This is obtained by simplifying and using a truth table.

$$\begin{aligned}&\overline{D}E + AB + \overline{D} + 0 + ABC\\ &= \overline{D}(E+1) + AB(1+C)\\ &= \overline{D} + AB\end{aligned}$$

A	B	D	$\overline{D}$	AB	$\overline{D}+AB$
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

6. $((A+B)A) + \overline{(BA)+B}$

Solutions: $\boxed{(1,0),(1,1)}$

A	B	$A+B$	$A*B$	$(A+B)*A$	$(B*A)+B$	$\overline{(B*A)+B}$	$((A+B)*A)$ $+\overline{(B*A)+B}$
0	0	0	0	0	0	1	0
1	0	1	0	1	0	1	1
0	1	1	0	0	1	0	0
1	1	1	1	1	1	0	1

Sample Theoretical Problem 4

Alex is competing in the Thomas Jefferson Invitational Open in Informatics and is on the practical section of the contest. He wants to make sure that his programs run within the 30-second runtime limit. For each of the code segments below, determine the runtime efficiency in big-O notation. Assume that all necessary header files have been included, all variables have been declared, and the program compiles with no error.

1.

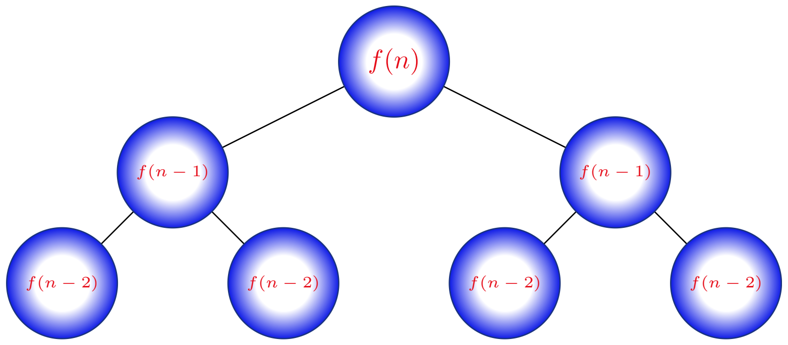
```
for(int i=0;i<N;i++){
  int a=0,b=1;
  while(a+b<i){
    a+=b;
  }
}
```

Solution: The outer for-loop runs N times for a contribution of N, and the inner while-loop runs i times. Since i is, at worst case, $N-1$, the inner while-loop contributes N as well. Nothing else in this code segment contributes significantly to the runtime, thus the big-O efficiency for this code segment is $O(N \cdot N) = \boxed{O(N^2)}$.

2.

```
int f(int n){
  if(n==0){
    return 1;
  }
  return f(n-1)+f(n-1);
}
int main(){
  ⋮ // has efficiency of O(1)
  f(N);
}
```

Solution: For each call of `f()`, 2 more calls to `f()` are made, to a depth of N. Thus, the recursion of the function is $O(2^N)$. Since the function itself runs in constant time, as does the unwritten section in the main function, the overall efficiency of this code segment is $\boxed{O(2^N)}$.



3. For $1 \leq M \leq 10000000$ and $1 \leq N \leq 25000000$

```
for(i=0;i<M;i++){
  a*=a;
}
for(i=0;i<N;i++){
  b+=b;
}
```

Solution: This is a tricky one. Usually, big-O notation contains only 1 variable, but here, because both M and N can grow to be very large, and because M and N are both the same order, they both have an effect on the runtime of this code segment.

Thus, the analysis is as follow: the first loop has an efficiency of $O(M)$, since at worst case, the loop will run M times, multiplied by the constant efficiency of the content of the loop. The same analysis can be made of the second loop, except with N instead of M. Thus, the overall efficiency for the entire code segment is $\boxed{O(M+N)}$.

4.

```
for(i=0;i<V;i++){
  for(j=0;j<V;j++){
    for(k=0;k<V;k++){
      path[j][k]=min(path[j][k],path[j][i]+path[i][k]);
    }
  }
}
```

Solution: This is simply three nested for-loops, each costing $O(V)$, so the overall runtime is $O(V \cdot V \cdot V) = \boxed{O(V^3)}$

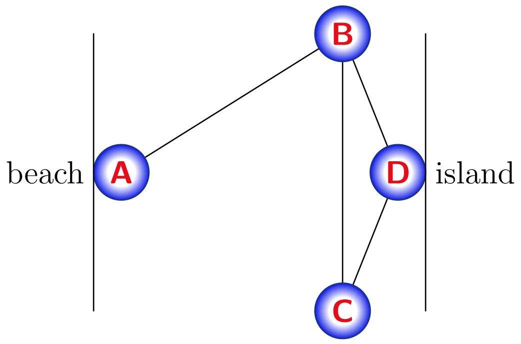
Sample Theoretical Problem 5

Alex the Aardvark and Arjun the Armadillo are questing for treasure when they meet Saketh the Sponge at the shoreline. Being friends, Saketh agrees to help Alex and Arjun cross the great Guralese Ocean. However, because Saketh likes puzzles (and difficult math problems) very much, he has placed several buoys joined by some logs to help Alex and Arjun cross the ocean. Of course, Alex and Arjun can't swim, so they must walk on the logs and buoys.

Given each of Saketh's descriptions of the layout of the logs and buoys, help Alex and Arjun draw out a map. Note that Alex and Arjun start on the beach and must reach the island. Note that there will always be at least one buoy connected to the beach and one to the island, which do not require the use of logs.

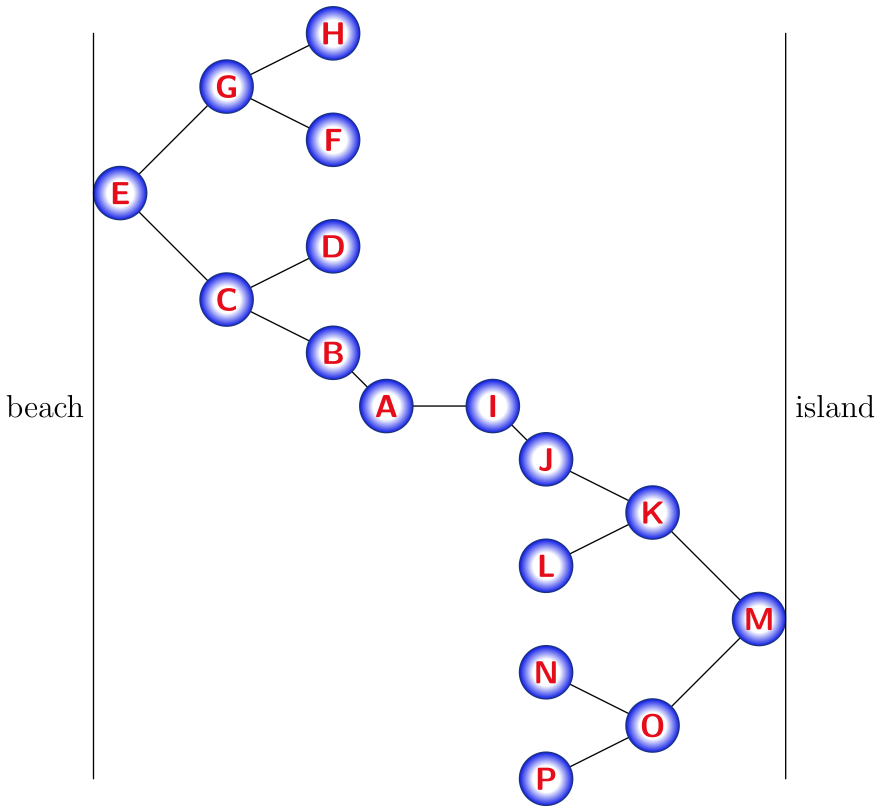
1. Saketh says: "I've placed 4 buoys and 4 logs. buoy A is right next to the beach, and is connected to buoy B and no other buoy. buoy C and D are both connected to buoy B, and buoy D is next to the island. Furthermore, C and D are each connected to another buoy, which is different between them."

Solution: (Note that the beach and island are not necessary, but should still be diagrammed.)



2. Seeing that Alex and Arjun quickly figured out the map, Saketh decided to play even more games, telling them, "Sorry, guys, I was kidding. There are way more buoys and logs - 10 buoys and 15 logs total, to be exact. In fact, the 16-buoy, 15-log structure resembles two complete binary trees - one rooted near the beach and one rooted near the island. Also, only the 'bottomost' buoys in both trees are connected by a single log. Furthermore, both trees are balanced and complete, with the tree rooted near the beach having buoys A-H and the one near the island having buoys I-P."

Solution:


H
G
F
E
D
C
B
beach
A
I
island
J
K
L
M
N
O
P