Командная олимпиада по программированию
Витус Беринг - 2018

Задача A. Поклонник Фибоначчи

Ограничение по времени: 1 с
Ограничение по памяти: 64 Мб
Максимальный балл: 50

Лёша очень любит математику. Его кумир итальянский математик Леонардо Пизанский по прозвищу Фибоначчи, придумавший известную числовую последовательность. Лёша мечтает стать похожим на своего кумира и на досуге изобретает новые числовые последовательности. В этот раз он придумал последовательность

1 2 3 4 2 3 4 5 3 4 5 6 4 5 ...

После этого Лёша задался вопросом, на каком месте в ней впервые встретится число k? Напишите программу, которая ответит на его вопрос.

Входные данные. Единственная строка входного файла INPUT.txt искомое число k ($1 \le k \le 10^6$).

Выходные данные. Выходной файл OUTPUT.txt содержит единственное число, позицию первого вхождения числа k в последовательность.

INPUT.txt	OUTPUT.txt
2	2
5	8
10	28

Решение

Начиная со 2го элемента, придуманная Лёшей последовательность представляет собой последовательность четвёрок следующего вида $(i, i+1, i+2, i)$ при $i \ge 2$, следовательно, число $i+2$ при $i \ge 4$ в первый раз встречаются в четверке, соответствующей числу i, а числа 1,2 и 3 стоят на 1й, 2й и 3й позициях соответственно. Т.о. для числа k позицию первого вхождения можно определить по формулам

$$pos(k) = k \text{ при } k \le 3,$$

$$pos(k) = (k-3) \cdot 4 \text{ при } k \ge 4.$$

Пример работающей программы на C++

```
#include <stdio.h>
#include <iostream>

using namespace std;
int main(int argc, char **argv)
{
    cout << "Inter the number: ";
    int k;
    cin >> k;
    int pos = (k<=3)? k : (k-3)*4;
    cout << "The position " << k << " is " << pos << endl;
        return 0;
}
```

Задача B. Новые правила

Ограничение по времени: 1 с
Ограничение по памяти: 64 Мб
Максимальный балл: 50

Как известно, сортировка строк сильно отличается от сортировки чисел. Лёша всегда хотел исправить это несоответствие и решил ввести новые правила для сортировки чисел. Теперь числа сортируются так же, как и строки, в лексикографическом порядке (сначала по первой цифре, потом по второй и т.д.). Однако с непривычки Лёше трудно ответить на какой позиции в списке чисел будет стоять заданное число. Напишите программу, которая по заданному числу n определяет его позицию в списке чисел от 1 до a.

Входные данные. Единственная строка входного файла INPUT.txt содержит два числа: максимальное a число, содержащиеся в списке чисел ($1 \le a \le 10^9$), искомое число n ($1 \le n \le a$).

Выходные данные. Выходной файл OUTPUT.txt содержит единственное число, позицию числа n в списке чисел от 1 до a, отсортированном в лексикографическом порядке.

INPUT.txt	OUTPUT.txt
10 10	2
30 21	14

Решение

Пусть имеется список чисел от 1 до 273, если отсортировать его в лексикографическом порядке, то получим

1
10
100 . . . 109
11
110 . . . 119
. . .
190 . . . 199
2
20
200. . . 209
. . .
23
. . .
3
30 . . . 39
4
и.т.д.

Допустим нужно определить позицию числа 23. Перед ним в списке находятся все однозначные числа от 1 до 2, все двузначные от 10 до 22 и трехзначные числа от 100 до 229. Следовательно, позиция 23 равна $2 + 13 + 130 + 1 = 146$.

Позиция числа 34 равна $3 + 24 + 174 + 1 = 202$. Т.о. положение для определения позиции числа n нужно подсчитать количество располагающихся выше чисел.

$$pos(n) = \sum_{i=1}^{D_a} K_i$$

Если $i = D_a$, то $K_{D_a} = min(a, n \cdot 10^{D_a - D_n} - 1) - 10^{D_a - 1} + 1$.

Если $i > D_n$, то $K_i = (n \cdot 10^{i - D_n} - 1) - 10^{i-1} + 1$.

Если $i \le D_n$, то $K_i = n \div 10^{D_n - i} - 10^{i-1} + 1$.

Пример работающей программы на C++

```
#include <stdio.h>
#include <iostream>
#include <cmath>

using namespace std;
int Func(int num){
    int ans = 0;
    while (num>0) {
        num/=10;
        ans++; //digit count
    }
    return ans;
}

int main(int argc, char **argv)
{
    int a,n;
    cout << "Inter the list size: "; cin >> a;
    cout << "Inter the number: "; cin >> n;

    int Da = Func(a);
    int Dn = Func(n);

    int ch, k, pos=0;

    for (int i = Da; i >= 1; i--){
        if (i > Dn)
            ch = n * pow(10,i-Dn)-1;
        else if (i == Dn)
            ch = n;
        else
            ch /= 10;
        if (i == Da)
            ch = min(ch,a);
        pos += ch - pow(10,i-1) + 1;
    }

    cout << "The position of " << n << " is " << pos << endl;

        return 0;
}
```

Задача C. Жажда

Ограничение по времени: 5 с
Ограничение по памяти: 64 Мб
Максимальный балл: 100

В Лёшиной школе затеяли ремонт. И если раньше было достаточно кулера в одном из кабинетов, поскольку из любого кабинета можно было пройти в любой другой, то теперь некоторые проходы перекрыты, при этом по открытым проходам разрешено движение только в одну сторону. Лёша задумался, какое минимальное количество кулеров для воды надо расставить по кабинетам, чтобы для любого кабинета существовал кулер, из которого ученики могут принести воду, двигаясь по открытым проходам. Помогите Лёше, напишите программу, решающую поставленную задачу.

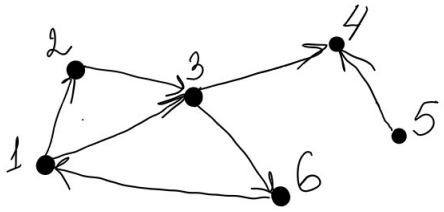
Входные данные. В первой строке входного файла INPUT.txt указано количество кабинетов N ($N < 2000$) и количество открытых проходов M ($M < 1001001$). Затем идёт M строчек, в каждой из которых дано описание прохода — два числа a и b ($1 \leq a \leq N, 1 \leq b \leq N$), обозначающих номера кабинетов, которые данный проход соединяет.

Выходные данные. Выходной файл OUTPUT.txt содержит единственное число, минимальное необходимое количество кулеров.

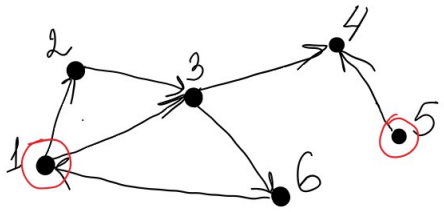
INPUT.txt	OUTPUT.txt
3 3 1 2 2 3 3 1	1
6 7 1 2 2 3 3 6 6 1 3 4 5 4 1 3	2

Решение

Данную задачу можно представить в виде ориентированного графа. Вершины обозначают кабинеты, а дуги — графы. Рассмотрим второй пример.



По графу видно, что необходимо минимум два кулера, в пятом и первом (может быть несколько вариантов) кабинетах.



Для определения количества кулеров, можно воспользоваться следующим алгоритмом. Для каждого кабинета будем отмечать из какого кабинета можно принести воду и сохранять полученный номер в массиве. До считывания информации о первом проходе предполагаем, что в кабинет можно доставить воду только из этого же кабинета.

1	2	3	4	5	6
1	2	3	4	5	6

Считываем информацию о проходе из 1 в 2.

1	2	3	4	5	6
1	1	3	4	5	6

Считываем информацию о проходе из 2 в 3. Но так как в 2 можно попасть из 1, то и в 3 можно попасть из 1. Аналогично с проходом из 3 в 6.

1	2	3	4	5	6
1	1	1	4	5	1

Проход из 6 в 1 порождает цикл, поскольку в 6 можно попасть из 1, поэтому ничего не меняем. Проход из 3 в 4.

1	2	3	4	5	6
1	1	1	1	5	1

Наконец, проход из 5 в 4.

1	2	3	4	5	6
1	1	1	5	5	1

Итого имеем два кабинета, в которых можно расположить кулеры.

Пример работающей программы на C++

```
#include <stdio.h>
#include <iostream>
#include <set>

using namespace std;
int main(int argc, char **argv)
{
    int N,K;
    cout << "Inter the cabinets count: "; cin >> N;
    cout << "Inter the ways count: "; cin >> K;

    int* comps = new int [N];
    for (int i=0; i<N; i++)
        comps[i] = i;

    for (int i=0; i<K; i++){
        int from, to;
        cout << i+1 << " way: ";
        cin >> from >> to;

        if (comps[from-1]!=comps[to-1]) {
            comps[from-1] = comps[to-1];
            for (int j=0; j<N; j++)
                if (comps[j] == from-1)
                    comps[j] = comps[from-1];
        }
    }

    set<int> n;
    for (int j=0; j<N; j++)
        n.insert(comps[j]);

    cout << "The number of water coolers is " << n.size() << endl;
```

```
    return 0;
}
```

Задача D. Головоломка

Ограничение по времени: 2 с
Ограничение по памяти: 64 Мб
Максимальный балл: 100

Лёша — любитель различных головоломок, за решением которых он коротает свое свободное время. Его любимая головоломка «8 Puzzle». Это игра на основе 8 скользящих блоков, расположенных в поле 3×3 ячейки. У каждого блока есть свой порядковый номер, за один ход блок разрешено перемещать на один из свободных (не занятых другими блоками) соседних ячеек. Цель игры - из заданной начальной позиции получить конечную (см. рисунок). Напишите для Лёши программу, вычисляющую минимальное число ходов, необходимое для получения конечной позиции из исходной.

1	2	3
4	5	6
7	8	

Конечная позиция

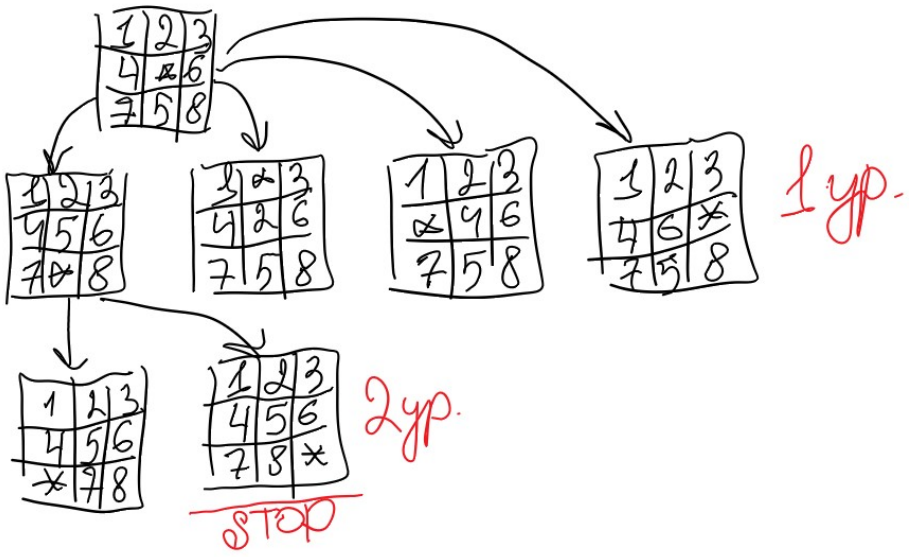
Входные данные. Во входном файле INPUT.txt подается описание начальной позиции в виде строки, в которой перечислены номера блоков, разделенные пробелами в квадрате слева направо сверху вниз. Вместо свободной ячейки напечатана *.

Выходные данные. Первая строка выходного файла OUTPUT.txt содержит YES или NO в зависимости от того, можно ли получить конечную позицию или нет. Если YES, то вторая строка содержит единственное число, минимальное количество ходов, за которое из начальной позиции можно получить конечную.

INPUT.txt	OUTPUT.txt
1 2 3 4 5 6 7 8 *	YES 0
* 1 2 3 4 5 6 7 8	YES 22

Решение

Для решения данной задачи необходимо строить дерево возможных ходов и осуществлять поиск в ширину, для каждого узла дерева нужно хранить номер уровня. Номер уровня равен количеству ходов. Если конечная позиция, так и не была найдена, то вывести на экран NO. Например, рассмотрим начальную позицию «1 2 3 4 * 6 7 5 8».



Конечную позицию можно получить за два хода.

Пример работающей программы на C++

```
#include <stdio.h>
#include <iostream>
#include <queue>
#include <set>

using namespace std;
struct node{
    string pole;
    int step_count;
};

string step(string str,int stepsize){
        int pos = str.find("*");

        if ( !(pos%3 == 0 && stepsize == -1) &&
             !(pos%3 == 2 && stepsize == 1) &&
             !(pos <= 2 && stepsize == -3) &&
             !(pos >= 6 && stepsize == 3)                ) {
                str[pos] = str[pos + stepsize];
                str[pos + stepsize] = '*';
        }
        return str;
}
int main(int argc, char **argv)
{
        queue<node> q;
        set<string> s;

        string stop = "12345678*";
    node start;
    start.step_count = 0;
    cout << "Inter start position: "; cin >> start.pole;

        q.push(start);
        s.insert(start.pole);
```

```
        bool flag = 1;

        while (!q.empty() && flag){
                start = q.front(); q.pop();
        if (start.pole == stop) {
            cout << "YES\n" << start.step_count << endl;
            flag = 0;
            break;
                }

                for (int sign=-1; sign <= 1; sign+=2)
            for(int size = 1; size <= 3; size+=2) {
                node newnode;
                newnode.pole = step(start.pole,sign*size);
                newnode.step_count = start.step_count + 1;
                if (s.find(newnode.pole) == s.end()) {
                                        q.push(newnode);
                                        s.insert(newnode.pole);
                                }
                        }
        }
        if (flag)
                cout << "NO\n";
    return 0;
}
```

Задача E. Пифагоровы тройки

Ограничение по времени: 1 с
Ограничение по памяти: 64 Мб
Максимальный балл: 100

Пифагорова тройка — это комбинация из трёх натуральных чисел, удовлетворяющих соотношению $a^2 + b^2 = c^2$. Лёша захотел составить табличку троек чисел, образующих пифагоровы тройки и отсортированных по возрастанию (по c, a, b). Для этого Лёше необходимо написать программу, вычисляющую по введенному номеру N числа a, b, c, образующие пифагорову тройку. Тройки a, b, c и b, a, c считать различными. К сожалению, Лёша не программист и попросил вас о помощи.

Входные данные. Входной файл INPUT.txt содержит целое число N ($0 < N < 10^6$).
Выходные данные. Выходной файл OUTPUT.txt содержит три целых числа a, b, c, разделённых пробелами.

INPUT.txt	OUTPUT.txt
1	3 4 5
3	6 8 10

Решение

Формула Евклида является основным средством построения пифагоровых троек. Согласно ей для любой пары натуральных чисел m и n ($m > n \geq 1$) целые числа:

$$a = m^2 - n^2,\ b = 2mn,\ c = m^2 + n^2$$

образуют пифагорову тройку. Минимально возможная пара $m = 2, n = 1$ порождает две пифагоровы тройки $(3, 4, 5)$ и $(4, 3, 5)$. Следующие пары $m = 3, n = 1$; $m = 3, n = 2$; $m = 4, n = 1$ и т.д. Т.е. n

пробегает значения от 1 до $m-1$ ($1 \le n \le m-1$), а как только $n = m-1$ значение m увеличивается на 1.

Для определения N-й пифагоровой тройки, необходимо определить $\lfloor N/2 \rfloor$-ю пару m и n, если N - нечётное, по получаем пифагорову тройку с $a < b$, иначе с $a > b$.

Пример работающей программы на C++

```
#include <stdio.h>
#include <iostream>
#include <cmath>

using namespace std;
int main(int argc, char **argv)
{
    int N;
    cout << "Inter the number of Pythagorean triplet: "; cin >> N;

    int i=1;
    int m=2;
    int n=1;

    int N_2 = ceil(N*1.0 / 2);

    while (i < N_2) {
        if (n == m-1) {
            n = 1;
            m++;
        } else
            n++;
        i++;
    }

    int a = m*m - n*n;
    int b = 2*m*n;
    int c = m*m + n*n;

    if (N%2)
        cout << min(a,b) << " " << max(a,b) << " " << c << endl;
    else
        cout << max(a,b) << " " << min(a,b) << " " << c << endl;

    return 0;
}
```

Задача F. Антифакториал

Ограничение по времени: 2 с
Ограничение по памяти: 64 Мб
Максимальный балл: 150

Изучая математику, Лёша отметил, что у каждой функции есть обратная, например, для сложения — вычитание, для умножения — деление, для возведения в степень — логарифм. Лёша задумался, почему же для факториала нет обратной функции. Помогите Лёше исправить несправедливость, напишите программу, для заданного натурального числа n определяющую факториалом какого числа оно является.

Входные данные. В единственной строке входного файла INPUT.txt указано значение факториала n ($n > 1$) (длина не превосходит 255 цифр).

Выходные данные. Выходной файл OUTPUT.txt содержит единственное число k, факториал которого равен n.

INPUT.txt	OUTPUT.txt
6	3
120	5

Решение

Для определения факториала необходимо последовательно вычислять факториалы натуральных чисел, начиная с 2, до тех пор, пока результат не совпадет с входным значением. Однако согласно условию, длина входного значения может достигать 255 значащих цифр, соответственно нельзя использовать числовые типы данных и все операции должны выполняться над числами, записанными в строки. Т.е. имеют место символьные вычисления (см. предлагаемое решение).

Пример работающей программы на C++

```
string delzeros(string s){
    int ind = s.find_first_not_of('0');
    if (ind!=string::npos)
        s=s.substr(ind,s.length()-ind);
    return s;
}

string sum(string s1, string s2){
    int k1 = s1.length();
    int k2 = s2.length();

    string part;
    if (k1>k2)  {
        s2 = string(k1-k2+1,'0')+s2;
        s1 = "0" + s1;
        part=s1;
    } else {
        s1 = string(k2-k1+1,'0')+s1;
        s2 = "0" + s2;
        part=s2;
    }
    int p = 0;
    for (int i=max(k1,k2); i>=abs(k1-k2); i--) {
        int sum = (s1[i]-'0')+(s2[i]-'0')+p;
        part[i] = '0'+sum % 10;
        p = sum / 10;
    }
    return delzeros(part);
}
string mult(const string& s1, const string& s2){
    int k1 = s1.length();
    int k2 = s2.length();

    string mul="0";
    for (int i=k2-1; i>=0; i--) {
        int p = 0;
        string part(k1+k2,'0');
        for (int j=k1-1; j>=0; j--) {
```

```
            int prod = (s1[j]-'0')*(s2[i]-'0')+p;
            p = prod / 10;
            part[i+j+1]='0' + prod % 10;
        }
        part[i] = '0' + p;
        mul = sum(mul,part);
    }
    return delzeros(mul);
}
int main(int argc, char **argv)
{
    string fact;
    cout << "Inter the factorial string: "; cin >> fact;

    int count5 = 0;
    for (int i=fact.length()-1; ;i--)
        if (fact[i] == '0')
            count5++;
        else
            break;

    string n = "1";
    string fact_test = "1";
    int k = 0;
    while (fact_test.compare(fact)){
       n=sum(n,"1");
       fact_test = mult(fact_test,n);
    }
    cout << "The anti-factorial is " << n << endl;
    return 0;
}
```