Московская олимпиада школьников по информатике 2025-2026 8 класс (первый отбор)

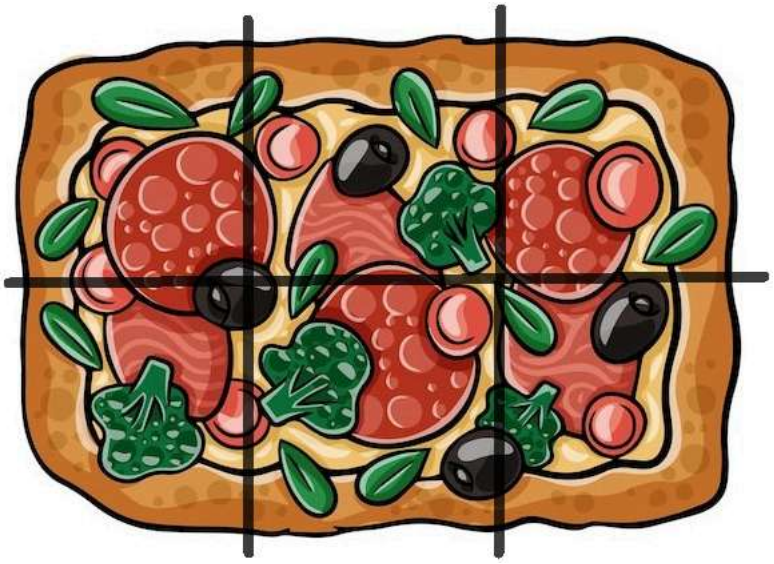
19 дек 2025 г., 10:00 — 28 дек 2025 г., 23:59

№ 1

100 баллов

Римская пицца

Дима и Лёша заказали огромную прямоугольную римскую пиццу. Дима ужасно проголодался и хочет поскорее приступить к еде. Единственный способ для него поскорее добраться до своей порции и начать радоваться жизни — это разрезать пиццу на как можно большее количество частей, используя доступное количество разрезов. Дима может делать только прямые горизонтальные и вертикальные разрезы, никак не перемещая образовавшиеся куски.



Три разреза, дающие шесть частей.

Ваша задача, зная количество разрезов N, которое Дима может сделать, найти максимальное количество частей, на которое можно порезать пиццу.

Входные данные

Первая строка входных данных содержит единственное число N ($1 \le N \le 100$) — количество разрезов, которое может сделать Дима.

Выходные данные

Выведите наибольшее количество частей, на которое Дима может разрезать римскую пиццу.

Система оценивания

Все тесты в этой задаче оцениваются независимо. Вы получите баллы за каждый пройденный тест.

Примеры

3	
6	

Ограничения

Время выполнения: 1 секунда
Память: 256 MB

Код C++

```cpp
#include <bits/stdc++.h>

using namespace std;


int main() {
    int n;
    cin >> n;
    cout << (n/2 + 1) * ((n+1)/2+1);

}

```

№ 2

100 баллов

Хижина в лесу

Коля решил стать лесником. Лес представляет собой квадрат 10×10 клеток. В лесу отмечены клетки, в которых растут деревья. Коля хочет построить хижину. Требуется определить максимальную длину хижины, которая может поместиться в свободных от деревьев клетках. Хижина представляет собой прямоугольник ширины 1 и располагается горизонтально или вертикально. (Гарантируется, что на поле есть хотя бы одна свободная клетка.)

Входные данные

Вводятся 10 строк по 10 чисел в каждой, числа разделены пробелами. Число 1 означает, что в соответствующей клетке есть дерево, число 0 — что клетка свободна.

Выходные данные

Требуется вывести одно число от 1 до 10 — максимальную возможную длину хижины.

Система оценивания

Все тесты в этой задаче оцениваются независимо. Вы получите баллы за каждый пройденный тест.

Примеры

```
0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0
```

```
10
```

```
1 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 1
```

```
8
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

```cpp

#include <bits/stdc++.h>
using namespace std;

int main()
{
    int a[10][10] = {0};
    for(int i=0; i<10; ++i)
        for(int j=0; j<10; ++j)
            cin>>a[i][j];
    for(int k=10; k>=1; --k){
        bool f = 0;
        for(int i=0; i<10; ++i)
            for(int j=0; j<10; ++j) {
                if (i + k <= 10) {
                    bool f1 = 1;
                    for (int ii = i; ii < i + k; ++ii)
                        if (a[ii][j])
                            f1 = 0;
                    f |= f1;
                }

                if (j + k <= 10) {
                    bool f2 = 1;
                    for (int jj = j; jj < j + k; ++jj)
                        if (a[i][jj])
                            f2 = 0;
                    f |= f2;
                }
            }
        if(f){
            cout<<k;
            return 0;
        }
    }
    return 0;
}
```

№ 3

100 баллов

Полтора кружка

Саша настолько сильно хочет учиться программировать, что записался сразу в два кружка. В каждом кружке учащимся задано N задач, причём Саша уже обучался в третьем кружке и имеет N решений задач, которые уже использовал для сдачи таких же задач в двух новых кружках.

Из-за нехватки времени Саша может решить не более K задач, а его преподаватели согласны совершенно безвозмездно подарить по одной печеньке Саше за каждую решённую задачу из своего кружка. Саша быстро сообразил, что ему выгодно в первую очередь решать задачи, заданные в обоих кружках. В таком случае он получает две печеньки за одну решённую задачу. Подскажите Саше, какое максимальное количество печенек он может получить.

Обратите внимание, что за старые решённые задачи Саша печеньки не получает, так как отправил их на первой же минуте кружков, и преподаватели не считают эти решения полезными.

Входные данные

В первой строке записано два целых числа N и K ($1 \le N \le 100, 1 \le K \le 100$). Во второй строке записаны N целых чисел - номера задач, решённых Сашей ранее. Третья строка содержит номера задач, заданных на первом кружке в том же формате, а четвёртая - номера задач, заданных на втором кружке. Каждая из этих трёх строк содержит последовательность различных целых чисел от 1 до 1000.

Выходные данные

Выведите единственное число — количество печенек, которое получит Саша, действуя оптимальным способом.

Система оценивания

Все тесты в этой задаче оцениваются независимо. Вы получите баллы за каждый пройденный тест.

Примеры

```
2 1
103 107
103 108
107 108
```

```
2
```

```
2 100
13 17
15 18
18 17
```

```
3
```

```
3 2
2 3 1
5 3 4
7 5 2
```

```
3
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include <bits/stdc++.h>

using namespace std;


int main() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n), b(n), c(n), cnt(1001, 0);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    for (int i = 0; i < n; ++i) {
        cin >> b[i];
        cnt[b[i]]++;
    }
    for (int i = 0; i < n; ++i) {
        cin >> c[i];
        cnt[c[i]]++;
    }
    for (int i = 0; i < n; ++i) {
        cnt[a[i]] = 0;
    }
    std::sort(cnt.rbegin(), cnt.rend());
    int ans = 0;
    for (int i = 0; i < k; ++i)
        ans += cnt[i];
    cout << ans;
}

```

№ 4

100 баллов

Остров Буян

На острове Буяне каждый житель владеет несколькими языками. Любые два жителя A и B могут вести между собой беседу только в двух случаях: если они оба владеют каким-нибудь общим для них языком, или с помощью нескольких жителей-переводчиков $p_1, p_2, \ldots, p_k$. Последнее означает, что любую фразу жителя A переводчик p_1 сможет перевести так, что его поймет p_2, житель p_2 сможет перевести эту фразу на другом языке жителю p_3 и так далее, наконец, переводчик p_k сможет общаться с p_{k-1} и B.

Вам нужно определить наименьшее количество переводчиков, которое необходимо для того, чтобы жители A и B могли поговорить друг с другом.

Входные данные

Первая строка содержит одно число N $(2 \le N \le 1000)$ — количество жителей на острове. В каждой i-й из N последующих строк записаны: число L_i $(1 \le L_i < N)$ — количество языков, которыми владеет i-й житель; а затем, через пробел, L_i различных целых чисел, не превосходящих N, — номера этих языков (на острове все языки пронумерованы целыми числами от 1 до N). В последней $(N+2)$-й строке записаны через пробел номера жителей A и B —различные натуральные числа, не превосходящие N.

Выходные данные

Выведите -1, если беседа жителей A и B невозможна. Выведите 0, если A и B владеют общим языком, то есть им переводчики не нужны. В противном случае в первой строке запишите единственное натуральное K — наименьшее количество переводчиков, необходимое для поддержания беседы жителей A и B. Во второй строке укажите через пробел номера жителей переводчиков $p_1, p_2, \ldots, p_k$, которые обеспечивают разговор между A и B. Если возможных решений несколько, выведите любое из них.

Примеры

```
2
1 1
1 2
1 2
```

```
-1
```

```
2
1 1
1 1
1 2
```

```
0
```

```
3
1 2
1 1
2 2 1
1 2
```

```
1
3
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

```cpp
#include <iostream>
#include <queue>
#include <algorithm>

using namespace std;

int main() {
	int n;
	cin >> n;

	vector< vector<int> > v(2 * n);

	for (int i = 0; i < n; i++) {
		int cnt;
		cin >> cnt;
		for (int j = 0; j < cnt; j++) {
			int x;
			cin >> x;
			x--;
			v[i].push_back(n + x);
			v[n + x].push_back(i);
		}
	}

	int s, f;
	cin >> s >> f;
	s--; f--;
	vector<int> d(2 * n, 1e9);
	d[s] = 0;
	vector<int> p(2 * n);
	queue<int> q;
	q.push(s);

	while (!q.empty()) {
		int x = q.front();
		q.pop();
		for (int to : v[x]) {
			if (d[x] + 1 < d[to]) {
				d[to] = d[x] + 1;
				p[to] = x;
				q.push(to);
			}
		}
	}

	if (d[f] == 1e9) {
		cout << -1 << '\n';
		return 0;
	}

	vector<int> ans;
	while (f != s) {
		ans.push_back(f);
		f = p[p[f]];
	}
	ans.push_back(s);

	cout << ans.size() - 2 << '\n';
	reverse(ans.begin(), ans.end());
	for (int i = 1; i < ans.size() - 1; i++) {
		cout << ans[i] + 1 << ' ';
	}
```

```
63
64          return 0;
65  }
66
```

№ 5

100 баллов

Очень правильное число

Правильными суммами называют суммы нескольких подряд идущих натуральных чисел. Например, суммы 7 + 8 и 4 + 5 + 6 — правильные, а сумма 3 + 5 + 7 — нет, хотя результат суммирования во всех случаях равен 15. (Сумма из одного слагаемого 15 тоже считается правильной.) Правильностью целого положительного числа будем называть количество представлений этого числа в виде правильных сумм. Например, правильность числа 15 равна 4, поскольку 15 представляется в виде правильных сумм четырьмя способами: $15 = 7 + 8 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5$.

Из двух целых чисел более правильным считается то, у которого больше представлений в виде правильных сумм. При равенстве количеств таких представлений предпочтение в правильности отдаётся меньшему из них. Например, у чисел 15 и 30 правильность одинаковая (равна 4), однако более правильным считается число 15.

Вам необходимо составить программу, которая в заданном наборе целых положительных чисел находит самое правильное число и определяет его правильность.

Входные данные

В первой строке записано целое n - количество чисел в наборе $(2 \le n \le 10^3)$. Во второй строке содержится n целых положительных чисел a_i, каждое из которых не превосходит $2 \cdot 10^9$.

Выходные данные

Выведите два целых числа — самое правильное из всех чисел набора и значение его правильности.

Система оценивания

Все тесты в этой задаче оцениваются независимо. Вы получите баллы за каждый пройденный тест.

Примеры

```
2
15 30
```

```
15 4
```

```
3
20 30 20
```

```
30 4
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include <iostream>
using namespace std;

long long primary(long long m)
{
    int k = 2 + m % 2;
    while((m % k != 0) && (k * k < m))
        k += 2;
    if (k * k > m) return m; else return k;
}

int main()
{
    int n;
    cin >> n;
    pair < int,int > beautiful(1,1);
    // Самое красивое число и его красота пока равны 1
    for (int i = 0; i < n; ++i) {
        long long a, aa;
        cin >> a;
        aa = a;

        int k = 0;
        long long d[50];
        while (aa != 1) {
            d[k] = primary(aa);
            aa = aa / d[k];
            if (d[k] > 2) k++;
        }
        d[k] = -1;

        int beauty = 1;
        int count = 1;
        for (int i = 0; i < k; i++)  {
            if (d[i] == d[i+1]) count++;
            else {
                beauty *= count + 1;
                count = 1;
            }
        }

        if (beauty == beautiful.second)
            if (beautiful.first > a) beautiful.first = a;
        if (beauty > beautiful.second) {
            beautiful.first = a;
            beautiful.second = beauty;
        }
    }
    cout << beautiful.first << " " << beautiful.second << endl;
}
```

№ 6

100 баллов

Слияние энергетических сгустков

Вдоль узкого силового луча в исследовательском центре «Альтаир» выстроена цепочка из n энергетических сгустков. Каждый i-й сгусток обладает определённым уровнем заряда a_i.

В системе наблюдается нестабильность: два соседних сгустка могут слиться в один. Слияние возможно только в том случае, если заряд одного сгустка строго больше заряда соседа, находящегося непосредственно слева или справа от него. Процесс происходит мгновенно:

- Выбранный активный сгусток поглощает соседний.
- Заряд активного сгустка увеличивается на величину заряда поглощённого.
- Поглощённый сгусток исчезает, а цепочка «схлопывается», сокращая свою длину на единицу.
- Один и тот же сгусток может последовательно поглотить несколько соседей.
- Никакие два процесса поглощения не могут происходить одновременно.

Пример: Если цепочка имела заряды $[1, 2, 2, 2, 1, 2]$, то:

- Первый сгусток (1) не может поглотить второй (2), так как $1 < 2$.
- Второй сгусток (2) не может поглотить третий (2), так как заряды равны.
- Второй сгусток может поглотить первый $(2 > 1)$, тогда цепочка примет вид: $[3, 2, 2, 1, 2]$.

После завершения серии реакций в цепочке осталось ровно $k (k \le n)$ сгустков, заряды которых равны $b_1, b_2, \ldots, b_k$. Порядок следования сгустков в обеих последовательностях (a и b) — от начала луча к его концу.

Ваша задача: Восстановите любую возможную последовательность действий, которая привела к итоговому состоянию цепочки, или укажите, что такая трансформация физически невозможна.

В первом примере изначально в луче находилось $n = 6$ сгустков, их заряды в порядке очереди были $[1, 2, 2, 2, 1, 2]$. В результате должны остаться два сгустка с зарядами $[5, 5]$. Такое возможно, например, при следующей последовательности действий:

- второй сгусток поглощает левого от себя (то есть первого), очередь принимает вид $[3, 2, 2, 1, 2]$;
- первый сгусток (обратите внимание, это тот, который на предыдущем шаге был вторым) поглощает правого от себя (то есть второго), очередь принимает вид $[5, 2, 1, 2]$;
- четвёртый сгусток поглощает левого от себя (то есть третьего), очередь принимает вид $[5, 2, 3]$;
- третий сгусток поглощает левого от себя (то есть второго), очередь принимает финальный вид $[5, 5]$.

Обратите внимание, что при выводе реакций используется нумерация сгустков в их текущем порядке в луче.

Входные данные

В первой строке входных данных содержится целое число n ($1 \le n \le 500$) — количество энергетических сгустков изначально.

Во второй строке входных данных содержится n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — заряды сгустков изначально.

В третьей строке входных данных содержится целое число k ($1 \le k \le n$) — количество сгустков после реакций.

В четвёртой строке входных данных содержится k целых чисел $b_1, b_2, \ldots, b_k$ ($1 \le b_j \le 5 \cdot 10^8$) — заряды сгустков после излечения.

Сгустки перечислены в порядке следования в луче от начала к концу.

Выходные данные

В случае, если ни какие действия не могли привести к конечной последовательности, в единственной строке выходных данных выведите **NO**.

Иначе, в первой строке выходных данных выведите **YES**. В следующих $n - k$ строках выведите реакции в хронологическом порядке. В каждой строке выводите x — порядковый номер сгустка в текущей последовательности, который будет поглощать, и через пробел символ **L**, если сгусток, на позиции x по порядку, поглощает сгусток, стоящий перед ним в луче, или **R**, в случае, когда сгусток, на позиции x по порядку в луче, поглощает сгусток, стоящий за ним в луче. После очередного поглощения луч пронумеровывается заново.

Когда один сгусток поглощает другой, очередь схлопывается. Если решений несколько, выведите любое из них.

Система оценивания

Все тесты в этой задаче оцениваются независимо. Вы получите баллы за каждый пройденный тест.

Примеры

```
6
1 2 2 2 1 2
2
5 5
```

```
YES
2 L
1 R
4 L
3 L
```

```
5
1 2 3 4 5
1
15
```

```
YES
5 L
4 L
3 L
2 L
```

```
5
1 1 1 3 3
3
2 1 6
```

```
NO
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int n, a[1101]={0};
bool f=0;
vector<int> oti;
vector<string> ots;
int poed(int x, int s, int k){
        int ma=0,mi=-1;
        int i=x;
        while(i<n && s>0){
                s=s-a[i];
                if(a[i]>=ma){
                        ma=a[i];
                        if(i>x && a[i]>a[i-1])
                                mi=i;
                        if(i<n && s>0 && a[i]>a[i+1])
                                mi=i;
                }
                i++;
        }
        if((s!=0 || mi==-1 || a[mi]!=ma)&&(i-x!=1 || s!=0))
                f=1;
        else if(mi!=-1 && a[mi]==ma){
                if(mi>x && a[mi-1]<a[mi]){
                        for(int j=mi-x; j>=1;j--){
                                oti.push_back(j+k+1);
                                ots.push_back("L");}
                        for(int j=mi+1;j<i;j++){
                                oti.push_back(k+1);
                                ots.push_back("R");}
                }
                else{
                        for(int j=mi+1;j<i;j++){
                                oti.push_back(k+mi-x+1);
                                ots.push_back("R");}
                        for(int j=mi-x; j>=1;j--){
                                oti.push_back(j+k+1);
                                ots.push_back("L");}

                }
        }
        return i;
}

int main(){
        cin>>n;
        for(int i=0;i<n;i++)
                cin>>a[i];
        int j=0;
        int m;
        cin>>m;
        for(int i=0;i<m;i++){
                int l;
                cin>>l;
                j=poed(j,l,i);}
        if(j!=n || f==1)
                cout<<"NO";
        else {
                cout<<"YES"<<endl;
                for(int i=0;i<n-m;i++)
```

```
63                          cout<<oti[i]<<' '<<ots[i]<<endl;
64          }
65
66          return 0;
67  //      system("pause");
68  }
```