Заключительный этап

Индивидуальный предметный тур

Информатика. 8-11 класс

Задача III.1.1.1. Средневзвешенное абсолютное значение (10 баллов)

В ряде приложений используется формула взвешенного среднего абсолютного значения. Пусть задано nn измерений $x_1 \dots x_n$ некоторого параметра, например, сигнала ЭМГ. Определим весовые коэффициенты w_i по следующей формуле

$$w_i = \begin{cases} 1.5 & \text{если } 0,25n < i \leqslant 0,75n \\ 0,5 & \text{иначе} \end{cases}$$

Тогда, взвешенное среднее абсолютное значение вычисляется по формуле:

$$\frac{1}{n}\sum_{i=1}^{n} w_i|x_i|$$

где $|x_i|$ — абсолютное значение величины x_i.

Ваша задача: написать программу для вычисления взвешенного среднего абсолютного значения по указанной формуле.

Формат входных данных

В первой строке на вход поступает одно натуральное число n, $(4 \leqslant n \leqslant 1000)$. В следующей строке записаны n вещественных чисел x_i не более чем с тремя знаками после точки, $(-100 \leqslant x_i \leqslant 100)$.

Формат выходных данных

Вывести одно вещественное число — ответ к задаче. Ответ будет считаться верным, если модуль разности вашего ответа и ответа жюри будет различаться не более чем на 10^{-5}.

Примеры

Пример №1

Стандартный ввод
4 -2.4 -0.8 1.6 3.8
Стандартный вывод
1.675

Пример №2

Стандартный ввод
5 0.1 0.2 0.3 0.4 0.5
Стандартный вывод
0.25

Пояснения к примерам

Ответ в первом примере получается в результате следующих вычислений:

$$\frac{0,5 \cdot |-2,4| + 1,5 \cdot |-0,8| + 1,5 \cdot |1,6| + 0,5 \cdot |3,8|}{4} = \frac{1,2 + 1,2 + 2,4 + 1,9}{4} = 1,625$$

Ответ во втором примере получается в результате следующих вычислений:

$$\frac{0,5 \cdot |0,1| + 1,5 \cdot |0,2| + 1,5 \cdot |0,3| + 0,5 \cdot |0,4| + 0,5 \cdot |0,5|}{5} =$$

$$= \frac{0,05 + 0,3 + 0,45 + 0,2 + 0,25}{5} = 0,25$$

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;

int main(){
    int n;
    cin >> n;
    long double ans = 0;
    for (int i = 1; i <= n; ++i){
        long double x;
        cin >> x;
        if ( 4 * i >  n && 4 * i <= 3 * n ){
            ans += abs(x);
        }
        else
            ans += 0.5 * abs(x);
    }
    ans *= 1.0 / n;
    cout << fixed << setprecision(15) << ans;
}
```

Задача III.1.1.2. Интервальная тренировка (15 баллов)

Интервальной тренировкой называется чередование упражнений с высокой и низкой интенсивностью нагрузки. Признаком высокой интенсивности нагрузки является частота сердечных сокращений (ЧСС), составляющая 80%–100% от максимальной ЧСС, которая измеряется индивидуально. Программа интервальной тренировки обычно задается длительностью интервала высокой интенсивности нагрузки в секундах и количеством интервалов.

Вы разрабатываете программное обеспечение для фитнесс-трекера и перед вами поставлена следующая задача. По известным данным ЧСС за каждую секунду тренировки определить количество интервалов в интервальной тренировке. Также заданы максимальная ЧСС спортсмена $maxhr$ и минимальная длительность интервала в секундах dur.

Формально, вы должны найти количество участков на которых ЧСС не становится меньше 80% от $maxhr$ таких, что длительность интервала не меньше dur. Кроме того, вы должны найти суммарную продолжительность всех найденных интервалов.

Формат входных данных

В первой строке на вход через пробел подаются 3 целых числа: n, $maxhr$, dur — продолжительность тренировки в секундах, максимальная ЧСС спортсмена, минимальная длительность интервала, ($1 \leqslant n \leqslant 10000$), ($150 \leqslant maxhr \leqslant 240$), ($1 \leqslant dur \leqslant 300$). Во второй строке записаны nn натуральных чисел h_i, задающих ЧСС в каждую секунду, ($40 \leqslant h_i \leqslant maxhr$).

Формат выходных данных

Вывести в одной строке через пробел два числа: количество интервалов и суммарную продолжительность найденных интервалов.

Примеры

Пример №1

Стандартный ввод
12 200 2 155 160 164 167 170 161 159 165 150 150 165 165

Стандартный вывод
2 7

Пояснение к примеру

Для $maxhr = 200$ признаком высокой интенсивности тренировки является ЧСС от 160 и выше. В примере два интервала: со 2 по 6 секунду включительно и с 11 по 12 секунду включительно. Измерение с номером 8 не стало интервалом, поскольку в данном тесте $dur = 2$. Также, это измерение не вошло в суммарную продолжительность интервалов. Два найденных интервала имеют длину в 5 и в 2 секунды, поэтому, второе число в ответе равно 7.

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;

int main(){
    int n, maxh, dur;
    cin >> n >> maxh >> dur;
    int len = 0;
    int ans_len = 0, ans_int = 0;
    for (int i = 0; i < n; ++i){
        int a;
        cin >> a;
        if (4 * maxh <= 5 * a){
            len++;
        }
        else{
            if (len >= dur){

            ans_int++;
            ans_len += len;

            }
            len = 0;
        }
    }
    if (len >= dur){
        ans_int++;
            ans_len += len;
    }
    cout << ans_int << " " << ans_len;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```python
n,maxhr,dur=map(int,input().split())
clen=0
slen=0
cnt=0
for x in map(int,input().split()):
    if x>=maxhr*0.8:
        clen+=1
    else:
        if clen>=dur:
            slen+=clen
            cnt+=1
        clen=0
if clen>=dur:
    slen+=clen
    cnt+=1
print(cnt,slen)
```

Задача III.1.1.3. Пороговая функция (20 баллов)

В упрощенной математической модели нейрон описывается логической пороговой функцией вида:

$$\sum_{i=1}^{n} w_i x_i \geqslant p$$

где p — некоторая константа, задающая порог активации нейрона; w_i — веса синапсов; x_i — уровень сигнала в синаптической связи, который в упрощенной модели может быть либо нулем, либо единицей. При выполнении указанного условия нейрон активируется.

Для тестирования нейронной сети потребовалось решить следующие задачи. Во-первых, потребовалось узнать, какое минимальное количество синапсов должно быть активно (иметь уровень 1) для активации рассматриваемого нейрона. Обозначим это число за m. Далее, рассмотрим все наборы ровно из m синапсов, активация которых позволяет активировать нейрон. При этом возможно, что некоторые из синапсов не попадут ни в один из таких наборов. Требуется найти количество таких синапсов k и их порядковые номера в модели.

Вы должны написать программу, которая решит поставленные задачи. Обратите внимание, что частичные решения здесь также будут оцениваться. В частности, программы, которые позволят найти только число n, получат часть баллов за некоторые пройденные тесты. Кроме того, программы, которые найдут число k, но не смогут правильно вывести номера синапсов, вновь получат дополнительные баллы к ранее набранным.

Учитывайте, что правильное решение этой задачи должно иметь сложность близкую к линейной, поэтому программы, неоптимальные по времени, не смогут пройти часть тестов.

Формат входных данных

В первой строке через пробел на вход подается два целых числа n и p — количество синапсов и порог активации, $(1 \leqslant n \leqslant 100000)$, $(1 \leqslant p \leqslant 10^9)$. Во второй строке записано n целых чисел w_i — веса синапсов, $(w_i \leqslant 10000)$. Гарантируется, что сумма всех w_i больше или равна p.

Формат выходных данных

В первой строке программа должна вывести одно число m.

Во второй строке программа должна вывести одно число k. Если вы не решали вторую часть задачи, то эта и следующая строки могут остаться пустыми.

В третьей строке программа должна вывести требуемые номера синапсов. Номера синапсов начинаются с единицы и определяются их порядком во входных данных. Номера должны быть упорядочены по возрастанию и разделяться ровно одним пробелом. Если таких номеров нет, или вы не решали эту часть задачи, то строка может остаться пустой.

Пояснение к примеру

Для следующих векторов $X = (x_1, x_2, \ldots, x_8)$ функция активации будет истинной.

$$\begin{array}{ll} X = (1,1,1,0,0,0,1,0) & 66 \geqslant 60 \\ X = (0,1,1,0,1,0,1,0) & 63 \geqslant 60 \\ X = (1,1,0,0,1,0,1,0) & 60 \geqslant 60 \\ X = (0,1,1,0,0,1,1,0) & 61 \geqslant 60 \end{array}$$

Это все возможные вектора X, в которых ровно четыре единицы и условие выполняется. Составить вектор из трех единиц, при котором условие выполняется, невозможно. Таким образом, $m = 4$, Две компоненты вектора с номерами 4 и 8 во всех случаях равны 0, что и дает ответ.

Этот пример не входит в состав тестов, на которых оценивается решение.

Примеры

Пример №1

Стандартный ввод
8 60 12 19 15 3 9 7 20 5

Стандартный вывод
4 2 4 8

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    int n,p;
    cin>>n>>p;
    vector<pair<int,int>> nums(n);
    for (int i=0;i<n;i++) {
        cin>>nums[i].first;
        nums[i].second=i+1;
    }
    sort(nums.begin(),nums.end(),greater<pair<int,int>>());
    int m=0;
    int sum=0;
    for (;sum<p;m++)
        sum+=nums[m].first;
    cout<<m<<'\n';
    sum-=nums[m-1].first;
    int k=m;
```

```
    for (;k<n && sum+nums[k].first>=p;k++) {}
    vector<int> ans;
    for (;k<n;k++)
        ans.push_back(nums[k].second);
    sort(ans.begin(),ans.end());
    cout<<ans.size()<<'\n';
    for (unsigned i=0;i<ans.size();i++)
        cout<<ans[i]<<(i+1==ans.size()? "":" ");
    cout<<'\n';
    return 0;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
import operator

n, p = input().split()

n, p = int(n), int(p)

sinaps = list(map(int, input().split()))

norm_sinaps = []

for i in range(n):
    sinap_struc = {'num': i + 1, 'value': sinaps[i]}
    norm_sinaps.append(sinap_struc)

norm_sinaps.sort(key=operator.itemgetter('value'))

m = 0
sum = 0

for i in range(n - 1, 0, -1):
    sum += norm_sinaps[i]['value']
    m += 1
    if (sum >= p):
        break

pre_sum = 0
for i in range(n - 1, n - m, -1):
    pre_sum += norm_sinaps[i]['value']

exus = []

for i in range(0, n - m):
    if pre_sum + norm_sinaps[i]['value'] >= p:
        break
    exus.append(norm_sinaps[i]['num'])

exus.sort()

print(m)
print(len(exus))
for ex in exus:
    print(ex, end=' ')
```

Задача III.1.1.4. Отслеживание сна (25 баллов)

Современные умные часы или фитнесс-браслеты часто обладают функцией отслеживания сна. В каждый момент времени, анализируя показатели движения и сердечного ритм, они определяют спит ли человек и, если да, то в какой фазе сна (медленной или быстрой) он сейчас находится. Однако, на основании лишь этой информации достоверно определить фазу сна не всегда возможно. Датчики часто «теряют пульс», могут неверно интерпретировать движения и так далее. В связи с этим, возникла необходимость в программной коррекции поступившей информации, для чего был предложен следующий алгоритм.

На вход программы поступает последовательность нулей и единиц, кодирующих первичную классификацию фаз сна. Если последовательность одинаковых значений имеет длину большую некоторой заданной константы k, то мы считаем эти фрагменты последовательностей достоверными и не изменяем. Таким образом, вся последовательность разбивается на достоверные и недостоверные участки. Например для $k = 5$, последовательность 00111111001111010100000110000001 будет разбита следующим образом:
*00***111111***0011110101***00000***11***000000***1*.* Здесь жирным шрифтом выделены достоверные участки, а курсивом — недостоверные.

Далее, выполняются следующие преобразования. Если недостоверный фрагмент находится между двумя достоверными фрагментами с одинаковым символом или расположен с края, то все символы в нем меняются на символ ближайшего достоверного фрагмента. В частности, после этого преобразования рассмотренная последовательность примет вид
*11***111111***0011110101***00000***00***000000***0. Будем называть такие преобразования заменами первого вида.

Если же недостоверный фрагмент находится между двумя достоверными фрагментами с различными символами, то он разбивается на две части, каждая из которых может быть пустой. После этого символы в каждой части по отдельности заменяются на символ ближайшего достоверного фрагмента. При этом, разбиение выбирается так, чтобы количество замен было минимальным. Если разбиений с указанным свойством будет несколько, то выбирается такое разбиение, чтобы после выполнения замен нулей в последовательности стало как можно больше. Будем называть такие преобразования заменами второго вида.

Например, недостоверная последовательность 0011110101 должна быть разбита на две части так, чтобы слева после замен оказались единицы, а справа — нули. Если разбить ее на равные части 00111+10101, то слева придется сделать две замены, а справа — три, всего пять. Однако, есть несколько способов разбиения, при которых замен будет всего четыре. Это: 001111+0101, 00111101+01, 0011110101+. В последнем способе правая часть является пустой, что допустимо. Из этих трех способов выбирается первый, поскольку в этом случае количество нулей в ответе будет наибольшим. Таким образом, окончательный ответ: *11***111111***111111000***00000***00***000000***0*.*

В этой задаче часть тестов будут подобраны так, чтобы в них не было замен второго вида. Таким образом, программы в которых реализованы только замены первого вида получат часть баллов.

Обратите внимание, что в данной задаче правильным является решение линейной сложности, поэтому, часть тестов будет подобрана так, чтобы неэффективные решения не получили полных баллов.

Пример входных данных из условия задачи не входит в состав тестов, на которых проверяются решения участников.

Формат входных данных

В первой строке на вход подается одно целое число k — минимальная длина достоверного фрагмента последовательности, ($1 \leqslant k \leqslant 200000$). Во второй строке записана последовательность из нулей и единиц без пробелов и других разделителей. Длина последовательности не превосходит 200000. Гарантируется, что во входных данных есть хотя бы один достоверный фрагмент.

Формат выходных данных

В одной строке без пробелов и других разделителей требуется вывести скорректированную последовательность.

Примеры

Пример №1

Стандартный ввод
5 00111111001111010100000110000001
Стандартный вывод
11111111111111000000000000000000

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
vector<int> color;
void stclr(int l, int r, int cl){
    for (l; l <= r; ++l)
        color[l] = cl;
}

void setchar(string &s, int l, int r, char c){
    stclr(l, r, c == '1' ? 2 : 1);
    for (l; l <= r; ++l)
        s[l] = c;
}

int main(){
    int k; cin >> k;
    string s; cin >> s; s += '-';
    color.assign(s.size(), 0);
    int ln = 0;
    char prev = '!';
    vector<int> l, r;
```

```
    char ans_prev = '-';
    for (int i = 0; i < s.size(); ++i){
        if (s[i] != prev){
            if (ln >= k){
                if (prev == ans_prev){
                    setchar(s, r.back() + 1, i - ln - 1, prev);
                }

                l.push_back(i - ln);
                r.push_back(i - 1);
                ans_prev = prev;

                stclr(i - ln, i - 1, prev == '1' ? 2 : 1);
            }
            ln = 1;
        }
        else{
            ln++;
        }
        prev = s[i];
    }
    s.pop_back();
    for (int i = 0; i < l[0]; ++i)
        s[i] = s[l[0]];
    for (int i = (int)s.size() - 1; i > r.back(); --i)
        s[i] = s[r.back()];
    char left = s[l[0]], right;
    for (int i = r[0] + 1; i < l.back(); ++i){
        if (color[i] != 0){
            left = s[i];
        }
        else{
            int fl = i;
            while (color[i] == 0){
                i++;
            }
            right = s[i--];
            int fr = i;
            int kol1= 0;
            int ans = 10000000;
            char sc = right;
            for (int j = fl; j <= fr; ++j){
                if (s[j] != right)
                    kol1++;
            }
            ans = kol1;
            int razb = fl - 1;
            for (int j = fl; j <= fr; ++j){
                if (s[j] != right)
                    kol1--;
                else
                    kol1++;
                if (kol1 < ans || (kol1 == ans && left == '0')){
                    ans = kol1;
                    razb = j;
                }

            }
            setchar(s, fl, razb, left);
            setchar(s, razb + 1, fr, right);
```

```
83          }
84      }
85      cout << s << endl;
86  }
```

Задача III.1.1.5. Анализ ЭЭГ (30 баллов)

В этой задаче от вас требуется проанализировать оцифрованный сигнал волновой природы, например, ЭЭГ. Вы должны будете написать программу, которая найдет гребни и впадины всех волн. Это не такая простая задача. Во-первых, сигнал может затухать или пропадать, кроме того, сигнал может искажаться помехами, наконец, волна может иметь сложную форму и содержать несколько зубцов. Поэтому, мы будем решать задачу формально. Пусть известны n значений $h_1, h_2, \ldots, h_n$, задающих уровни сигнала в моменты времени от 1 до n, а также величина минимальной амплитуды m. Требуется выбрать некоторый набор из k значений $h_{i_1}, h_{i_2}, \ldots h_{i_k}$ так, что $i_1 < i_2 < \ldots < i_k$. Для каждого выбранного значения необходимо указать, гребнем или впадиной оно является. При этом должны выполняться следующие условия.

1. Если значение h_{i_j} является гребнем, то соседние значения $h_{i_{j-1}}$ и $h_{i_{j+1}}$ должны быть впадинами и наоборот, если h_{i_j} является впадиной, то $h_{i_{j-1}}$ и $h_{i_{j+1}}$ должны быть гребнями. Иными словами, гребни и впадины должны чередоваться.
2. Если значение h_{i_j} является гребнем то $h_{i_j} = \max(h_{i_{j-1}}, h_{i_{j-1}+1}, \ldots, h_{i_{j+1}})$. Аналогично, если h_{i_j} является впадиной то $h_{i_j} = \min(h_{i_{j-1}}, h_{i_{j-1}+1}, \ldots, h_{i_{j+1}})$. Иными словами, гребень должен быть максимумом на участке между двумя соседними впадинами, также как и впадина должна быть минимумом на участке между соседними гребнями. Если впадина или гребень является первым (последним) измерением в наборе, то рассматривается участок от начального (до конечного) измерения.
3. Для всех j от 1 до $k-1$ должно выполняться неравенство $|h_{i_j} - h_{i_{j+1}}| \geqslant m|$. Иными словами, амплитуда волны, то есть расстояние от гребня до впадины, должно быть не меньше чем минимальное значение амплитуды m.
4. Значение k должно быть максимально возможным. Это условие должно гарантировать, что мы найдем все гребни и впадины.

Вы должны написать программу, которая найдет подмножество номеров i_1, $i_2, \ldots, i_k$, для которого выполнятся все указанные условия. Если таких подмножеств окажется несколько, то можно вывести любое.

Обратите внимание, что здесь требуется формальное решение поставленной задачи. Это означает, что входные данные не будут результатами реальных измерений, а будут подобраны так, чтобы проверить правильность решения для произвольных, в том числе случайных, чисел. Кроме того, предполагается, что решение этой задачи должно иметь линейную или почти линейную сложность. Поэтому, неэффективные алгоритмы не получат полных баллов за эту задачу.

Пример входных данных из условия задачи не входит в состав тестов, на которых проверяются решения участников.

Формат входных данных

В первой строке на вход подаются два целых числа n и m — количество измерений и минимальная амплитуда, $(1 \leqslant n \leqslant 200000)$, $(1 \leqslant m \leqslant 10^9)$. Во второй строке через пробел записаны n измерений h_i. Все измерения являются целыми числами, $(0 \leqslant h_i \leqslant 10^9)$.

Формат выходных данных

В первой строке вывести число k — общее количество гребней и впадин. Во второй строке через пробел вывести номера измерений, которые являются гребнями или впадинами. Нумерация измерений начинается с единицы. Возможен случай, что гребней или впадин не найдется. В этом случае вторую строку надо оставить пустой.

Примеры

Пример №1

Стандартный ввод
20 10 4 1 3 5 9 10 8 5 9 11 7 4 1 2 3 0 5 7 10 7

Стандартный вывод
4 2 10 16 19

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    int n,m;
    cin>>n>>m;
    vector<int> h(n),ans;
    for (auto &x:h)
        cin>>x;
    int vmin=2000000000,vmax=-2000000000,pmin=-1,pmax=-1,kind=0;
    /* kind=1 - ищем впадину (pmin<=pmax), -1 - гребень (pmax<=pmin)*/
    for (int i=0;i<n;i++) {
        if (h[i]<vmin) {
        vmin=h[i],pmin=i;
        if (kind==1)
            vmax=h[i],pmax=i;
        }
        if (h[i]>vmax) {
        vmax=h[i],pmax=i;
        if (kind==-1)
            vmin=h[i],pmin=i;
        }
```

```
25          if (vmax-vmin>=m) {
26          if (pmin<pmax)
27              ans.push_back(pmin+1),pmin=pmax,vmin=vmax,kind=-1;
28          else
29              ans.push_back(pmax+1),pmax=pmin,vmax=vmin,kind=1;
30          }
31      }
32      if (kind==1)
33          ans.push_back(pmin+1);
34      else if (kind==-1)
35          ans.push_back(pmax+1);
36      cout<<ans.size()<<'\n';
37      for (unsigned i=0;i<ans.size();i++)
38          cout<<ans[i]<<(i+1==ans.size() ? "":" ");
39      cout<<'\n';
40      return 0;
41  }
```

Математика. 8-9 класс

Задача III.1.2.1. (15 баллов)

При анализе некоторого текста выяснилось, что частота вхождения буквы «и» в нем не менее 7,4 % и не более 7,5 %. Какое наименьшее количество букв может быть в тексте?

Решение

Пусть всего в тексте n букв, из них k букв «и». Тогда по условию $0,074 \leqslant \frac{k}{n} \leqslant 0,075$, откуда $\frac{40}{3}k \leqslant n \leqslant \frac{500}{37}k$.

При $k = 1$ получаем $13\frac{1}{3} \leqslant n \leqslant 13\frac{19}{37}$; таких натуральных n не существует.

При $k = 2$ имеем: $26\frac{2}{3} \leqslant n \leqslant 27\frac{1}{37}$, откуда $n = 27$. При $k \geqslant 3$: $n \geqslant 40$. Итак, наименьшее возможное $n = 27$ (при этом $k = 2$).

Система оценки

1. Полное обоснованное решение (включая оценку) — 15 баллов.
2. Приведен только верный ответ — 5 баллов.
3. Приведен верный ответ и пример (значение k) — 10 баллов.
4. Задача не решена или решена неверно — 0 баллов.

Ответ: 27.

Задача III.1.2.2. (15 баллов)

Подводя итоги года, директор турфирмы выяснил, что всего за год было продано 4140 путевок, причем в каждом месяце их количество могло составить 270, 370 или

420 (все три варианта встречались в отчете). В скольких месяцах могло быть продано ровно 370 путевок?

Решение

Пусть в x месяцах было продано по 270 путевок, в y месяцах — по 370 и в z месяцах — по 420. Тогда получаем систему уравнений:

$$\begin{cases} x + y + z = 12, \\ 270x + 370y + 420z = 4140. \end{cases}$$

Из второго уравнения получаем: $37y = 414 - 27x - 42z$. Правая часть уравнения делится на 3, поэтому $y \vdots 3 : y = 3y_1$. Отсюда следует: $9x + 37y_1 + 14z = 138$. С учетом первого уравнения системы $z = 12 - x - 3y_1$, тогда $9x + 37y_1 + 1412 - x - 3y_1 = 138$, откуда $x + y_1 = 6$ и $z = 6 - 2y_1$. Поскольку $z > 0$, получаем $y_1 \leqslant 2$.

Рассмотрим два возможных варианта:

1) $y_1 = 1$. Тогда $x = 5$, $y = 3$, $z = 4$.

2) $y_2 = 2$. Тогда $x = 4$, $y = 6$, $z = 2$.

Итак, возможные значения y равны 3 и 6.

Система оценки

1. Полное обоснованное решение — 15 баллов.
2. Имеются частичные продвижения, но итоговое количество посчитано неверно — 5 баллов.
3. Задача не решена или решена неверно — 0 баллов.

Ответ: 3 или 6.

Задача III.1.2.3. (20 баллов)

На конкурс роботов-вездеходов для освоения Венеры были представлены 50 машин. Каждый образец проходил три теста:

1. устойчивость к высокой температуре,
2. устойчивость к высокому давлению,
3. надежность механики.

Только 14 машин прошли проверку высокой температурой. Кроме того, 28 роботов провалили более одного из трех тестов. Наконец, 36 образцов провалили тест на давление или тест на надежность механики, но никто не провалил эти два теста одновременно. Сколько роботов прошли все три теста?

Решение

Поскольку никто не провалился ни в тесте на давление, ни в тесте проверки механики одновременно, все роботы, провалившие, по крайней мере, два теста, должны были провалиться из-за высокой температуры. Это дает нам $(50-14)-28=8$ образцов, не прошедших только проверку высокой температурой. Вместе с 36 машинами, не прошедшие проверку давлением или недостаточно надежные, было отсеяно 44 образца, поэтому из всех кандидатов были отобраны только 6.

Система оценки

1. Полное обоснованное решение — 20 баллов.
2. Имеются частичные продвижения, но итоговое количество посчитано неверно — 5 баллов.
3. Найдено количество роботов, не прошедших испытание температурой — 10 баллов.
4. Задача не решена или решена неверно — 0 баллов.

Ответ: 6.

Задача III.1.2.4. (20 баллов)

Компьютерная сеть состоит из 10 компьютеров, каждый из которых с вероятностью 13 может выйти из строя (независимо от каждого из остальных). Компьютеры соединены последовательно, и при выходе из строя одного из них (с номером $n \geqslant 2$) автоматически выходят из строя и все компьютеры с номерами от 1 до n-1 включительно. Найдите вероятность того, что из строя выйдут ровно 4 компьютера.

Решение

Вероятность того, что компьютер **не** выйдет из строя сам по себе (хотя может выйти из строя из-за поломки следующих за ним), равна $1-\dfrac{1}{3}=\dfrac{2}{3}$.

Ровно 4 компьютера выйдут из строя только в одном случае: если выйдет из строя 4-й компьютер и **не** выйдет из строя ни один из компьютеров с номерами от 5 до 10. Вероятность этого события равна $\dfrac{1}{3}\cdot\left(\dfrac{2}{3}\right)^6=\dfrac{64}{2187}$.

Система оценки

1. Сформулирована идея, когда из строя могут выйти ровно 4 компьютера — 10 баллов.
2. Найдена вероятность того, что компьютер не выйдет из строя сам по себе — 5 баллов.
3. Верно найдена искомая вероятность — 5 баллов.
4. Вычислительная ошибка: минус 5 баллов.
5. Задача не решена или решена неверно — 0 баллов.

Ответ: $\frac{64}{2187}$.

Задача III.1.2.5. (30 баллов)

Четырехугольник $PQRS$ выпуклый, $PQ = QR = RS$. O — точка пересечения диагоналей, M — точка пересечения продолжения сторон PQ и RS. Из точки O опущен перпендикуляр ON на сторону QR. Известно, что $QM = 3$, $QN = 4,5$, $RM = 6$. Докажите, что N — точка касания вневписанной окружности ΔQMR, и найдите RN.

(Вневписанная окружность треугольника касается одной из его сторон и продолжения двух других).

Решение

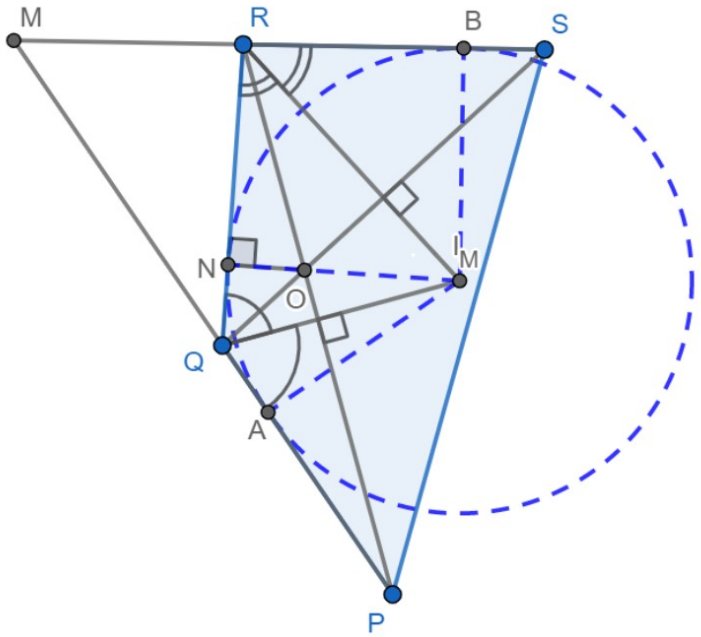
Центр вневписанной окружности I_M треугольника QMR лежит на пересечении биссектрис QI_M и RI_M двух внешних углов (при вершинах Q и R соответственно).

Треугольники PQR и QRS — равнобедренные, поэтому их биссектрисы также являются высотами. Следовательно, $QI_M \perp OR$ и $RI_M \perp QO$. Таким образом, I_M — ортоцентр (точка пересечения прямых, содержащих высоты) треугольника QOR. Поэтому точки N, O и I_M лежат на одной прямой и N — точка касания вневписанной окружности со стороной QR. Первая часть задачи решена.

Далее, обозначим как A и B точки касания вневписанной окружности с прямыми MQ и MR соответственно. Тогда по свойству отрезков касательных: $QN = QA$, $RN = RB$, $MA = MB$. Отсюда:

$$MQ + QN = MQ + QA = MA = MB = MR + RB = MR + RN$$

Т. е. $MQ + QN = MR + RN$. Подставляя длины QN, QR и RM из условия, получаем $RN = 1,5$.



Система оценки

1. Полное обоснованное решение — 30 баллов.
2. Доказано, что N — точка касания вневписанной окружности — 10 баллов.
3. Сформулировано и/или доказано свойство точки касания вневписанной окружности ($MQ + QN = MR + RN$) — 10 баллов.
4. Задача не решена или решена неверно — 0 баллов.

Ответ: 1,5.

Математика. 10-11 класс

Задача III.1.3.1. (15 баллов)

Петя забыл пароль от сайта подготовки к ЕГЭ. Он помнит, что пароль представляет собой набор из 11 букв русского алфавита (используются 30 букв) и в нем есть подряд идущие буквы, образующие слова «рация» и «опера». Сколько существует возможных вариантов такого пароля?

Решение

Возможны два вида паролей: 1) в которых слова «рация» и «опера» не пересекаются; 2) в которых есть слово «операция». Паролей вида 1) может быть 180. В паролях вида 2) слово «операция» может быть на четырех местах, остальные три места можно заполнить 30^3 способами. Всего $4 \cdot 27000 = 108000$ паролей. Количество всех паролей 108180.

Система оценки

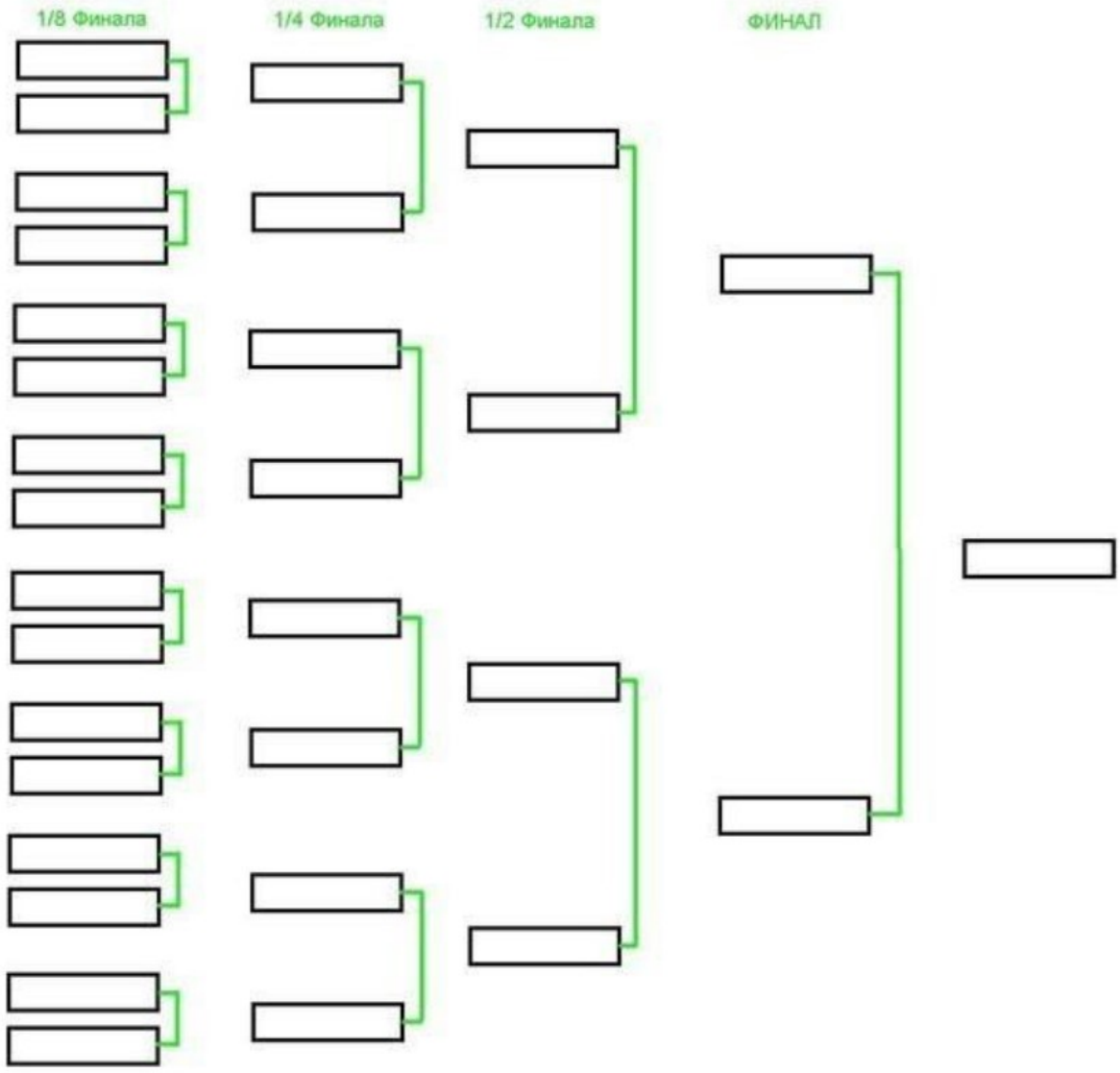
1. Верно посчитано количество паролей типа 1) — 5 баллов.
2. Верно посчитано количество паролей типа 2) — 10 баллов.
3. Задача не решена или решена неверно — 0 баллов.

Ответ: 108 180.

Задача III.1.3.2. (15 баллов)

В межгалактическом турнире по квидичу принимают участие 16 команд, в том числе сборные Альфа-Центавра и Бета-Лиры. Уровень мастерства всех команд одинаков, и в каждой игре с вероятностью $\frac{1}{2}$ может победить любая из двух команд (ничьих не бывает). Турнир проходит по схеме «на выбывание»: сначала проходят 8 отборочных матчей, затем их победители встречаются в четвертьфиналах, после чего их победители выходят в полуфиналы и, наконец, два победителя полуфиналов встречаются в финале (см. рис.).

Распределение команд в отборочном туре происходит путем случайной жеребьевки. Найдите вероятность того, что сборные Альфа-Центавра и Бета-Лиры встретятся в финале турнира.



Решение

Для того чтобы команды Альта-Центавра (АЦ) и Бета-Лиры (БЛ) встретились в финале, они должны, во-первых, попасть в разные подгруппы, а во-вторых, каждая в своей подгруппе должна выйти в финал.

Пусть команда АЦ попала в первую подгруппу, тогда вероятность попадания БЛ во вторую подгруппу составляет $\frac{8}{15}$. Вероятность выхода каждой из команд в финал в своей подгруппе равна $(\frac{1}{2})^3 = \frac{1}{8}$.

Для нахождения искомой вероятности перемножим все три вероятности:

$$\frac{8}{15} \cdot \frac{1}{8} \cdot \frac{1}{8} = \frac{1}{120}$$

.

Система оценки

1. Полное обоснованное решение — 15 баллов.
2. Частичное продвижение, но искомая вероятность не найдена — 5 баллов.
3. Одна вычислительная ошибка: минус 5 баллов.
4. Задача не решена либо решена неверно — 0 баллов.

Ответ: $\frac{1}{120}$.

Задача III.1.3.3. (20 баллов)

Ослику Иа на день рождения подарили 2021 шишку, которые он сложил в большую кучу. В гости к Иа пришел Пятачок и предложил сыграть в следующую игру. За один ход игры нужно выбрать кучу, содержащую не менее трех шишек, выбросить в пруд одну из них, а оставшиеся шишки разделить на две кучи (не обязательно поровну). Ослик и Пятачок ходили по очереди. В какой-то момент (оба игрока успели сделать хотя бы один ход) выяснилось, что в каждой куче оказалось ровно d ($d > 3$) шишек. Чему может равняться d?

Решение

После каждого хода число шишек на 1 уменьшается, а число кучек на 1 увеличивается. Поскольку первоначально шишек было 2021, а кучек — одна, то после n ходов ($n \geqslant 2$) шишек окажется $(2021-n)$, а кучек станет $(n+1)$. В задаче требуется, чтобы выполнялось равенство $2021 - n = d(n+1)$, или $d = \frac{2022}{n+1} - 1$. Разложим 2022 на простые множители: $2022 = 2{\cdot}3{\cdot}337$. Откуда $n+1$ может быть равно $3, 6, 337, 2{\cdot}337, 3{\cdot}337$, что дает значения $d = 1010; 673; 336; 5; 2; 1$. Условию удовлетворяют: $673, 336, 5$. Способ получения этих кучек таков: с самого начала, выбрасывая одну шишку, отделяем от первоначальной кучи нужное число шишек, потом с большей из оставшихся куч делаем то же самое и т. д.

Система оценки

1. Полное обоснованное решение — 20 баллов.
2. Найдены какие-либо значения d — 5 баллов.
3. Составлено уравнение, связывающее n и d, но значения не найдены — 10 баллов.
4. Найдены все значения без описания процедуры получения нужного результата — 15 баллов.
5. Задача не решена или решена неверно — 0 баллов.

Ответ: $d = 673; 336; 5$.

Задача III.1.3.4. (20 баллов)

Решите уравнение:

$$(\sqrt{2x+11} - x - 2)(\sqrt{x+3} + x) = 3 - x^2 + x$$

.

Решение

Областью допустимых значений уравнения является промежуток $[-3;+\infty)$. На этом множестве выполняется равенство $3-x^2+x=(\sqrt{x+3}+x)(\sqrt{x+3}-x)$. Поэтому исходное уравнение можно записать в виде $(\sqrt{x+3}+x)(\sqrt{2x+11}-\sqrt{x+3}-2)=0$, что равносильно:

$$\begin{cases} x+3+x=0, & \text{(III.1.1)} \\ 2x+11-x+3-2=0 & \text{(III.1.2)} \end{cases}$$

Решим уравнение (III.1.1):

$$\sqrt{x+3}=-x \Leftrightarrow \begin{cases} x+3=x^2, \\ x\leqslant 0 \end{cases} \Leftrightarrow \begin{cases} x=\frac{1\pm\sqrt{13}}{2}, \\ x\leqslant 0 \end{cases} \Leftrightarrow x=\frac{1-\sqrt{13}}{2}$$

Решим уравнение (III.1.2):

$$\sqrt{2x+11}=\sqrt{x+3}+2 \Leftrightarrow 2x+11=x+3+4\sqrt{x+3}+4 \Leftrightarrow$$

$$\Leftrightarrow 2\sqrt{x+3}=x+2 \Leftrightarrow \begin{cases} x^2=8, \\ x\geqslant -2 \end{cases} \Leftrightarrow x=2\sqrt{2}$$

Система оценки

1. Полное обоснованное решение — 20 баллов.
2. Имеются частичные продвижения, но задача полностью не решена — 5 баллов.
3. Уравнение сведено к совокупности более простых — 10 баллов.
4. Вместе с решениями найдены посторонние корни — 15 баллов.
5. Задача не решена или решена неверно — 0 баллов.

Ответ: $2\sqrt{2}; \frac{1-\sqrt{13}}{2}$.

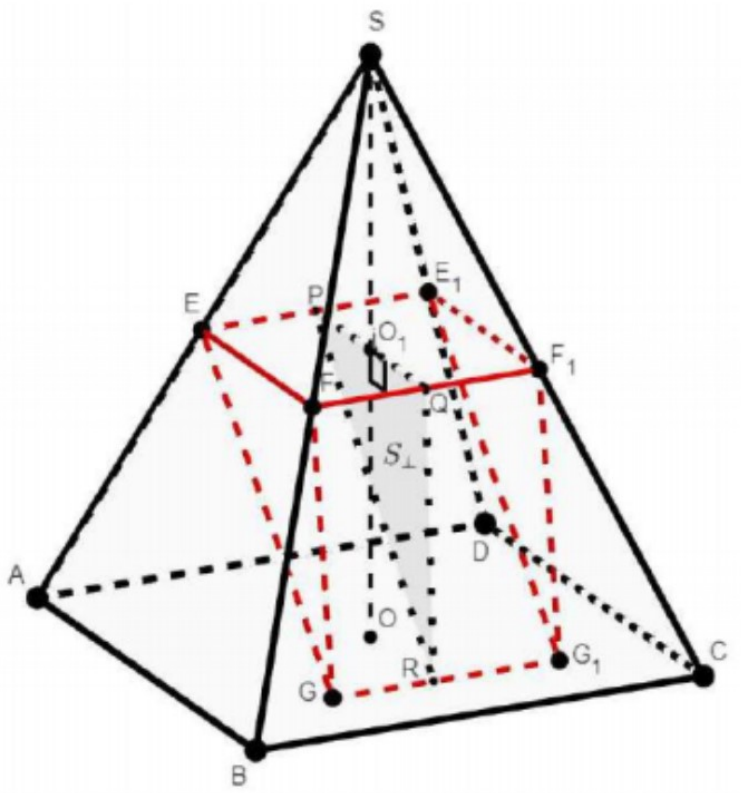
Задача III.1.3.5. (30 баллов)

В защитный кожух, поверхность которого сделана из платины в виде правильной четырехугольной пирамиды, помещен процессор в виде треугольной призмы, боковая грань которой параллельна основанию пирамиды (вершины этой грани лежат на боковых ребрах пирамиды), а одно из боковых ребер принадлежит основанию пирамиды. Найдите наибольшее возможное значения отношения объема процессора к объему кожуха.

Решение

Обозначим пирамиду $SABCD$ (S — вершина, $ABCD$ — основание), а призму $EFGE_1F_1G_1$. Из условия параллельности основания призмы основанию пирамиды

следует, что $EF||AB$, $FF_1||BC$, $F_1E_1||CD$ и $EE_1||AD$. Таким образом, боковая грань призмы EFF_1E_1 — квадрат, а боковое ребро GG_1 лежит в плоскости ABC и параллельно ребрам AD и BC основания пирамиды (обратите внимание: призма не обязательно прямая, а ребро GG_1 не обязательно проходит через центр основания пирамиды O). Обозначим O_1 — центр квадрата EFF_1E_1.



Пусть $AB = a, SO = H, SO_1 = x$.

Объем кожуха равен $V_{SABCD} = \frac{1}{3}S_{ABCD} \cdot SO = \frac{1}{3}a^2H$.

Вычислим объем процессора по формуле $V_{EFGE_1F_1G_1} = S_\perp \cdot EE_1$, где $S_\perp = S_{PQR}$ — объем перпендикулярного сечения призмы, т. е. треугольника PQR.

Основание треугольника $PQ = EF = AB \cdot \frac{EF}{AB} = AB \cdot \frac{SO_1}{SO} = a \cdot \frac{x}{H} = EE_1$, высота треугольника равна $OO_1 = SO - SO_1 = H - x$.

Таким образом, $S_\perp = S_{PQR} = \frac{1}{2}PQ \cdot OO_1 = \frac{1}{2}a \cdot \frac{x}{H} \cdot (H - x)$, тогда:

$$V_{EFGE_1F_1G_1} = S_\perp \cdot EE_1 = \frac{1}{2}\frac{a^2x^2(H-x)}{H^2}$$

Искомое отношение объемов процессора и кожуха равно:

$$f(k) = \frac{V_{EFGE_1F_1G_1}}{V_{SABCD}} = \frac{1}{2}\frac{a^2x^2(H-x)}{H^2} \cdot \frac{1}{\frac{1}{3}a^2H} = \frac{3}{2}k^2(1-k),$$

где $k = \frac{x}{H} \in (0;1)$.

Найти наибольшее значение этого отношения можно по-разному.

1 способ. Исследуем функцию $f(k) = \frac{3}{2}k^2(1-k) = \frac{3}{2}(k^2-k^3)$ с помощью производной: $f'(k) = \frac{3}{2}(2k-3k^2) = \frac{9}{2}k(\frac{2}{3}-k)$. Таким образом, $f'(k) > 0$ при $0 < k < \frac{2}{3}$, $f'(k) < 0$

при $\frac{2}{3} < k < 1$, поэтому $k = \frac{2}{3}$ — локальный максимум на $(0; 1)$ и в этой точке достигается наибольшее значение на рассматриваемом промежутке:

$$f_{max} = f(\frac{2}{3}) = \frac{3}{2}(\frac{2}{3})^2(1 - \frac{2}{3}) = \frac{2}{9}$$

.

2 способ. Воспользуемся неравенством между средним арифметическим и средним геометрическим для трех положительных чисел: k, k и $2 - 2k$:

$$\sqrt[3]{k \cdot k \cdot (2 - 2k)} \leqslant \frac{1}{3}(k + k + 2 - 2k) = \frac{2}{3}$$

причем равенство достигается только при $k = 2 - 2k$, т. е. при $k = \frac{2}{3}$.

Таким образом,

$$f(k)\frac{3}{2} \cdot \frac{1}{2}(\sqrt[3]{k \cdot k \cdot (2 - 2k)})^3 \leqslant \frac{3}{4}(\frac{2}{3})^3 = \frac{2}{9}$$

Наибольшее значение $f_{max} = f(\frac{2}{3}) = \frac{2}{9}$.

Система оценки

1. Полное обоснованное решение — 30 баллов.
2. Составлена функция объема призмы — 15 баллов.
3. Функция исследована на наибольшее значение на промежутке — 10 баллов.
4. Верно найдено отношение объемов — 5 баллов.

Ответ: $\frac{2}{9}$.