Заключительный этап

Индивидуальный предметный тур

Информатика. 8–11 класс

Задача III.1.1.1. Выбор места для дата-центра (7 баллов)

Крупная IT-компания планирует построить очередной дата-центр. Для постройки нужно выбрать одну из n заданных местностей. В данной задаче нас будет интересовать только один параметр — стоимость поддержания заданной температуры в помещении планируемого дата-центра. Этот параметр формируется из двух составляющих: среднегодовой температуры в текущей местности и стоимости электроэнергии. Для упрощения рассуждений будем считать, что среднегодовая температура в текущей местности держится постоянно и в помещении дата-центра будет именно эта температура, если не предпринимать дополнительных мер. Этот параметр важен в связи с тем, что для оптимальной работы дата-центра необходимо в нем поддерживать заданную заранее оптимальную температуру T. При необходимости можно установить оборудование для охлаждения или, наоборот, для подогрева. И то, и другое потребляет электроэнергию.

В связи с этим каждая местность имеет следующие параметры:

- название (непустая строка из больших и малых букв латиницы, а также пробелов, длина названия не превосходит 60);
- среднегодовая температура (целое число от -50 до $+50$ включительно, знак «+» не пишется в явном виде);
- стоимость 1 кВт электроэнергии (целое число от 1 до 10^5 включительно).

Формат входных данных

В первой строке содержатся три числа через пробел: T — оптимальная температура функционирования дата-центра, C — потребляемая оборудованием мощность в кВт для охлаждения помещения на 1 градус в течение года, W — потребляемая оборудованием мощность в кВт для подогрева помещения на 1 градус в течение года ($-50 \leqslant T \leqslant 50$, $1 \leqslant C, W \leqslant 10^5$).

Во второй строке находится число n — количество рассматриваемых местностей. $1 \leqslant n \leqslant 100$.

В следующих n строках находятся описания параметров местностей в формате <название><стоимость 1 кВт электроэнергии> <среднегодовая температура>. Параметры разделены одним пробелом. Внутри названия могут так же присутствовать пробелы. В начале или в конце названия пробелы отсутствуют. Все названия различны.

Формат выходных данных

Требуется вывести список названий заданных местностей в порядке возрастания затрат на поддержание заданной оптимальной температуры. В случае равенства затрат названия нужно выводить в алфавитном порядке. Каждое название должно быть выведено в отдельной строке. Для каждой местности в списке в ее строке после названия нужно вывести еще одно число ровно через один пробел — затраты на поддержание в помещении дата-центра заданной температуры T в течение года. Название должно быть выведено точно в том виде, в каком оно встречается в исходных данных.

Примеры

Пример №1

Стандартный ввод
-2 1345 269 7 Oregon 24 10 California 32 16 Costa Rica 12 22 Yakutia 13 -10 The United Kingdom of Great Britain and Northern Ireland 25 9 Magadan 15 -3 Antarctic coast 240 -8

Стандартный вывод
Magadan 4035 Yakutia 27976 The United Kingdom of Great Britain and Northern Ireland 369875 Antarctic coast 387360 Costa Rica 387360 Oregon 387360 California 774720

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    ll T, C, W;
    cin >> T >> C >> W;
    int n;
    cin >> n;
```

```
    vector<pair<ll, string> > v;
    string s;
    getline(cin, s);
    for(int i = 0; i < n; i++){
        getline(cin, s);

        int y = 0;
        while((s[y] > '9' || s[y] < '0') && s[y] != '-')
            y++;
        string name = s.substr(0, y);
        string ds = s.substr(y);

        stringstream ss;
        ss << ds;
        int cost, t;
        ss >> cost >> t;
        ll tp;
        if(t < T)
            tp = (T - t) * W * cost;
        else
            tp = (t - T) * C * cost;

        v.push_back({tp, name});
    }

    sort(v.begin(), v.end());
    for(auto q : v)
            cout<<q.second<<q.first<<endl;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
ss = input().split()
temp = int(ss[0])
cool = int(ss[1])
heat = int(ss[2])
n = int(input())
ans = []
for k in range(n):
    s = input()
    a = ""
    yk = 0
    cur_temp = ""
    price = ""
    for j in range(len(s) - 1, -1,-1):
        if s[j].isalpha():
            name = s[:j + 1]
            break
        elif s[j] == ' ':
            if (yk == 0 and len(a) > 0):
                cur_temp = int(a)
                yk += 1
            elif (yk == 1 and len(a) > 0):
                price = int(a)
            a = ''
        else:
            a = s[j] + a
```

```
26      if (cur_temp > temp):
27          ans.append([(cur_temp - temp) * cool * price, name])
28      else:
29          ans.append([(temp - cur_temp) * heat * price, name])
30  ans.sort()
31  for i in ans:
32      print(i[1],i[0])
```

Задача III.1.1.2. Пасьянс для WALL-E (15 баллов)

Робот WALL-E нашел игральную колоду, в которой изначально было 54 карты, но к моменту нахождения некоторые карты, возможно, были утеряны. Теперь робот использует колоду для развлечения, которое люди бы назвали раскладыванием пасьянса. В свободное от уборки мусора время, он выкладывает все имеющиеся у него карты в ряд, при этом некоторые из них оказываются лицевой стороной вверх, а некоторые обратной. После этого он много раз проделывает следующую операцию: находит первую слева карту, лежащую лицевой стороной вверх, после чего переворачивает все карты слева от нее и саму эту карту. Процесс продолжается до тех пор, пока в ряду есть хотя бы одна карта, лежащая лицевой стороной вверх.

Для заданного исходного расположения карт требуется выяснить, сколько раз придется WALL-E проделать указанную операцию, чтобы пасьянс был завершен.

Формат входных данных

В первой строке находится одно число n — количество карт в игральной колоде, $1 \leqslant n \leqslant 54$. Во второй строке находится строка из n символов A и B, описывающая положение карт в ряду слева направо. К сожалению, неизвестно, какая буква соответствует лицевой, а какая обратной стороне карты, но известно, что различно расположенные карты обозначены разными буквами, а одинаково расположенные карты обозначены одинаковыми буквами.

Формат выходных данных

Вывести два числа через пробел по неубыванию — количество операций WALL-E в случае каждого из двух возможных расположений карт, соответствующих исходной строке.

Примеры

Пример №1

Стандартный ввод
4 ABBA

Стандартный вывод
6 9

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int n;
    cin >> n;
    string s;
    cin >> s;
    reverse(s.begin(), s.end());

    ll A = 0, B = 0;
    for(int i = 0; i < n; i++){
      A = A * 2 + (s[i] == 'A');
      B = B * 2 + (s[i] == 'B');
    }

    if(A > B) swap(A, B);
    cout<<A<<' '<<B;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```python
n = int(input())
s = input()
dp = [0] * 100
dp[1] = 1
for i in range(2, 60):
    dp[i] = dp[i - 1] * 2 + 1
ans1 = 0
ans2 = 0
for i in range(n):
    if (s[i] == 'A'):
        ans1 += dp[i] + 1
    else:
        ans2 += dp[i] + 1
print(min(ans1, ans2), max(ans1,ans2))
```

Задача III.1.1.3. Квантовая зависимость (18 баллов)

При разработке прототипа квантового компьютера было принято решение расположить n его регистров на плоскости в пределах некоторого прямоугольного участка ширины k. Если ввести координаты на плоскости, то первые k регистров будут иметь координаты от $(0, 0)$ до $(k-1, 0)$, следующие k регистров (если они есть) будут иметь

координаты от (0, 1) до $(k-1, 1)$, следующие k регистров (если они есть) будут иметь координаты от (0, 2) до $(k-1, 2)$ и т. д. Последний ряд может состоять из меньшего чем k числа регистров.

При исследовании этой компоновки было замечено, что между некоторыми парами регистров возникает плохо влияющая на работу системы зависимость. Более точно, регистр A с координатами (x_i, y_i) влияет на работу регистра B с координатами x_j, y_j, если $x_i \leqslant x_j$ и $y_i \leqslant y_j$. Это значит, в частности, что регистр воздействует и сам на себя. Это воздействие весьма незначительное, но, суммируясь, может привести к непрогнозируемым ошибкам в работе системы.

Требуется по заданному количеству регистров n найти такое k, при котором суммарное количество зависимых пар было бы как можно меньше.

Формат входных данных

На вход подается одно число n $1 \leqslant n \leqslant 10^6$.

Формат выходных данных

Вывести одно число k — ширину прямоугольного участка, в пределах которого можно расположить n регистров так, что число зависимых пар будет наименьшим. Если минимум достигается на нескольких значениях k, вывести наименьшее из них.

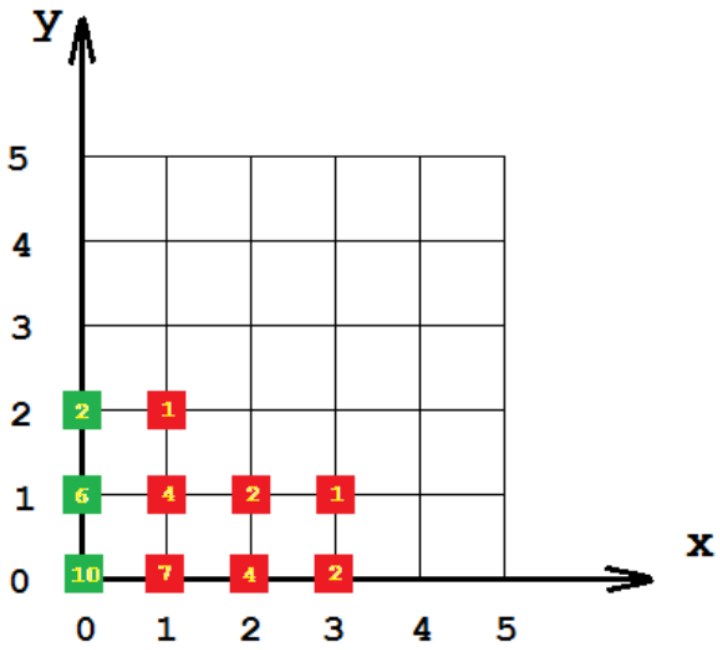
Примеры

Пример №1

Стандартный ввод
10

Стандартный вывод
4

Пояснение к примеру 1.



На рисунке представлена компоновка для $n = 10$ и $k = 4$. 10 регистров располагаются по 4 в ряд, последний ряд неполный. Для понимания условия на рисунке красным выделены те регистры, на которые воздействует регистр $(1, 0)$. Для каждого регистра желтым указано, на сколько регистров он воздействует. Даная компоновка является оптимальной для $n = 10$, так как в ней достигается минимальное возможное суммарное количество зависимостей — 39.

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

ll S(ll n){
    return (1LL + n) * n / 2;
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    ll n, mn = 1e18, ans;
    cin >> n;

    for(ll k = 1; k <= n; k++){
        ll v = n / k + (n % k > 0);
        ll d = S(k)*S(n/k) + S(v)*S(n%k) - S(n/k)*S(n%k);

        if(mn > d){
            mn = d;
            ans = k;
```

```
        }
    }
    cout << ans;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
n = int(input())
control = 10**20
ans = 1
for k in range(1, n + 1):
    nn = n // k
    f = n - n // k * k
    if (control > nn * (nn + 1) // 2 * k * (k + 1) // 2 + f * (f + 1) // 2 + f * (f + 1) // 2 * nn):
        control = nn * (nn + 1) // 2 * k * (k + 1) // 2 + f * (f + 1) // 2 +  nn  * f * (f + 1) // 2
        ans = k
print(ans)
```

Задача III.1.1.4. Прямоугольное хеширование (25 баллов)

В задачах на быструю обработку больших объемов данных важную роль играет хеширование. Если в общих словах, то хеширование — это некоторый способ поставить каждому объекту некоторое достаточно случайное, быстро вычислимое число (хеш) из относительно не очень большого интервала (например от 0 до 10^{18}). Под объектами могут пониматься любые, доступные для обработки, например записи в базе данных, строки, матрицы и т. п. При этом обратного восстановления по хешу исходного объекта не требуется. Нужно только, чтобы для двух идентичных объектов метод всегда выдавал одинаковые хеши, а для двух разных очень желательно, чтобы метод выдавал различные хеши. Второе требование на самом деле не может быть выполнено в полном объеме, так как количество больших, например матриц, заведомо больше 10^{18}. Но так как обработать мы можем не очень большое количество таких матриц, то вероятность совпадения хеша у разных объектов (коллизии) при правильном методе построения хеша очень мала.

В данной задаче потренируемся быстро строить хеши для достаточно больших матриц, состоящих из однозначных чисел от 0 до 9. Выберем три простых числа P_1, P_2 и P в пределах от 2 до 10^{18}.

Далее будем считать, что нумерация строк от 0 до $n-1$, нумерация столбцов от 0 до $m-1$. Пусть нам дана прямоугольная матрица A из n строк и m столбцов, на пересечении i-ой строки и j-го столбца в которой стоит число $A_{i,j}$. Тогда поставим этому числу на этой позиции в соответствие другое число $Z_{i,j} = (A_{i,j} \cdot P_1^i \cdot P_2^j) mod\, P$, где mod — это остаток от деления. Тогда произвольной подматрице исходной матрицы A поставим в соответствие сумму $Z_{i,j}$ по всем позициям в этой подматрице. Сумма также берется по модулю P. Однако если две одинаковые подматрицы находятся в разных позициях, то эти суммы для них все еще будут различны. Это можно скорректировать, если каждую «сдвинуть» в правый нижний угол, домножив сумму на некоторые степени P_1 и P_2. Более точно: пусть левый верхний угол подматрицы

размера $h \times w$ ($h \leqslant n$ строк и $w \leqslant m$ столбцов) находится в x-ой строке и y-м столбце. Тогда хеш этой подматрицы вычислим как сумму по всем ее $Z_{i,j}$, домноженную на P_1^{n-h-x} и на P_2^{m-w-y} по модулю P. Для лучшего понимания процесса смотрите примеры.

Основная сложность этой задачи (помимо сложности самого условия) заключается в том, что хеш нужно находить для большого числа достаточно больших подматриц, то есть в больших тестах каждый раз пробежаться по подматрице и посчитать ее хеш по определению не получится.

Формат входных данных

В первой строке находятся через пробел три различных простых числа P_1, P_2 и P, каждое в пределах от 2 до 10^{18}. Во второй строке находятся через пробел два числа n и m — число строк и столбцов исходной матрицы A ($1 \leqslant n, m \leqslant 500$). В следующих n строках содержится по m цифр от 0 до 9 (цифры записаны без пробелов между ними) — содержимое матрицы. В последней строке после матрицы содержатся через пробел два числа h и w ($1 \leqslant h \leqslant n$, $1 \leqslant w \leqslant m$) — размер искомых подматриц.

Формат выходных данных

Вывести $n - h + 1$ строк по $m - w + 1$ чисел через пробел в каждой — хеши всех подматриц размера $h \times w$. Хеш каждой матрицы располагается в соответствии с расположением ее левого верхнего угла.

Пояснения к примеру

Пояснение к примерам. Можно заметить, что в примере 1 есть четыре подматрицы 2×3, среди них две одинаковые, имеющие вид:

504

047

Соответственно, для этих двух одинаковых подматриц их хеши совпадают и равны 1605. Для попарно различных матриц хеши не совпадают.

Во втором примере есть две одинаковые подматрицы:

11011

01110

с левыми верхними углами в $(0, 0)$ и $(1, 2)$ у которых хеши равны 83793 и еще есть три одинаковые подматрицы:

10111

11101

с левыми верхними углами в $(0, 1)$, $(2, 1)$ и $(3, 3)$ у которых хеши равны 399453.

Примеры

Пример №1

Стандартный ввод
137 157 2027 3 4 3504 5047 0476 2 3
Стандартный вывод
1346 1605 1605 1451

Пример №1

Стандартный ввод
16769023 22447 1046527 5 8 11011110 01110110 01011101 11110111 00111101 2 5
Стандартный вывод
83793 399453 907550 997749 824324 433971 83793 429774 542305 399453 937871 433971 387715 816688 197340 399453

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;

ll M1, M2, M;

vector<ll> S1, S2;
vector<string> v;
vector<vector<ll> > H;
int n, m, h, w;

inline ll binprod (ll a, ll n) {
    ll res = 0LL;
    while (n) {
            res = (res + (n%2) * a) % M;
        a = (2LL * a) % M;
```

```
        n /= 2;
    }
    return res;
}

inline ll hsh(int x1, int y1, int x2, int y2){
    ll r = H[x2][y2];
    if(x1 > 0) r = (r + (M -  H[x1-1][y2])) % M;
    if(y1 > 0) r = (r + (M -  H[x2][y1-1])) % M;
    if(x1 > 0 && y1 > 0) r = (r + H[x1-1][y1-1]) % M;

    return r;
}

inline ll hrt(int x1, int y1, int h, int w){
    int x2 = x1 + h - 1;
    int y2 = y1 + w -1;
    ll ty =  hsh(x1, y1, x2, y2);
    ty = binprod(ty, S1[n - h - x1]);
    ty = binprod(ty, S2[m - w - y1]);

    return ty;
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cin >> M1 >> M2 >> M;
    cin >> n >> m;
    v.resize(n);
    for(int i = 0; i < n; i++) cin >> v[i];
    cin >> h >> w;

    ll st = 1;
    for(int i = 0; i < 510; i++){
        S1.push_back(st);
        st = binprod(st, M1);
    }

    st = 1;
    for(int i = 0; i < 510; i++){
        S2.push_back(st);
        st = binprod(st, M2);
    }

    H.resize(n, vector<ll>(m));
    for(int i = 0; i < n; i++)
        for(int j = 0; j < m; j++){
            H[i][j] = binprod(binprod((v[i][j] - '0'), S1[i]),  S2[j]);
            if(i > 0) H[i][j] = (H[i][j] + H[i-1][j]) % M;
            if(j > 0) H[i][j] = (H[i][j] + H[i][j-1]) % M;
            if(i > 0 && j > 0) H[i][j] = (H[i][j] + (M - H[i-1][j-1])) % M;
    }

    for(int i = 0; i < n - h + 1; i++){
        for(int j = 0; j < m - w + 1; j++)
        cout<<hrt(i, j, h, w)<<' ';
        cout<<endl;
    }
}
```

Задача III.1.1.5. Наибольшая перевернутая подстрока (35 баллов)

Применим теперь идеи, упомянутые в предыдущей задаче.

Дана строка, состоящая из маленьких латинских букв. Нужно найти наибольшую по длине ее подстроку, которая встречается как в прямом, так и в перевернутом виде. Эти две подстроки могут пересекаться и даже совпадать.

Формат входных данных

На вход подается единственная непустая строка, состоящая из маленьких латинских букв от a до z.

Длина строки не превосходит 10^5 символов.

Формат выходных данных

Вывести подстроку исходной строки, такую, что ее перевернутый вариант также является подстрокой, при этом длина этой подстроки максимально возможная. Если различных максимальных вариантов несколько, вывести самую левую такую подстроку.

Пояснение к примерам.

В примере 1 есть две подстроки длины 5, которые содержатся в исходной строке как в прямом, так и в обратном виде. Это подстрока «aobab», и подстрока «obabo». Подстрок большей длины с такими свойствами нет. Первая из них встречается раньше второй, поэтому она и будет ответом.

В примере 2 единственной подстрокой длины 7 с указанным свойством является палиндром «aobaboa».

Примеры

Пример №1

Стандартный ввод
`baobabobbaboa`
Стандартный вывод
`aobab`

Пример №2

Стандартный ввод
`baobabaobaboa`
Стандартный вывод
`aobaboa`

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;

ll P1 = 1073676287;
ll P = 1125899839733759;
vector<ll> H, S1, RH;
string s;
int n;

inline ll binprod (ll a, ll n) {
    ll res = 0LL;
    while (n) {
            res = (res + (n%1024) * a) % P;
        a = (1024LL * a) % P;
        n /= 1024;
    }
    return res;
}

inline ll hsh(int x1, int x2, vector<ll> &H){
    ll r = H[x2];
    if(x1 > 0) r = (r + (P -  H[x1-1])) % P;

    return r;
}

inline ll hsub(int x1, int w, vector<ll> &H){
    int x2 = x1 + w - 1;
    ll ty =  hsh(x1, x2, H);
    ty = binprod(ty, S1[n - w - x1]);

    return ty;
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cin >> s;
    n = s.size();
    string rs = s;
    reverse(rs.begin(), rs.end());

    ll st = 1;
    for(int i = 0; i < n; i++){
        S1.push_back(st);
        st = binprod(st, P1);
    }

    H.resize(n);
    RH.resize(n);
    for(int i = 0; i < n; i++){
        H[i] = binprod((s[i] - 'a' + 1), S1[i]);
        if(i > 0) H[i] = (H[i] + H[i-1]) % P;

```

```
        RH[i] = binprod((rs[i] - 'a' + 1), S1[i]);
        if(i > 0) RH[i] = (RH[i] + RH[i-1]) % P;
    }

    int mxL = 1, rD = 0;
    int L = 1, R = n + 1;
    while(R - L > 1){
        ll M = (R + L) / 2;

        set<ll> mask;
        bool p = 0;
        for(int i = 0; i < n - M + 1; i++){
            mask.insert(hsub(i, M, RH));
        }

        for(int i = 0; i < n - M + 1; i++){
            ll y = hsub(i, M, H);

            if(mask.find(y) != mask.end()){
                p = 1;
                if(M > mxL){
                    mxL = M;
                    rD = i;
                }
                break;
            }
        }
        if(p) L = M; else R = M;
    }

    cout<<s.substr(rD, L);
}
```

Математика. 8–9 класс

Задача III.1.2.1. (20 баллов)

Найдите все многочлены $P(x)$, при любом x удовлетворяющие равенству:

$$P(2x) = \frac{P(3x) - P(x)}{2}$$

Решение

Пусть целое неотрицательное n – степень многочлена $P(x)$. Если $n = 0$, то $P(x) = c$ – константа, и единственное c, удовлетворяющее равенству в условии, равно 0.
Теперь рассмотрим $n > 0$: тогда $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, где $a_n \neq 0$. Тогда n должно удовлетворять

$$2^n = \frac{3^n - 1}{2},$$

откуда $n = 2$. Докажем, что $2^n < \frac{3^n-1}{2}$ для натуральных $n > 2$. Для этого докажем по индукции, что для тех же n выполнено $3^n - 2^{n+1} > 1$.
База индукции ($n = 3$): очевидно, $3^3 - 2^4 > 1$.

Шаг индукции: пусть для некоторого натурального $k > 2$ выполнено $3^k - 2^{k+1} > 1$. Докажем это для $k + 1$. Действительно:

$$3^k > 2^k \Rightarrow 2 \cdot 3^k > 2^{k+1} \Rightarrow 3^{k+1} - 3^k > 2^{k+2} - 2^{k+1} \Rightarrow 3^{k+1} - 2^{k+2} > 3^k - 2^{k+1} > 1,$$

что завершает доказательство.

Значит, $3^n - 2^{n+1} > 1 \Rightarrow 2^n < \frac{3^n-1}{2}$.
Итак, $P(x) = ax^2 + bx + c$. Подставив это выражение в равенство в условии задачи, получим $b = c = 0$, при этом a – любое, включая 0 (так мы включим в ответ случай $P(x) = 0$).

Ответ: $P(x) = ax^2$ для любого a.

Задача III.1.2.2. (35 баллов)

Правильный шестиугольник Г получается поворотом шестиугольника Γ_0 на 30° вокруг его центра. Найдите вероятность того, что точка, случайным образом размещенная внутри Γ_0, окажется внутри Г.

Решение

Искомая вероятность ввиду случайности расположения упомянутой точки равна $P = \frac{S_0}{S}$, где S – площадь Г, а S_0 – общая площадь Г и Γ_0. Пусть сторона шестиугольника равна a, тогда $S = \frac{3\sqrt{3}a^2}{2}$ (делим шестиугольник на шесть равных правильных треугольников со стороной a).

После поворота вершины Γ_0 окажутся вне Г вместе с шестью равнобедренными (ввиду симметрии относительно наибольших диагоналей) треугольниками, составляющими часть Γ_0. Пусть основание такого треугольника равно y, а боковая сторона – x. Тогда угол между основанием и боковой стороной равен 30° (угол поворота) и $2x + y = a$ ($y = x\sqrt{3}$). Высота такого треугольника равна половине боковой стороны, а площадь равна:

$$S_\triangle = \frac{x^2\sqrt{3}}{4} = \frac{a^2 \cdot (2 - \sqrt{3})^2\sqrt{3}}{4}.$$

Значит:

$$S_0 = S - 6 \cdot S_\triangle \Rightarrow P = 1 - 6 \cdot \frac{S_\triangle}{S} = 4\sqrt{3} - 6.$$

Ответ: $4\sqrt{3} - 6$.

Задача III.1.2.3. (45 баллов)

Пусть сформулирован признак делимости на некоторое натуральное $n > 1$, выражающийся через некоторые свойства цифр числа и выполняющийся при любой перестановке цифр числа. Найдите все возможные n.

Решение

Очевидно, n – не степень десятки. Значит, одно из чисел $2n, \ldots, 9n$ начинается с 1. Поменяв эту единицу местами с последней цифрой числа, мы должны получить число, кратное n. Значит, n не делится ни на 2, ни на 5.

Поскольку уравнение $ax + by = c$ при $(a, b) = 1$ имеет решение в целых числах, уравнение $100x - ny = 12$ имеет решение в натуральных числах (одно из следствий соотношения Безу). Это значит, что некоторое число, оканчивающееся на $\ldots 12$, делится на n. Поменяв местами последние две цифры, получим, что $21 - 12 = 9$ – делится на n, откуда $n = 3$ или $n = 9$.

Ответ: $3; 9$.

Математика. 10–11 класс

Задача III.1.3.1. (20 баллов)

Дан $\triangle ABC$. Точка A_1 симметрична точке A относительно B, точка B_1 симметрична B относительно C, точка C_1 симметрична C относительно A. Найдите отношение площадей треугольников $A_1B_1C_1$ и ABC.

Решение

Введем обозначения: $a = |BC|, b = |AC|, c = |AB|, \alpha = \angle BAC, \beta = \angle CBA$, $\gamma = \angle ACB$. Тогда $S_{ABC} = \frac{1}{2}ab\sin\gamma = \frac{1}{2}bc\sin\alpha = \frac{1}{2}ca\sin\beta$.

Согласно условию и определению симметрии относительно точки, B – середина отрезка AA_1, C – середина отрезка BB_1, A – середина отрезка CC_1.

$$S_{C_1AA_1} = \frac{1}{2}\cdot|AC_1|\cdot|AA_1|\cdot\sin\angle A_1AC_1 = \frac{1}{2}\cdot b\cdot 2c\cdot\sin\alpha = bc\sin\alpha = 2\cdot S_{ABC}.$$

Аналогично,

$$S_{A_1BB_1} = ac\sin\beta = 2\cdot S_{ABC} \text{ и } S_{B_1CC_1} = ab\sin\gamma = 2\cdot S_{ABC}.$$

$$S_{A_1B_1C_1} = S_{ABC} + S_{C_1AA_1} + S_{A_1BB_1} + S_{B_1CC_1} = 7\cdot S_{ABC}.$$

Откуда получаем требуемое отношение площадей.

Ответ: 7.

Задача III.1.3.2. (35 баллов)

Решите уравнение:

$$[x + x\cdot[x]] = 2021$$

$[x]$ – *наибольшее целое число, не превосходящее* x.

Решение

$$[x \cdot ([x] + 1)] = 2021$$

Очевидно, 0 не является решением уравнения. Если $x < 0$, то $[x] < 0$ и $[x] + 1 \leqslant 0$: при $x < -45$ имеем $[x \cdot ([x]+1)] > 2025$, при $-45 \leqslant x < 0$ получим $[x \cdot ([x]+1)] \leqslant 1980$.

Теперь рассмотрим $x > 0$. Тогда возможны следующие случаи:

1) $x \geqslant 45 \Rightarrow x \cdot ([x] + 1) > 45^2 > 2022$;

2) $x < 44 \Rightarrow x \cdot ([x] + 1) < 44^2 < 2021$;

3) $44 \leqslant x < 45 \Rightarrow x = 44 + \beta$ для $\beta \in [0; 1)$.

Значит:

$$[(44 + \beta) \cdot 45] = [1980 + 45\beta] = 2021.$$

Откуда:

$$\frac{41}{45} \leqslant \beta < \frac{14}{15} \Rightarrow x \in [44\frac{41}{45}; 44\frac{14}{15}).$$

Ответ: $x \in [44\frac{41}{45}; 44\frac{14}{15})$.

Задача III.1.3.3. (45 баллов)

Последовательность $\{a_n\}_{n\geqslant 0}$ задается рекуррентным соотношением $a_k = 2a_{k-1} - 1$ для всякого натурального k, причем $a_0 = p$, где $p > 2$ – простое число. Может ли любой член этой последовательности быть простым числом?

Решение

Докажем по индукции, что $a_n = 2^n \cdot (p - 1) + 1$ для всех натуральных n.

База индукции: $a_1 = 2a_0 - 1 = 2^1(a_0 - 1) + 1$.

Шаг индукции: пусть для некоторого натурального k выполнено $a_k = 2^k(p-1)+1$. Докажем, что $a_{k+1} = 2^{k+1}(p - 1) + 1$. Действительно,

$$a_{k+1} = 2a_k - 1 = 2 \cdot (2^k(p - 1) + 1) - 1 = 2^{k+1}(p - 1) + 1,$$

что и требовалось доказать.

Итак, $a_n = 2^n \cdot (p - 1) + 1$ для всех натуральных n.

Согласно малой теореме Ферма, $a^{p-1} \equiv 1 \pmod p$ для простого p и натурального a, не кратного ему. Очевидно, 2 не кратно простому $p > 2$ – значит,

$$2^{p-1} \equiv 1 \pmod p \Rightarrow 2^{p-1}(p - 1) \equiv p - 1 \pmod p \Rightarrow 2^{p-1}(p - 1) + 1 \equiv 0 \pmod p.$$

Т. е. a_{p-1} кратно p и (поскольку $a_{p-1} > a_{p-2} > \ldots > a_0 = p \Rightarrow a_{p-1} > p$) не может быть простым.

Ответ: Нет, не может.