9-11 класс

1. «Прогулка по магазинам» (100 баллов)

Галя может сразу выйти к первому магазину, который она хочет посетить на улице, и посетить K магазинов подряд. При этом она пройдет расстояния между этими магазинами. Промежутков между K магазинами будет K – 1. Задача сводится к поиску суммы $K-1$ подряд идущих элемента массива.

Поскольку расстояние, пройденное Галей по улице, должно быть минимальным, то она выйдет сразу к нужному магазину, а после посещения последнего уйдет с улицы. a_0 и a_{N+1} учитывать при нахождении суммы не нужно. По этим участкам Галя точно может не проходить.

Можно для каждого начального элемента каждый раз явно суммировать K – 1 элемент и среди полученных сумм искать минимальную. Такое решение может получить 80 баллов.

```
n = int(input())
a = []
for i in range(n + 1):
a.append(int(input()))
k = int(input())
sminn = 1000 * 1000000
for i in range(1, n + 1 - k + 1):
    s = 0
    for j in range(i, i + k - 1):
        s += a[j]
sminn = min(sminn, s)
print(sminn)
```

Сложность данного алгоритма O(N^2).

Можно двумя вложенными циклами перебирать номера первого и последнего посещенных Галей магазинов. Для каждой пары нужно проверять условие, что они отличаются на $K-1$. После этого третьим вложенным циклом суммировать элементы между этими индексами, включая элементы с выбранными индексами. Среди найденных сумм искать минимальную. Сложность данного алгоритма O(N^3). Данное решение может набрать 60 баллов.

Полный балл получит решение, которое имеет сложность O(N). Найдем сумму

$a_1+ a_2 + … + a_{K-1}$ элементов. Чтобы из этой суммы получить сумму $a_2+ a_3 + … + a_K$ элементов достаточно из предыдущей суммы вычесть a_1 и добавить a_K. Аналогичным образом можно, скорректировав предыдущую сумму $K-1$ элементов, получить сумму следующих $K-1$ элементов. Среди всех найденных сумм найдем минимальную.

```
n = int(input())
a = []
for i in range(n + 1):
a.append(int(input()))
```

```
k = int(input())


s = sum(a[1: k])
sminn = s
for i in range(1, n - k + 1):
    s -= a[i]
    s += a[i + k - 1]
sminn = min(sminn, s)
print(sminn)
```

Номергруппы	Номера тестов	Баллы загруппу тестов	Баллы за тест
0 (тесты из условия)	01	0	0
1	02-13	60	5
2	14-17	20	5
3	18-21	20	5

2. «Букет в подарок» (100 баллов)

Баллы за первые две группы тестов может набрать переборное решение. В двух вложенных циклах будем перебирать все возможные варианты для количества роз и хризантем в букете. Для каждого варианта будем проверять, чтобы количество роз было кратно *a*, а количество хризантем было кратно *b* или наоборот, количество роз было кратно *b*, а количество хризантем кратно *a*. Также будем проверять, что общее количество цветов в букете нечетное. Из всех подходящих вариантов выберем максимальное количество цветов в букете. Если ни один вариант не подошел, то выведем «0 0».

```
n = int(input())
m = int(input())
a = int(input())
b = int(input())
maxx = 0
for r in range(1, n + 1):
  for h in range(1, m + 1):
    if r % a==0 and h % b == 0 or r % b == 0 and h % a==0:
            if (r + h) % 2 != 0 and r + h >maxx:
maxx = r + h
```

```
                rm = r
                hm = h
if maxx == 0:
    rm = 0
    hm = 0
print(rm, hm)
```

Сложность данного алгоритма O(N^2).

Для прохождения третьей группы решений необходимо проверить несколько условий, не перебирая все возможные варианты. Если продается цветов меньше, чем любимые числа девушки, то составить букет невозможно и выведем «0 0». Найдем, какое максимальное количество раз мы можем добавить в наш букет по *a* или *b* роз (*kra* и *krb*), а также по *a* или *b* хризантем (*kha* и *khb*). Сначала попробуем составить букет из максимального количества роз, кратного *a*, и максимального количества хризантем, кратного *b* (*kra * a + khb * b*). Если мы получим четное количество цветов в букете, то можно попробовать взять на *a* роз меньше ((*kra* – 1) ** a + khb * b*) или на *b* хризантем меньше (*kra * a +* (*khb* –1) ** b*). Из трех перечисленных выше вариантов будем искать максимальный с нечетным количеством цветов. Поскольку можно взять количество роз, кратное *b*, и количество хризантем, кратное *a*, то мы получим еще три симметричных варианта: (*krb * b + kha * a*), ((*krb* – 1) ** b + kha * a*), (*krb * b +* (*kha* – 1) ** a*). Если во всех мы получаем четное количество цветов, то составить букет невозможно. В противном случае из всех найденных значений ищем максимальное нечетное. Для найденного значения запоминаем количество роз и хризантем.

```
n = int(input())
m = int(input())
a = int(input())
b = int(input())
if n <min(a, b) or m < min(a, b):
print(0, 0)
else:
kra = n // a
krb = n // b
    kha = m // a
khb = m // b
maxx = 0
    k = kra * a + khb * b
    if k % 2 != 0 and k >maxx:
maxx = k
mr = kra * a
mh = khb * b
```

```
    k = (kra - 1) * a + khb * b
    if kra> 1 and k % 2 != 0 and k >maxx:
maxx = k
mr = (kra - 1) * a
mh = khb * b
    k = kra * a + (khb - 1) * b
    if khb> 1 and k % 2 != 0 and k >maxx:
maxx = k
mr = kra * a
mh = (khb - 1) * b

    k = krb * b + kha * a
    if k % 2 != 0 and k >maxx:
maxx = k
mr = krb * b
mh = kha * a
    k = (krb - 1) * b + kha * a
    if krb> 1 and k % 2 != 0 and k >maxx:
maxx = k
mr = (krb - 1) * b
mh = kha * a
    k = krb * b + (kha - 1) * a
    if kha > 1 and k % 2 != 0 and k >maxx:
maxx = k
mr = krb * b
mh = (kha - 1)  * a
    ifmaxx == 0:
mr = 0
mh = 0
print(mr, mh)
```

Сложность данного решения O(1).

Разбиение тестов по группам

Номер группы	Номера тестов	Баллы за группу тестов	Баллы за тест

0 (тесты из условия)	01-02	0	0
1	03-04	10	5
2	05-14	50	5
4	15-22	40	5

3. «Проверка последовательности» (100 баллов)

Для прохождения первой группы тестов можно просто написать циклический алгоритм, который вычисляет цифры последовательности по данному правилу.

```
a = int(input())
b = int(input())
n = int(input())


for i in range(3, n + 1):
    c = (a ** 2 + b ** 2) % 10
    a, b = b, c
print(c)
```

Для получения полного балла за задачу такой алгоритм уже не годится, поскольку потребует выполнения слишком большого количества вычислений. Рассмотрим каждые две цифры нашей последовательности, идущие подряд, как одно двухзначное число. Поскольку двухзначных чисел существует не более 100, то среди первых 100 чисел последовательности обязательно найдутся хотя бы два одинаковых. Выберем два такие совпадающих числа и обозначим их номера за n_1 и n_2. По правилам построения нашей последовательности понятно, что она является периодической с периодом $n_2 - n_1$. Причем первые n_1–1 чисел не попадают в период.

Пусть последовательность будет начинаться с 2 и 7. Тогда последовательность цифр будет выглядеть так: 2 7 3 8 3 3 8 и т.д.

Последовательность двухзначных чисел: 27 73 38 83 33 38.

Нашлось два одинаковых числа 38, которые стоят в последовательности двухзначных чисел под номерами $n_1 = 3$, $n_2 = 6$. Период будет 6 – 3 = 3. Первые два числа не входят в период, а следующие три числа будут повторяться.

Теперь нам осталось представить номер N в виде $n_1-1+(n_2-n_1)*N+r$, где $0 \le r < n_2-n_1$, и определить число с номером n_1-1+r, совпадающее с искомым числом.

Например, нам нужно выяснить, какое число стоит под номером 200. Уберем из номера два первых числа: 200 – 2 = 198 и найдем остаток от деления на 3. Это будет 0. Значит, искомое число будет равно 3.

```
a = int(input())
b = int(input())
n = int(input())


if n <= 100:
```

```
    for i in range(3, n + 1):
        c = (a ** 2 + b ** 2) % 10
        a, b = b, c
    print(c)
else:
    z = [a * 10 + b]
    for i in range(3, 100):
        c = (a ** 2 + b ** 2 ) % 10
zn = b * 10 + c
        if zn in z:
            n1, n2 = z.index(zn), i - 2
            break
z.append(zn)
        a, b = b, c


    p = n2 - n1
    z = z[n1:]
    z = [z[-1]] + z[:-1]
    if n1 > 0:
        n -= n1
    n = (n) % p
    print(z[n] // 10)
```

Разбиение тестов по группам

Номергруппы	Номера тестов	Баллы загруппу тестов	Баллы за тест
0 (тесты из условия)	01-02	0	0
1	03-14	60	5
2	15-18	20	5
3	19-22	20	5

4. «Яся продаёт рыбок» (100 баллов)

Для прохождения первой подгруппы тестов можно написать полный перебор.

Во второй подгруппе продавец один и он сможет обслуживать a_1 покупателей одновременно, поэтому если $k \le a_1$, направим всех покупателей к нему. В противном случае ответ -1.

В третьей подгруппе каждый продавец может обслуживать только одного покупателя. Будем поддерживать список свободных продавцов, а также массив z_1, z_2.., z_n, в котором z_i равен номеру продавца, который освободится к моменту времени i. Сначала проверим, есть ли освободившийся продавец. Если есть, отправим его обслуживать j-того покупателя. Если нет, попробуем взять нового продавца из списка. Если список пустой, ответ -1. Когда продавец зафиксирован за покупателем, запишем номер текущего продавца в z_{j+k}.

Чтобы решить задачу на полный балл, надо использовать идею третьей подгруппы, то есть честно моделировать процесс. Заметим, что Яся оставит $n - p$ продавцов с **максимальной** пропускной способностью. Поэтому отсортируем всех продавцов по невозрастанию их пропускной способности. Теперь разместим продавцов в список таким образом, чтобы сначала Яся брала продавца с максимальной пропускной способностью. Будем удалять продавца из списка тогда, когда он начинает обслуживать ровно a_i покупателей.

Асимптотика полного решения $O(n \cdot \log n)$.

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

#define X first
#define Y second

using namespace std;

typedef pair<int, int>ipair;

const int MAXN = 200200;
const int MAXM = MAXN;

int n, m, k;
ipairarr[MAXN];
ipair* fre[MAXM << 1];
bool used[MAXN];
set<ipair*> NZ;


inline bool comp(const ipair& a, const ipair& b) {
    return a.X>b.X;
}
```

```
void build() {
stable_sort(arr, arr + n, comp);
    for (int i = 0; i< n; ++i)
NZ.insert(arr + i);
}

int solve() {
    for (int j = 0; j < m; ++j) {
        if (fre[j]) {
fre[j + k] = fre[j];
            continue;
        }
        if (NZ.empty()) return -1;
        auto it = *NZ.begin();
        it->X--;
        if (!it->X) NZ.erase(it);
fre[j + k] = it;
        used[it->Y] = true;
    }
    return n - count(used, used + n, true);
}

int main() {
cin>> n >> m >> k;
    for (int i = 0; i< n; ++i) {
cin>>arr[i].X;
arr[i].Y = i;
    }
build();
    int ans = solve();
cout<<ans<<endl;
    if (ans == -1) return 0;
    for (int i = 0; i< n; ++i)
        if (!used[i]) cout<<i + 1 << ' ';
```

```
cout<<endl;
    for (int j = 0; j < m; ++j)
cout<<fre[j + k]->Y + 1 << ' ';
cout<<endl;
}
```

Разбиение тестов по группам

Номер группы	Номера тестов	Баллы за группу тестов	Баллы за тест
0 (тесты из условия)	01	0	0
1	02-06	25	5
2	07-08	10	5
3	09-12	20	5
4	13-21	45	5

5. «Горилла и скобочная последовательность»(100 баллов)

Давайте заменим открывающуюся скобку на 1, а закрывающуюся на -1. Посчитаем сумму этих 1 и -1 по всей строке, получим некоторое число b. Очевидно, что $|b|$ скобок мешают последовательности стать **почти правильной**, давайте вставим $|b|$ противоположных им скобок.

В первой и третьей подгруппе достаточно написать наивное решение. То есть честно заменять скобки в каждом запросе, после чего пересчитывать b и выводить в качестве ответа $|b|$.

Если не пересчитывать b каждый раз, а обновлять его только для изменяемых элементов отрезка запроса, вторая и четвёртая подгруппы решаются аналогично.

Для решения пятой подгруппы напишем дерево отрезков для сумм на нашем массиве из 1 и -1. Чтобы выполнить операцию на отрезке, надо умножить каждое число на этом отрезке на -1. Такую операцию можно выполнять отложенно, используя дерево отрезков с массовыми операциями. Чтобы получить ответ, надо выполнить push из корня и взять модуль значения суммы в нём.

Асимптотика полного решения $O(n + q \cdot \log n)$.

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

using namespace std;

const int MAXN = 200200;

struct node {
    node *lv = nullptr, *rv = nullptr;
    int sum = 0;
```

```
    bool prom = false;
};

int n;
string s;
node mem[MAXN << 1];
node *mpos = mem, *root = nullptr;

void build(node* &v, int l, int r) {
    if (v == nullptr) v = mpos++;
    if (r - l == 1) {
        v->sum = (s[l] == '(' ?1 : -1);
        return;
    }
    int mid = (l + r) >> 1;
    build(v->lv, l, mid);
    build(v->rv, mid, r);
    v->sum = v->lv->sum + v->rv->sum;
}

void push(node *v, int l, int r) {
    if (!v->prom) return;
    v->sum *= -1;
    if (r - l > 1) {
        v->lv->prom ^= 1;
        v->rv->prom ^= 1;
    }
    v->prom = false;
}

void inv_range(node *v, int l, int r, int ql, int qr) {
push(v, l, r);
    if (qr<= l || r <= ql) return;
    if (ql <= l && r <= qr) {
```

```
        v->prom = true;
push(v, l, r);
        return;
    }
    int mid = (l + r) >> 1;
inv_range(v->lv, l, mid, ql, qr);
inv_range(v->rv, mid, r, ql, qr);
    v->sum = v->lv->sum + v->rv->sum;
}

void build() {
build(root, 0, n);
}

int main() {
    int m; cin>> n >> m >> s;
build();
    while (m--) {
        int l, r; cin>> l >> r, --l;
inv_range(root, 0, n, l, r);
push(root, 0, n);
cout<< abs(root->sum) << '\n';
    }
}
```

Разбиение тестов по группам

Номер группы	Номера тестов	Баллы за группу тестов	Баллы за тест
0 (тесты из условия)	01	0	0
1	02-04	12	4
2	05-09	20	4
3	10-11	8	4
4	12-16	20	4
5	17-26	40	4