Задача A. Раскраска

Автор задачи: Савинов Сергей

Разбор подготовили: Савинов Сергей, Тропин Даниил

Первый AC: Ярослав Чиж +1 (0:44)

Общее количество AC: 37

A. Раскраска

- Формула Эйлера (для несвязного графа):

$$В + Г - Р = К + 1$$

где В – количество вершин (константа в нашей задаче), Г – количество граней (требуемое число фигур), Р – количество ребер, К – количество компонент связности.

$$Г = -В + Р + К + 1$$

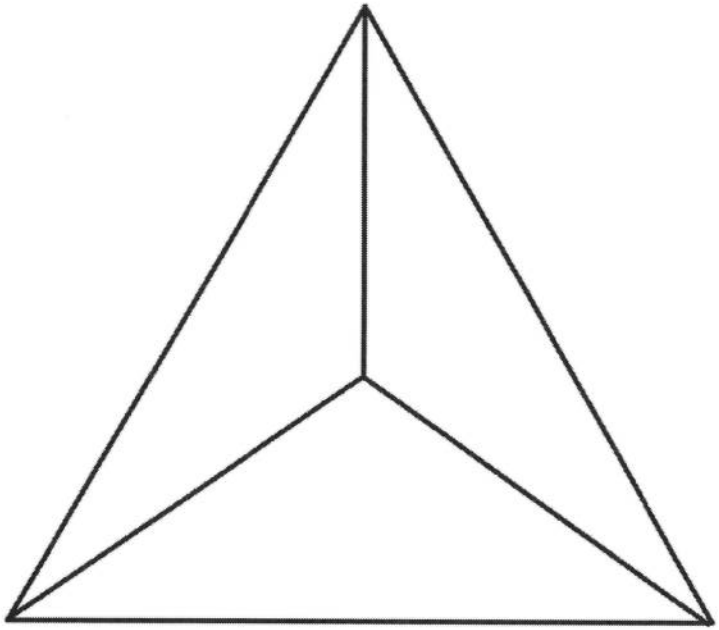
Необходимо максимизировать количество ребер и компонент связности.

A. Раскраска (продолжение)

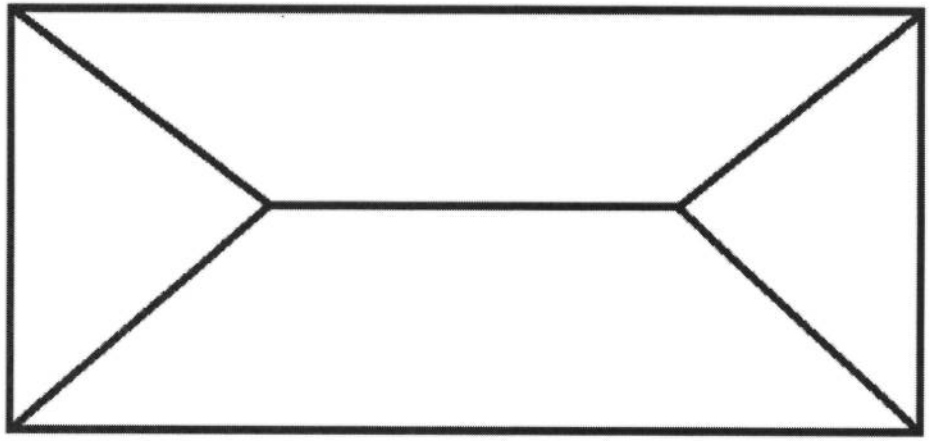
- Чтобы максимизировать число компонент связности, используем наименьшее возможное количество вершин в каждой.
- Для того, чтобы максимизировать количество ребер, добьемся полной «занятости» вершин. Т.к. максимальная степень вершины 3, то число вершин в компоненте связности должно быть четно.

$$P = \frac{3 \cdot B}{2}$$

A. Раскраска (продолжение)



N = 4



N = 6

A. Раскраска (продолжение)

- Жадно заполняем «4»-ми результирующий граф.
- При 6 оставшихся вершинах вставляем «6»-ку.
- В случае 3 оставшихся вершин, не ленимся рисовать треугольник.
- Не забываем обработать тривиальные остатки: 1, 2.
- Асимптотика O(N)

Задача B. Опционы

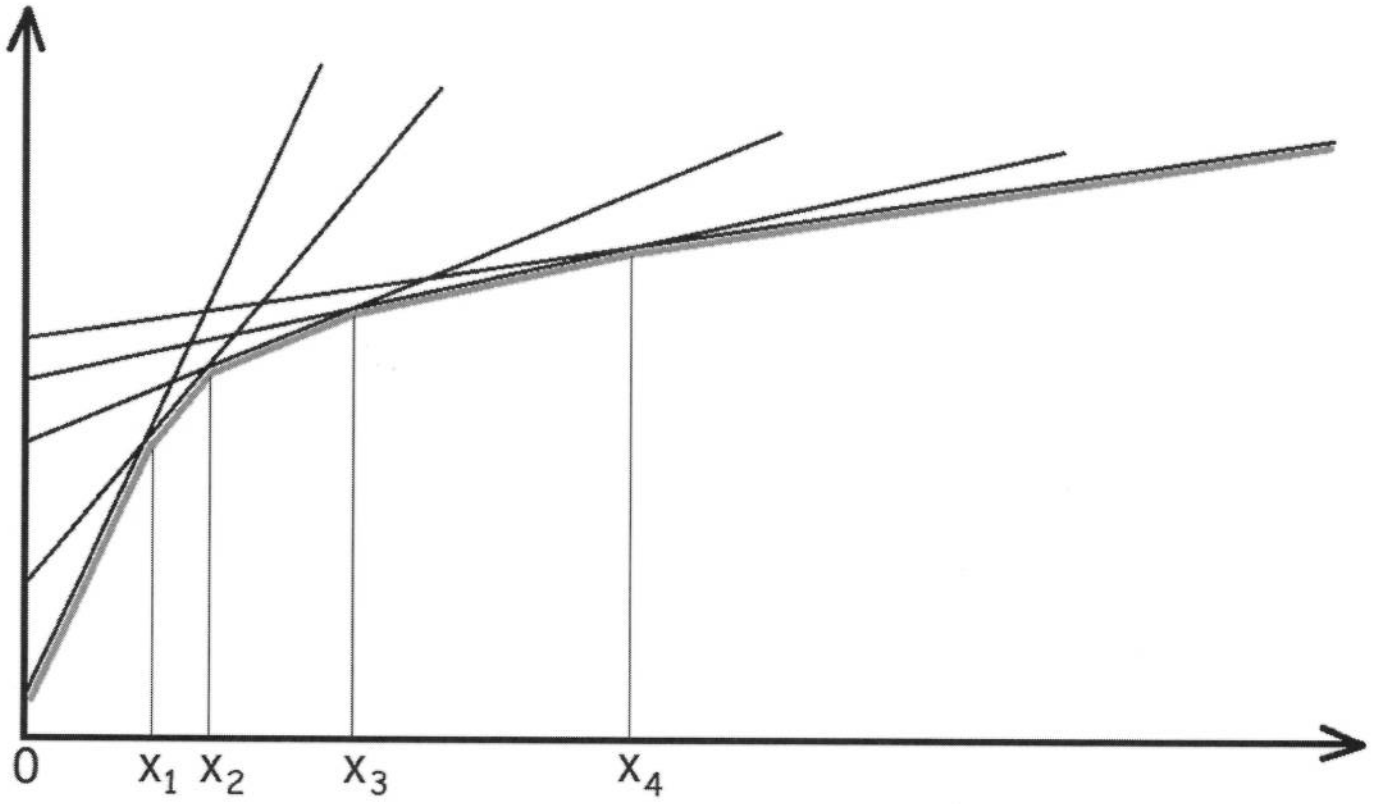
Автор задачи : Мамай Игорь

Разбор подготовил: Мамай Игорь

Первый AC: Станислав Наумов +4 (1:39)

Общее количество AC: 4

B. Опционы



B. Опционы (продолжение)

1. Поддерживаем выпуклую оболочку, порожденную прямыми.
2. Так как прямые отсортированы «плохо», то при добавлении новой прямой нужно вынуть пять последних прямых из оболочки и добавить их вместе с новой прямой в порядке убывания.
3. При ответе на запрос нужно бинарным поиском найти необходимый отрезок и дать ответ за O(1).

Асимптотика алгоритма O(n log(n)).

Задача C. Странная функция

Автор задачи : Мамай Игорь

Разбор подготовил: Мамай Игорь

Первый AC: Роман Коробков + (0:03)

Общее количество AC: 125

C. Странная функция

1.Нужно понять, что F(a, b) = a + b.

2.Ответ на задачу дается формулой:

$$\frac{1}{2}(a + b)(b - a + 1)$$

Асимптотика алгоритма O(1).

Задача D. Доменная печь 2

Автор задачи: Булатов Константин

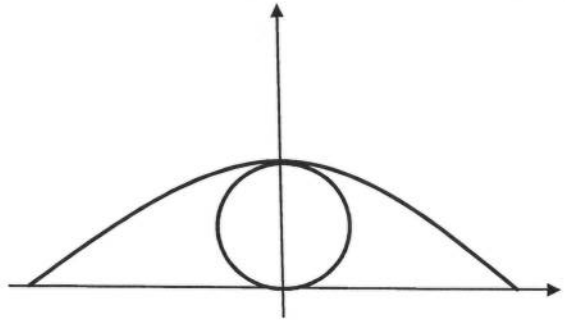
Разбор подготовил: Кодосов Никита

Первый AC: Михаил Путилин + (0:50)

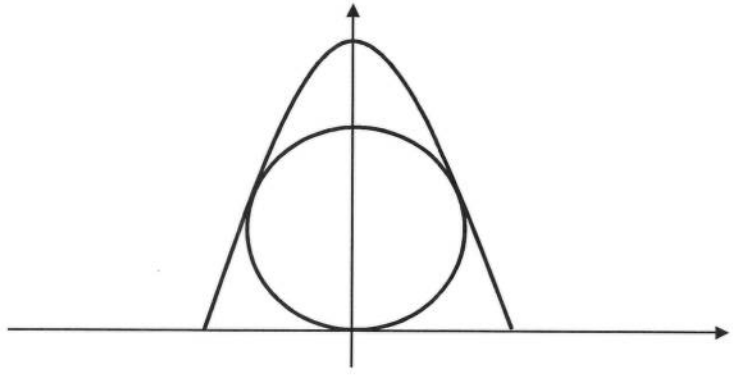
Общее количество AC: 71

D. Доменная печь 2

1. При $T \geq H$ ответ $\frac{H}{2}$.



2. Если $T < H$, то нужно выписать уравнения из условий касания окружности и параболы. Из уравнений выводится формула для радиуса $R = T - \frac{T^2}{2 \cdot H}$



Задача E. Стража

Автор задачи : Булатов Константин

Разбор подготовили: Булатов Константин, Мамай Игорь

Первый AC: Михаил Путилин +2 (2:10)

Общее количество AC: 1

E. Стража

1. Если треугольник вырожден, то для $n \leq 2$ ответ 1, а для $n \geq 3$ ответ 0.
2. Для всех невырожденных треугольников ответ одинаковый.
3. Можно применить метод Монте-Карло: достаточное количество раз независимо генерируем n равномерно распределенных внутри треугольника точек, проверяем их расположение на выпуклость. Как только процесс стабилизировался запоминаем ответ.

E. Стража (продолжение)

4. Генерировать случайную точку удобно для треугольника с координатами (0,0), (1,0), (0,1). Генерируем независимо две координаты $x \in [0,1]$ и $y \in [0,1]$ и проверяем, лежит ли точка (x, y) в треугольнике ($x + y \leq 1$). Если точка в треугольнике не лежит, то мы ее пропускаем.

5. Эта задача носит название «***Задача Сильвестра для треугольника***», для нее существует точная формула:

$$P(n) = \frac{2^n \cdot (3n - 3)!}{[(n - 1)!]^3 \cdot (2n)!}$$

Задача F. График совещаний

Автор задачи : Мамай Игорь

Разбор подготовил: Мамай Игорь

Первый AC: Денис Солонков +2 (1:26)

Общее количество AC: 8

F. График совещаний

1. Вычислим самые выгодные варианты сдвигов для всех возможных значений a, b, c, d и пяти возможных вариантов количества опечаток в день.
2. Будем заполнять динамику dp[i][j], где i – количество дней, j – количество опечаток, а dp[i][j] – минимальная сумма длин совещаний $1 \le i \le n$, $0 \le j \le 4n$.
3. Чтобы восстановить расписание делаем обратный проход. Для этого дополнительно в состояниях динамики нужно хранить информацию откуда пришли в это состояние.

Задача G. Игра Васи

Автор задачи: Булатов Константин

Разбор подготовил: Тропин Даниил

Первый AC: Баян Сайдолда + (0:10)

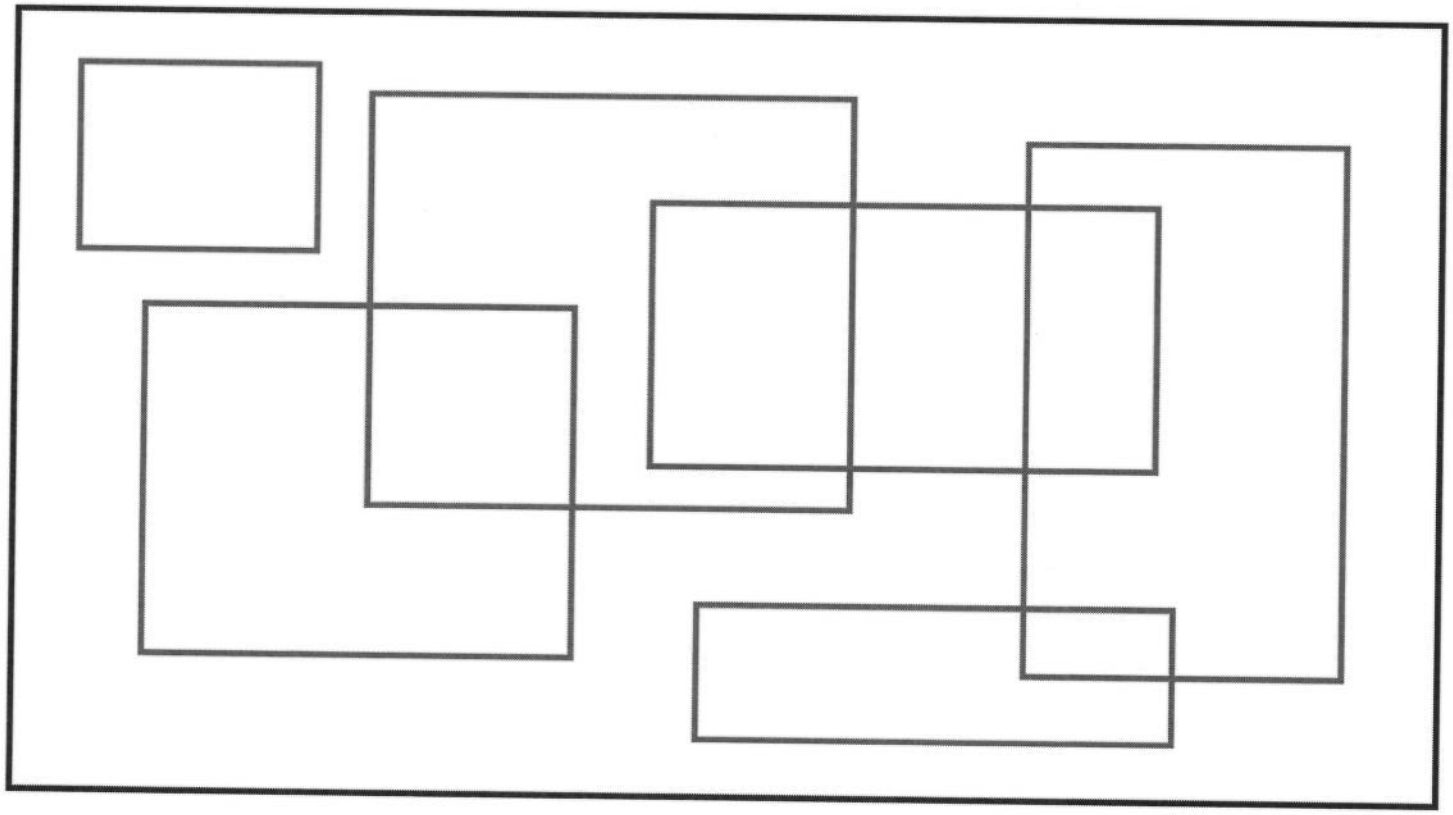
Общее количество AC: 120

G. Игра Васи

- Решение проходит в «оффлайн»-е.
- Полезно подумать о двумерных частичных суммах.
- Сложность O(NM + Q)

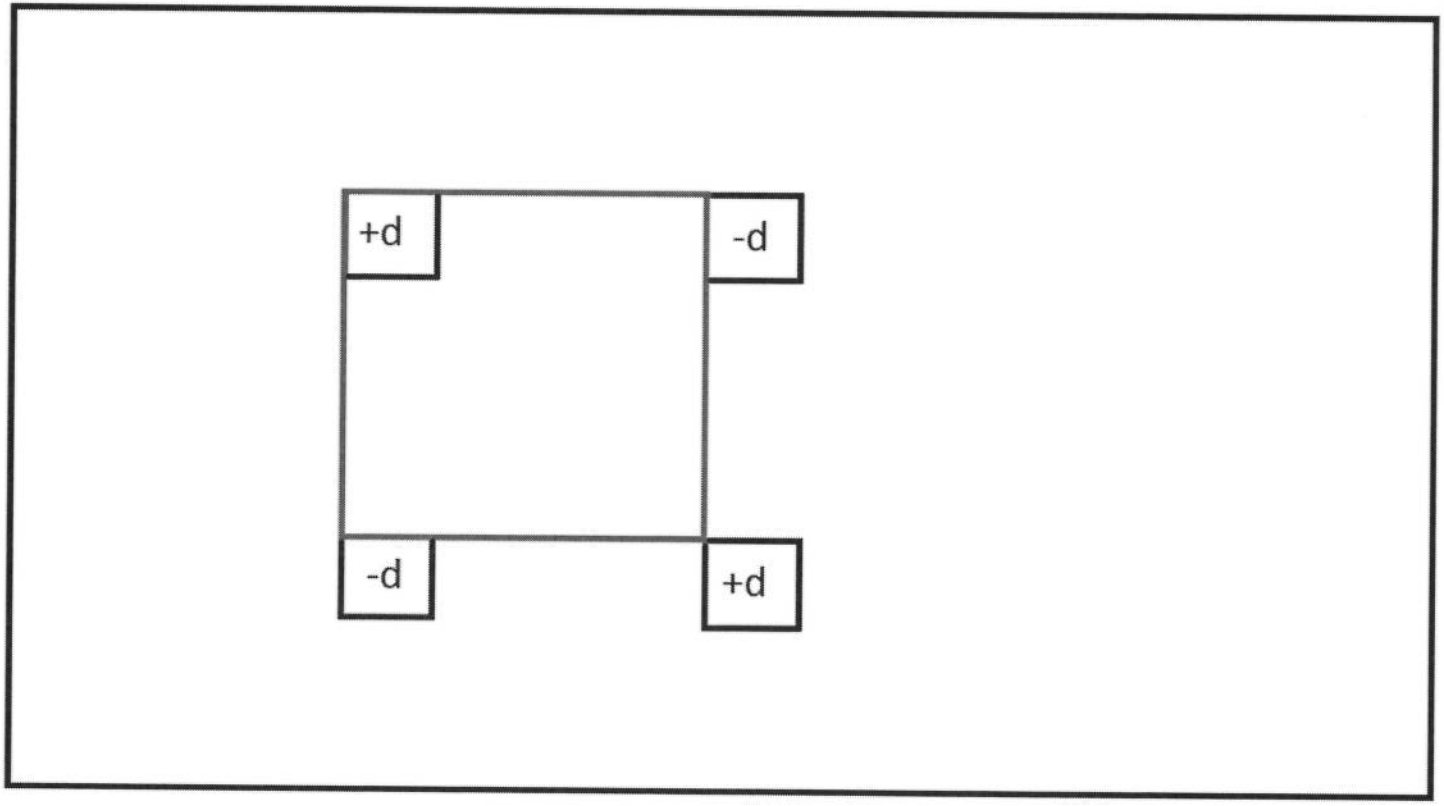
G. Игра Васи (продолжение)

- Как обработать все запросы?



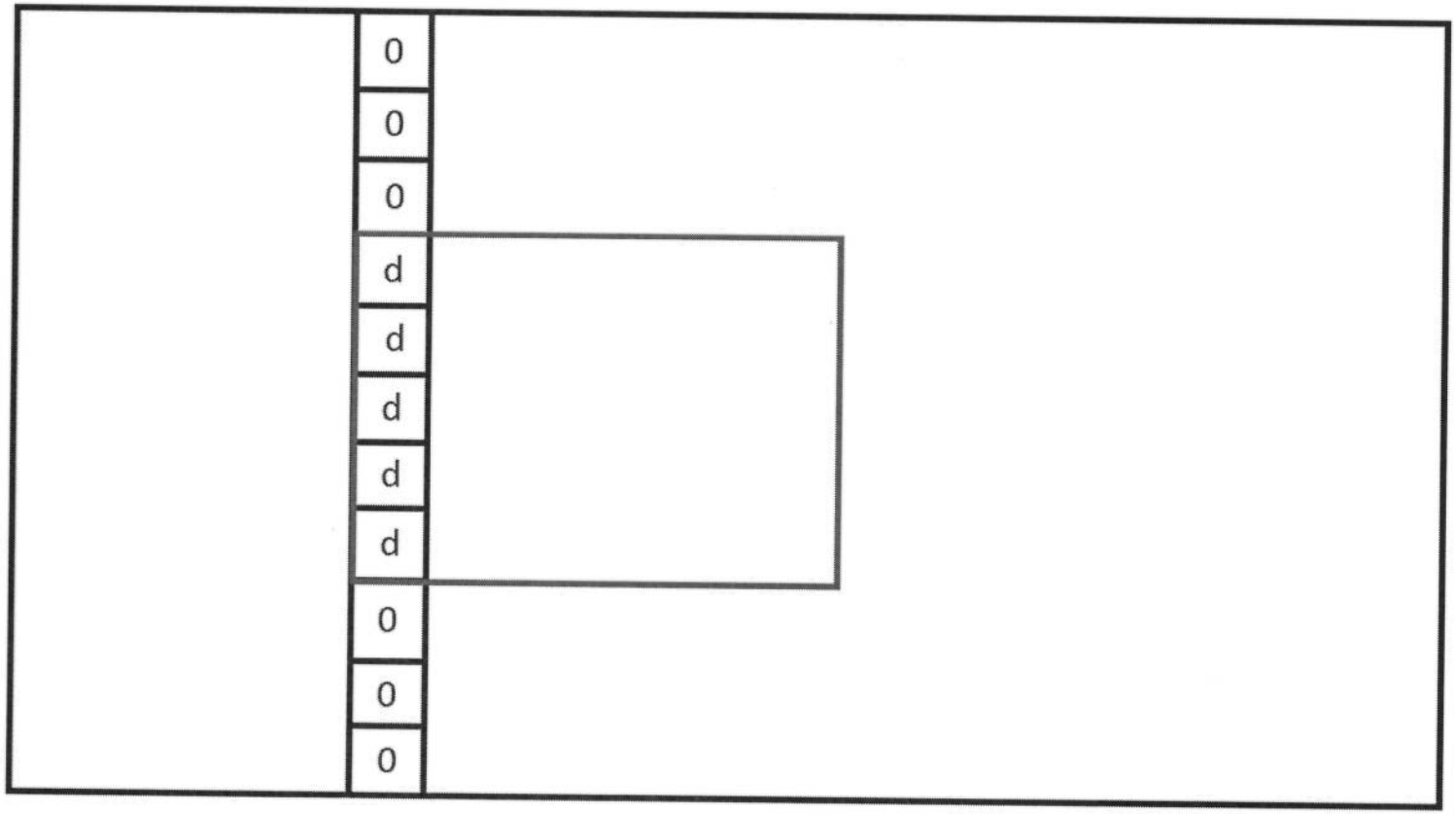
G. Игра Васи (продолжение)

- Обработка запроса сводиться к изменению 4 элементов матрицы



G. Игра Васи (продолжение)

- Формирование профиля.
- Проходом по всей таблице считаем ответ.



Задача H. LED-Цифры

Автор задачи: Булатов Константин

Разбор подготовил: Кодосов Никита

Первый AC: Назарбек Алтыбай +4 (1:05)

Общее количество AC: 27

H. LED-Цифры

1. Научимся считать ответ F(N) на отрезке [0,N]
2. Разбиваем на две подзадачи:
   a) Для горизонтальных (обязательно) и вертикальных (возможно)
   b) Только для вертикальных

H. LED-Цифры (продолжение)

а) Сюда подойдут цифры $1, 3, 8$ и 0, которые отражаются горизонтально. В ответ попадут и числа, состоящие только из 8 и 0, которые отражаются вертикально. Перебираем длину числа l:

I. Если длина меньше N, то в ответ добавляем $3 \cdot 4^{l-1}$

II. Иначе, генерируем число последовательно от наибольшего разряда. Если на текущей итерации мы взяли меньшую цифру, чем в соответствующем разряде, то добавляем в ответ 4^i, в противном случае при равенстве продвигаемся дальше.

H. LED-Цифры (продолжение)

b) Здесь подходят цифры $2, 5, 0$ и 8. Так как первая половина числа однозначно определяет вторую, то эту часть можно решить перебором за $O(4^{l/2})$, где l – длина числа. Не забываем отсекать варианты без двоек или пятерок, а также проверяем варианты с нечетной длиной (ставить в центр можем только 0 и 8).

H. LED-Цифры (продолжение)

3. Ответом на задачу будет $F(B) - F(A - 1)$

Задача I. Недели искусства

Автор задачи: Мамай Игорь

Разбор подготовил: Кодосов Никита

Первый AC: Айдарбек Хусаинов + (0:01)

Общее количество AC: 126

I. Раскраска

1. Перебираем значения y от 1 до 10^6.
2. Если число $(n - y^3)$ является квадратом натурального числа, увеличиваем счетчик на единицу.

Асимптотика алгоритма $O(\sqrt[3]{n})$

Задача J. Игра Пети

Автор задачи: Булатов Константин

Разбор подготовил: Тропин Даниил

Первый AC: Станислав Наумов + (0:16)

Общее количество AC: 98

J. Игра Пети

- Наблюдение #1: задача подозрительно похожа на G.
- Наблюдение #2: задача усложнилась большей размерностью M (до 10^9)
- Наблюдение #3: в задаче меньшее количество запросов Q (до 10^4).

J. Игра Пети (продолжение)

- Решение: Сжатие координат + Повторение задачи G.
- Для сжатия потребуется сортировка запросов по их координатам.
- Сложность задачи $O(NQ + Q\log_2 Q)$

Решения задач олимпиады (на языке C++)

Задача A

```
#include <iostream>
#include <math.h>
using namespace std;

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

	int f = max(0, (n / 4) - ((n % 4)!= 0 && (n % 4 < 3)));
	int l = n - 4 * f;
	int x = 0;
	int y = 0;
	int max = 900000000;
	for (int i = 0; i < f; ++i)
	{
		if (x > max)
		{
			y += 3;
			x = 0;
		}
		cout<<x<<' '<<y<<endl;
		cout<<x + 2<<' ' <<y<< endl;
		cout<<x + 1<<' '<<y + 1<<endl;
		cout<<x + 1<<' '<<y + 2<<endl;
		x += 3;
	}
	int r = 6 * f;

	switch (l)
	{
	case 1:
		cout<<"-1 -1\n";
		break;
	case 2:
		cout<<"-1 -1\n-2 -2\n";
		break;
	case 3:
		cout<<"-1 -1\n-2 -2\n-3 -1\n";
		r += 3;
		break;
	case 5:
		cout<<"-2 -1\n-1 -2\n-2 -3\n-2 -2\n-3 -2\n";
		r += 7;
		break;
	case 6:
		cout<<"-1 -1\n-2 -2\n-2 -3\n-3 -2\n-3 -3\n-4 -1\n";
		r += 9;
```

```
            break;
        default: break;
        }

        cout<<r<<endl;

        for (int i = 0; i < f; ++i)
        {
            cout<<4 * i + 1<<' '<<4 * i + 2<<endl;
            cout<<4 * i + 1<<' '<<4 * i + 3<<endl;
            cout<<4 * i + 1<<' '<<4 * i + 4<<endl;
            cout<<4 * i + 2<<' '<<4 * i + 3<<endl;
            cout<<4 * i + 2<<' '<<4 * i + 4<<endl;
            cout<<4 * i + 4<<' '<<4 * i + 3<<endl;
        }
        switch (l)
        {
        case 3:
            cout<<4 * f + 1<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 3<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 1<<' '<<4 * f + 3<<endl;
            break;
        case 5:
            cout<<4 * f + 1<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 1<<' '<<4 * f + 4<<endl;
            cout<<4 * f + 1<<' '<<4 * f + 5<<endl;
            cout<<4 * f + 3<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 4<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 3<<' '<<4 * f + 5<<endl;
            cout<<4 * f + 4<<' '<<4 * f + 3<<endl;
            break;
        case 6:
            cout<<4 * f + 1<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 1<<' '<<4 * f + 3<<endl;
            cout<<4 * f + 1<<' '<<4 * f + 6<<endl;
            cout<<4 * f + 3<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 4<<' '<<4 * f + 2<<endl;
            cout<<4 * f + 3<<' '<<4 * f + 5<<endl;
            cout<<4 * f + 4<<' '<<4 * f + 5<<endl;
            cout<<4 * f + 4<<' '<<4 * f + 6<<endl;
            cout<<4 * f + 5<<' '<<4 * f + 6<<endl;
            break;
        default: break;
        }

        return 0;
}
```

Задача B

```
#include <utility>
#include <vector>
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cstdio>
#include <cmath>
#include <algorithm>

#define vi  std::vector<long long>
#define vvi std::vector<vi>
#define ll  long long

double inf = 1e18;
struct line {
   double k, b, x;
   line(double k = 0, double b = 0, double x = -inf) {
      this->k = k;
      this->b = b;
      this->x = x;
   }
};

std::vector<line> v;

void add_line(double k, double b) {
   line l(k, b);
   while (true) {
      if (v.size() == 0) {
         v.push_back(l);
         return;
      }
      line p = v.back();
      if (l.k == p.k) {
         if (l.b >= p.b)
            return;
         v.pop_back();
      }
      else {
         double x = (l.b - p.b) / (p.k - l.k);
         if (x <= p.x)
            v.pop_back();
         else {
            l.x = x;
            v.push_back(l);
            return;
         }
      }
   }
}
```

```
long long get_val(double x) {
  int l = 0;
  int r = v.size() - 1;
  if (v[r].x <= x)
    return (ll)(v[r].k * x + v[r].b + 0.5);
  while (r - l > 1) {
    int mid = (l + r) / 2;
    if (v[mid].x <= x)
      l = mid;
    else
      r = mid;
  }
  return (ll)(v[l].k * x + v[l].b + 0.5);
}

void solve(int n, vi& k, vi& b, vi& c, vi& res) {
 res.resize(n);
 for (int i = 0; i < n; ++i) {
  std::vector<std::pair<double, double> > w;
  w.push_back(std::make_pair(k[i], b[i]));
  while (v.size() > 0 && v.back().k < k[i]) {
   w.push_back(std::make_pair(v.back().k, v.back().b));
   v.pop_back();
  }
  std::sort(w.begin(), w.end());
  std::reverse(w.begin(), w.end());
  for (int j = 0; j < w.size(); ++j)
   add_line(w[j].first, w[j].second);
  res[i] = get_val(c[i]);
 }
}

int main() {
 std::ios_base::sync_with_stdio(false);
 int n;
 vi k, b, c, res;
 std::cin >> n;
 k.resize(n);
 b.resize(n);
 c.resize(n);
 for (int i = 0; i < n; ++i) {
  std::cin >> b[i] >> k[i] >> c[i];
  b[i] -= k[i];
 }
 solve(n, k, b, c, res);
 for (int i = 0; i < n; ++i)
  std::cout << res[i] << "\n";

 return 0;
}
```

Задача C

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

int main() {
	long long a, b;
	cin >> a >> b;
	cout << ((a + b) * (b - a + 1) / 2) << endl;
	return 0;
}
```

Задача D

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <cstring>
#include <vector>
#include <string>
#include <cstdio>
#include <queue>
#include <ctime>
#include <list>
#include <set>
#include <map>

double solve(double h, double t) {
  if (t < h) {
    return t - t * t / (2.0 * h);
  }
  else {
    return h / 2.0;
  }
}

int testgen();

int main() {
  double t = 0.0;
  double h = 0.0;
  scanf("%lf\n", &t);
  scanf("%lf\n", &h);
  printf("%.6lf\n", solve(h, t));
  return 0;
}
```

Задача E

```
#include "testlib.h"
#include <cmath>

using namespace std;

const double EPS = 1E-4;

int main(int argc, char * argv[])
{
    setName("compare two sequences of doubles, max absolute or relative error = %.5lf", EPS);
    registerTestlibCmd(argc, argv);

    int n = 0;
    double j, p;

    while (!ans.seekEof())
    {
        n++;
        j = ans.readDouble();
        p = ouf.readDouble();
        if (!doubleCompare(j, p, EPS))
        {
            quitf(_wa, "%d%s numbers differ - expected: '%.5lf', found: '%.5lf', error = '%.5lf'",
                n, englishEnding(n).c_str(), j, p, doubleDelta(j, p));
        }
    }

    if (n == 1)
        quitf(_ok, "found '%.5lf', expected '%.5lf', error '%.5lf'", p, j, doubleDelta(j, p));

    quitf(_ok, "%d numbers", n);
}
```

Задача F

```
#include <cmath>
#include <vector>
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cstdio>

const int inf = 100000;
#define vp  std::vector<std::pair<int, int> >
#define vvp std::vector<vp >
#define vi  std::vector<int>
#define vvi std::vector<vi >

bool check_typos(int a, int b, int c, int d) {
 if(a < 8 || b < 8 || c < 8 || d < 8)
  return false;
 if(a > 18 || b > 18 || c > 18 || d > 18)
  return false;
 if(a > b || c > d || c < a)
  return false;
 return true;
}

int meeting_len(int a, int b, int c, int d) {
 if(c >= b)
  return (b - a + d - c);
 if(d <= b)
  return (b - a);
 return (std::min(18, d + b - c) - a);
}

void key_to_typos(int T, int key, int& ia, int& ib, int& ic, int& id) {
 ia = key % 3;
 ib = ((key - ia) / 3) % 3;
 ic = ((key - ia - 3 * ib) / 9) % 3;
 id = key / 27;
 --ia; --ib; --ic; --id;
 ia *= T; ib *= T; ic *= T; id *= T;
}

int typos_key(int ia, int ib, int ic, int id, int T) {
 ia /= T; ib /= T; ic /= T; id /= T;
 ++ia; ++ib; ++ic; ++id;
 return (ia + 3 * (ib + 3 * (ic + 3 * id)));
}

int typos_num(int ia, int ib, int ic, int id, int T) {
 ia /= T; ib /= T; ic /= T; id /= T;
 return (std::abs(ia) + std::abs(ib) + std::abs(ic) + std::abs(id));
}
```

```
void make_precalc(vvp& precalc, vi& a, vi& b, vi& c, vi& d, int n, int k, int T) {
 for(int i = 0; i < n; ++i) {
  for(int ia = -T; ia <= T; ia += T) {
   for(int ib = -T; ib <= T; ib += T) {
    for(int ic = -T; ic <= T; ic += T) {
     for(int id = -T; id <= T; id += T) {
      if(check_typos((a[i]+ia), (b[i]+ib), (c[i]+ic), (d[i]+id))) {
       int key = typos_key(ia, ib, ic, id, T);
       int len = meeting_len((a[i]+ia), (b[i]+ib), (c[i]+ic), (d[i]+id));
       int num = typos_num(ia, ib, ic, id, T);
       if(len < precalc[i][num].first) {
        precalc[i][num].first  = len;
        precalc[i][num].second = key;
       }
      }
     }
    }
   }
  }
 }
}

void dp_init(vvi& dp, vvi& dp2, vvi& dp3, vi& a, vi& b, vi& c, vi& d, vvp& precalc, int n, int
k, int T) {
 for(int j = 0; j < std::min(k+1, 5); ++j) {
  dp[0][j]  = precalc[0][j].first;
  dp2[0][j] = precalc[0][j].second;
 }
}

void dp_fill(vvi& dp, vvi& dp2, vvi& dp3, vi& a, vi& b, vi& c, vi& d, vvp& precalc, int n, int k,
int T) {
 for(int i = 1; i < n; ++i) {
  for(int j = 0; j < k+1; ++j) {
   for(int l = 0; l < 5; ++l) {
    if(j + l <= k && dp[i-1][j] + precalc[i][l].first < dp[i][j+l]) {
     dp[i][j+l]  = dp[i-1][j] + precalc[i][l].first;
     dp2[i][j+l] = precalc[i][l].second;
     dp3[i][j+l] = j;
    }
   }
  }
 }
}

void solve(vvi& dp, vvi& dp2, vvi& dp3, vi& a, vi& b, vi& c, vi& d, vvp& precalc, int n, int k,
int T) {
 int typos = k;
 for(int i = n-1; i >= 0; --i) {
  int ia, ib, ic, id;
  int key = dp2[i][typos];
  key_to_typos(T, key, ia, ib, ic, id);
```

```
    a[i] += ia;
    b[i] += ib;
    c[i] += ic;
    d[i] += id;
    typos = dp3[i][typos];
  }
  for(int i = 0; i < n; ++i)
    std::cout << a[i] << " " << b[i] << " " << c[i] << " " << d[i] << std::endl;
}

int main() {
  std::ios_base::sync_with_stdio(false);
  int n, k, T;
  std::cin >> n >> k >> T;

  vi a(n, 0);
  vi b(n, 0);
  vi c(n, 0);
  vi d(n, 0);
  for(int i = 0; i < n; ++i)
    std::cin >> a[i] >> b[i] >> c[i] >> d[i];

  vp  initpre(5, std::make_pair(inf, inf));
  vvp precalc(n, initpre);
  vi  init(k+1, inf);
  vvi dp(n, init);  // minimal schedule
  vvi dp2(n, init); // typos type(key)
  vvi dp3(n, init); // parent -> number of typos on previous step

  make_precalc(precalc, a, b, c, d, n, k, T);
  dp_init(dp, dp2, dp3, a, b, c, d, precalc, n, k, T);
  dp_fill(dp, dp2, dp3, a, b, c, d, precalc, n, k, T);
  if(dp[n-1][k] != inf)
    solve(dp, dp2, dp3, a, b, c, d, precalc, n, k, T);
  else
    std::cout << "Impossible" << std::endl;

  return 0;
}
```

Задача G

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <cstring>
#include <vector>
#include <string>
#include <cstdio>
#include <queue>
#include <ctime>
#include <list>
#include <set>
#include <map>

void solve(int q,
        std::vector<std::pair<int, int> >& l_top,
        std::vector<std::pair<int, int> >& r_bot,
        std::vector<int>& addition,
        int m,
        int n,
        std::vector<std::vector<int> >& bias,
        std::vector<std::vector<int> >& ans) {
 ans.assign(m + 1, std::vector<int>(n + 1, 0));

 for (int i = 0; i < q; ++i) {
  ans[l_top[i].first][l_top[i].second] += addition[i];
  ans[l_top[i].first][r_bot[i].second + 1] -= addition[i];
  ans[r_bot[i].first + 1][l_top[i].second] -= addition[i];
  ans[r_bot[i].first + 1][r_bot[i].second + 1] += addition[i];
 }

 for (int i = 0; i < m; ++i) {
  for (int j = 1; j < n; ++j) {
   ans[i][j] += ans[i][j - 1];
  }
 }

 for (int j = 0; j < n; ++j) {
  for (int i = 1; i < m; ++i) {
   ans[i][j] += ans[i - 1][j];
  }
 }

 for (int i = 0; i < m; ++i) {
  for (int j = 0; j < n; ++j) {
   ans[i][j] += bias[i][j];
  }
 }
}

int main() {
```

```
  int q = 0;
  std::vector<std::pair<int, int> > l_top;
  std::vector<std::pair<int, int> > r_bot;
  std::vector<int> addition;
  int m = 0;
  int n = 0;
  std::vector<std::vector<int> > bias;
  std::vector<std::vector<int> > ans;

  ::scanf("%d", &q);
  l_top.resize(q);
  r_bot.resize(q);
  addition.resize(q);

  for (int i = 0; i < q; ++i) {
    ::scanf("%d %d %d %d %d", &l_top[i].first, &l_top[i].second,
                              &r_bot[i].first, &r_bot[i].second,
                              &addition[i]);
    --l_top[i].first;
    --l_top[i].second;
    --r_bot[i].first;
    --r_bot[i].second;
  }

  ::scanf("%d %d", &m, &n);
  bias.assign(m, std::vector<int>(n, 0));

  for (int i = 0; i < m; ++i) {
    for (int j = 0; j < n; ++j) {
      ::scanf("%d", &bias[i][j]);
    }
  }

  solve(q, l_top, r_bot, addition, m, n, bias, ans);

  for (int i = 0; i < m; ++i) {
    for (int j = 0; j < n; ++j) {
      if (j) ::printf(" ");
      ::printf("%d", ans[i][j]);
    }
    ::printf("\n");
  }

  return 0;
}
```

Задача H

```
/**
 * 'LED-digits' problem.
 *
 * Compiling solution: $ g++ -std=c++11 led.cpp -o led
 * Compiling test gen: $ g++ -std=c++11 led.cpp -o led-testgen -DTEST_GENERATOR
 */

#include <algorithm>
#include <iostream>
#include <string>
#include <vector>
#include <random>

std::string num_2_string(long long a) {
  std::string str;
  if (a == 0) {
    str = "0";
  } else while (a) {
    str += (char)('0' + (a % 10));
    a /= 10;
  }
  std::reverse(str.begin(), str.end());
  return str;
}

long long string_2_num(const std::string& str) {
  long long ret = 0;
  for (size_t i = 0; i < str.length(); ++i) {
    ret *= 10;
    ret += (long long)(str[i] - '0');
  }
  return ret;
}

bool horizontally_symmetric_only(long long a) {
  std::string led = num_2_string(a);
  for (size_t i = 0; i < led.length(); ++i) {
    if (led[i] != '0' && led[i] != '1' && led[i] != '3' && led[i] != '8') {
      return false;
    }
  }
  return true;
}

bool vertically_symmetric_only(long long a) {
  std::string led = num_2_string(a);
  if (led.length() % 2 == 1) {
    if (led[led.length() / 2] != '8' &&
        led[led.length() / 2] != '0') {
      return false;
    }
```

```
  }
  bool found_25 = false;
  for (size_t i = 0; i < led.length() / 2; ++i) {
    if (led[i] != '8' && led[i] != '0' && led[i] != '2' && led[i] != '5') {
      return false;
    }
    if (led[i] == '8' && led[led.length() - i - 1] != '8')
      return false;
    if (led[i] == '0' && led[led.length() - i - 1] != '0')
      return false;
    if (led[i] == '2' && led[led.length() - i - 1] != '5')
      return false;
    if (led[i] == '5' && led[led.length() - i - 1] != '2')
      return false;
    if (led[i] == '2' || led[i] == '5') {
      found_25 = true;
    }
  }
  return found_25;
}

// Counts horizontally symmetric numbers in range [0, n]
long long count_only_horizontally_symmetric(long long n) {
  std::string n_str = num_2_string(n);
  long long ret = 0;
  long long digless[10] = {0, 1, 2, 2, 3, 3, 3, 3, 3, 4};
  bool exited = false;
  for (size_t i = 0; i < n_str.length(); ++i) {
    ret += digless[n_str[i] - '0'] * (1ll << (2 * (n_str.length() - i - 1)));
    if (n_str[i] != '0' &&
        n_str[i] != '1' &&
        n_str[i] != '3' &&
        n_str[i] != '8') {
      exited = true;
      break;
    }
  }
  if (!exited) {
    ret++;
  }
  return ret;
}

// Procedure generates all only-verticall symmetric numbers with k digits
std::string gen_str = "00000000000000000000";
long long gen_num = 0;
std::vector<long long> gen_pow10;
long long gen_counter = 0;
long long gen_compare = 0;

inline void register_number() {
  if (gen_num <= gen_compare) {
```

```
    ++gen_counter;
  }
}

inline void put_symbol(int pos, char sym) {
  gen_num -= gen_pow10[pos] * (gen_str[pos] - '0');
  gen_str[pos] = sym;
  gen_num += gen_pow10[pos] * (sym - '0');
}

inline void put_string(int pos, const std::string& str) {
  for (size_t i = 0; i < str.length(); ++i) {
    put_symbol(pos + i, str[i]);
  }
}

void generate_only_vertically_symmetric(int k, int start, bool has_25, bool may_put_zero) {
  if (k == 2) {
    put_string(start, "25"); register_number();
    put_string(start, "52"); register_number();
    if (has_25) {
      put_string(start, "88"); register_number();
      if (may_put_zero) {
        put_string(start, "00"); register_number();
      }
    }
  } else if (k == 3) {
    put_string(start, "205"); register_number();
    put_string(start, "285"); register_number();
    put_string(start, "502"); register_number();
    put_string(start, "582"); register_number();
    if (has_25) {
      put_string(start, "808"); register_number();
      put_string(start, "888"); register_number();
      if (may_put_zero) {
        put_string(start, "000"); register_number();
        put_string(start, "080"); register_number();
      }
    }
  } else {
    // 2.....5
    put_symbol(start, '2');
    put_symbol(start + k - 1, '5');
    generate_only_vertically_symmetric(k - 2, start + 1, true, true);
    // 5.....2
    put_symbol(start, '5');
    put_symbol(start + k - 1, '2');
    generate_only_vertically_symmetric(k - 2, start + 1, true, true);
    // 8.....8
    put_symbol(start, '8');
    put_symbol(start + k - 1, '8');
    generate_only_vertically_symmetric(k - 2, start + 1, has_25, true);
```

```
    if (may_put_zero) {
      put_symbol(start, '0');
      put_symbol(start + k - 1, '0');
      generate_only_vertically_symmetric(k - 2, start + 1, has_25, true);
    }
  }
}

// Counts only-vertically symmetric numbers in range [0, n]
long long count_only_vertically_symmetric(long long n) {
  if (n < 10ll) {
    return 0ll; // no 1-digit numbers
  }

  // f[k]: number of only-vertically symmetric numbers with k digits
  std::vector<long long> f(20), f0(20);

  f[0] = f0[0] = 0;
  f[1] = f0[1] = 0;
  f[2] = f0[2] = 2;

  for (int k = 3; k < 20; ++k) {
    f[k] = 3ll * f0[k - 2] + (1ll << ((k - 1) / 2 + 1));
    f0[k] = 4ll * f0[k - 2] + (1ll << ((k - 1) / 2 + 1));
  }

  std::string n_str = num_2_string(n);
  long long ret = 0;
  for (size_t k = 0; k < n_str.length(); ++k) {
    ret += f[k];
  }

  // prepare generator
  gen_pow10.assign(20, 0);
  gen_pow10[0] = 1;
  for (size_t i = 1; i < 20; ++i) {
    gen_pow10[i] = 10ll * gen_pow10[i - 1];
  }
  gen_num = 0;
  gen_str = "00000000000000000000";
  gen_counter = 0;
  gen_compare = n;

  generate_only_vertically_symmetric(n_str.length(), 0, false, false);

  return ret + gen_counter;
}

long long solve(long long a, long long b) {
  long long ans = count_only_horizontally_symmetric(b) +
                  count_only_vertically_symmetric(b);
  if (a > 0) {
```

```
    long long sub = count_only_horizontally_symmetric(a - 1) +
                    count_only_vertically_symmetric(a - 1);
    ans -= sub;
  }
  long long modulo = 1000ll * 1000ll * 1000ll + 7ll;
  return ans % modulo;
}

int main() {

  long long a, b;
  std::cin >> a >> b;
  std::cout << solve(a, b) << std::endl;
  return 0;
}
```

Задача I

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cmath>

#define ll long long

ll solve(ll n) {
	ll ans = 0;
	for (ll y = 1; y * y * y < n; y++) {
		ll temp = n - y * y * y;
		ll x = (ll)std::sqrt((temp + 0.1));
		if (temp == x * x) {
			ans++;
		}
	}
	return ans;
}

int testgen();

int main(){
	std::ios_base::sync_with_stdio(false);
	ll n;
	std::cin >> n;
	std::cout << solve(n) << std::endl;

	return 0;
}
```

Задача J

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <cstring>
#include <vector>
#include <string>
#include <cstdio>
#include <queue>
#include <ctime>
#include <list>
#include <set>
#include <map>

void solve(int q,
        std::vector<std::pair<int, int> >& l_top,
        std::vector<std::pair<int, int> >& r_bot,
        std::vector<int>& addition,
        int m,
        int n,
        int& ans) {
 ans = -1e9;
 std::vector<std::pair<long long, int> > events;

 l_top.push_back(std::make_pair(0, 0));
 r_bot.push_back(std::make_pair(m - 1, n - 1));
 addition.push_back(0);
 for (int query = 0; query <= q; ++query) {
  for (int col = l_top[query].second; col <= r_bot[query].second; ++col) {
   events.push_back(std::make_pair(
       1ll * col * m + l_top[query].first, addition[query]));
   events.push_back(std::make_pair(
       1ll * col * m + r_bot[query].first + 1, -addition[query]));
  }
 }

 std::sort(events.begin(), events.end());

 int cur = 0;
 for (size_t i = 0; i < events.size(); ++i) {
  if (i > 0 && events[i].first != events[i - 1].first) {
   ans = std::max(ans, cur);
  }
  cur += events[i].second;
 }
}

int main() {
 int q = 0;
 std::vector<std::pair<int, int> > l_top;
 std::vector<std::pair<int, int> > r_bot;
 std::vector<int> addition;
```

```
  int m = 0;
  int n = 0;
  int ans = 0;

  ::scanf("%d", &q);
  l_top.resize(q);
  r_bot.resize(q);
  addition.resize(q);

  for (int i = 0; i < q; ++i) {
    ::scanf("%d %d %d %d %d", &l_top[i].first, &l_top[i].second,
                              &r_bot[i].first, &r_bot[i].second,
                              &addition[i]);
    --l_top[i].first;
    --l_top[i].second;
    --r_bot[i].first;
    --r_bot[i].second;
  }

  ::scanf("%d %d", &m, &n);

  solve(q, l_top, r_bot, addition, m, n, ans);

  ::printf("%d\n", ans);

  return 0;
}
```