Всероссийская олимпиада школьников по информатике 2023-2024

Районный тур, 7-11 класс
Санкт-Петербург, 4 декабря 2023 года

РАЗБОР ЗАДАЧ

Задачи и разбор

Андрей Сергеевич Станкевич

Маргарита Марковна Саблина

Николай Викторович Ведерников

Григорий Филиппович Шовкопляс

Задача A

Изобразить перекресток на клетчатой плоскости

Даны ширина дороги и размер клетки

Задача A - решение

Выводим (n - l)/2 строк ..***..

Выводим l строк *******

Выводим (n - l) / 2 строк ..***..

Задача A - код

```
n = int(input())
l = int(input())
q = (n - l) // 2

for i in range(q):
    print("." * q + "*" * l + "." * q)

for i in range(l):
    print("*" * n)

for i in range(q):
    print("." * q + "*" * l + "." * q)
```

Задача B

Даны целые положительные a и b

Найти количество целых положительных c, что существует треугольник со сторонами a, b, c

Задача B - решение

Неравенство треугольника:

- a + b > c
- a + c > b
- b + c > a

Пусть a ≥ b, тогда достаточно проверить

- a + b > c
- b + c > a, то есть c > a − b

Задача B - решение

Итого надо найти количество чисел c

- a - b < c < a + b

Таких чисел a + b - (a - b) - 1 = 2 * b - 1

- Или просто maxc = a + b - 1, minc = a - b + 1
- ans = maxc - minc + 1

Задача B - код

```
---

a = int(input())
b = int(input())

a, b = max(a, b), min(a, b)

minc = a - b + 1
maxc = a + b - 1

ans = maxc - minc + 1

print(ans)
```

Задача C

Вдоль дорожки растут деревья, i-е дерево на расстоянии d[i] от предыдущего дерева

Надо поставить старт и финиш дистанции около начала дорожки или у дерева

Длина должна быть от L до R

При прочих равных как можно ближе к началу финиш, снова при прочих равных старт

Если решения нет, вывести -1 -1

Задача C - решение

Сначала найдем позиции деревьев, массив префиксных сумм:

- p[0] = 0
- p[i + 1] = p[i] + d[i]

Используем два указателя, пусть финиш около дерева ti

Тогда двигаем старт si, пока длина не будет не больше R

Если оказалось не меньше L, то это ответ

Задача C - код

```
vector<long long> p(n + 1);
for (int i = 0; i < n; i++) {
     p[i + 1] = p[i] + d[i];
}

long long s = -1, t = -1;
int si = 0;
for (int ti = 1; ti <= n; ti++) {
     while (p[ti] - p[si] > R) {
          si++;
     }
     if (L <= p[ti] - p[si] && p[ti] - p[si] <= R) {
          s = p[si]; t = p[ti];
          break;
     }
}

cout << s << " " << t << endl;
```

Задача D

Представляем n в виде суммы целых неубывающих слагаемых

Вывести все способы сделать это, чтобы никакое слагаемое не было простым числом.

1+1+1+1+1

1+4

Задача D - решение

Сначала найдем простые числа

```
set<int> primes;

for (int i = 2; i <= n; i++) {
     bool is_prime = true;
     for (int x : primes) {
          if (i % x == 0) {
               is_prime = false;
               break;
          }
     }
     if (is_prime) {
          primes.insert(i);
     }
}
```

Задача D - решение

———

Теперь генерируем все разбиения рекурсивно, не используя простые числа

```
void gen(int n, int f, int p) {
     if (n == 0) {
          // вывод ответа
     }

     if (n < f) return; // отсечение

     for (int g = f; g <= n; g++) {
          if (primes.find(g) == primes.end()) {
               a[p] = g;
               gen(n - g, g, p + 1);
          }
     }
}
```

Задача E

Есть веревочка длиной n сантиметров

Каждый шаг берется самая длинная из имеющихся и разрезается на две примерно равные

После k разрезаний какие будут длины?

(вывести некторые из них, с номерами q[1], …, q[qn])

[100]
[50, 50]
[50, 25, 25]
[25, 25, 25, 25]
[25, 25, 25, 13, 12]
[25, 25, 13, 13, 12, 12]

Задача E - решение

Будем хранить map<long long, long long> - сколько веревочек какой длины

Берем самую длинную

- если их не больше, чем осталось разрезать - разрезаем все
- если их больше - разрезаем сколько осталось

Задача E - код

Разрезание

```
map<long long, long long> d;
d[n] = 1;
while (k > 0) {
      auto [u, c] = *d.rbegin();
      long long n1 = (u + 1) / 2;
      long long n2 = u / 2;
      if (c <= k) {
            d.erase(u);
            d[n1] += c;
            d[n2] += c;
            k -= c;
      } else {
            d[u] -= k;
            d[n1] += k;
            d[n2] += k;
            k = 0;
      }
}
```

Задача E - код

Вывод ответа

```
    vector<long long> ans;
    int pos = 0;
    long long m = 0;
    for (auto ptr = d.rbegin(); ptr != d.rend(); ptr++) {
        auto [u, c] = *ptr;
        while (pos < qn && q[pos] <= m + c) {
            ans.push_back(u);
            pos++;
        }
        m += c;
    }
```

Задача E - решение

Второй подход к решению: заметим, то различных текущих длин всегда не больше 4

- l, l+1, 2l, 2l+1

или

- l, l+1, 2l-1, 2l

Поддерживаем, какой из двух случаев имеет место и сколько какой длины

Задача E - решение

Второй подход позволяет упростить первое решение: вместо map можно использовать vector и искать максимум каждый раз вручную

Задача F

$n = (p * q)^k$

Сколько способов представить n в виде произведения двух и более слагаемых?

Задача F - решение

Ответ зависит только от k

Всем множители имеют вид $p^i q^j$

Упорядочим множители по возрастанию i, при равном i по возрастанию j

Используем динамическое программирование

dp[s][t][i][j] - количество способов разбить остаток $p^s q^t$, если можно использовать только множители $p^{i'} q^{j'}$ с i'>i или i'=i и j'≥j

Задача F - код

Хороший вариант - ленивые вычисления

```
long long go(pair<int, int> n, pair<int, int> last) {
    if (n == make_pair(0, 0)) {
        return 1;
    }
    if (mem.find({n, last}) != mem.end()) {
        return mem[{n, last}];
    }

    long long res = 0;
    if (last.first <= n.first && last.second <= n.second)
        res += go({n.first - last.first, n.second - last.second}, last);
```

Задача F - код

```
    auto next = last;
    if (next.second == k) {
        next.first++;
        next.second = 0;
    } else {
        next.second++;
    }

    if (last.first <= k) {
        res += go(n, next);
    }

    mem[{n, last}] = res;

    return res;
}
```

Спасибо за внимание

spbmunicipal@gmail.com