Всероссийская олимпиада школьников по информатике
Региональный этап, II тур
Разбор задач

Андрианов И. А.
Стрекаловский О. А.

Вологодский государственный педагогический университет
Факультет прикладной математики,
компьютерных технологий и физики

Вологда
2014 г.

«Светофоры»

Идея решения на 100 баллов
Целочисленное (кроме вывода ответа)
Сложность $O(X/1000)$

1. Заметим, что достаточно рассмотреть только два момента старта — когда первый светофор меняет состояние с красного на зеленый и с зеленого на красный.
2. Обозначим за t время движения, необходимое чтобы проехать на второй светофор на зелёный.

«Светофоры»

Идея решения на 100 баллов

Целочисленное (кроме вывода ответа)

Сложность $O(X/1000)$

1. Заметим, что достаточно рассмотреть только два момента старта — когда первый светофор меняет состояние с красного на зеленый и с зеленого на красный.
2. Обозначим за t время движения, необходимое чтобы проехать на второй светофор на зелёный.

Стартуем в момент смены зеленого на красный

Должно выполниться одно из следующих неравенств:

$$0 \le t \le a$$

$$a + b \le t \le 2 * a + b$$

$$2 * a + 2 * b \le t \le 3 * a + 2 * b$$

и т.д.

То есть $k * (a + b) \le t \le k * (a + b) + a$, где $k = 0, 1, 2, \ldots$.

Стартуем в момент смены красного на зеленый

Должно выполниться одно из следующих неравенств:

$$b \leq t \leq b + a$$

$$2 * b + a \leq t \leq 2 * b + 2 * a$$

и т.д.

То есть $k * (a + b) + b \leq t \leq (k + 1) * (a + b)$, где $k = 0, 1, \ldots$.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2_, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
   Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
   Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
   Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2_, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

1. Перебираем эти интервалы, увеличивая k.
2. Пусть $t_1 = k * (a + b)$, а $t_2 = k * (a + b) + a$.
3. Зная диапазон времени $[t_1, t_2]$, определяем диапазон скорости $[v_1, v_2]$, где $v_1 = \frac{x}{t_2}, v_2 = \frac{x}{t_1}$.
4. Если $v_1 > 1000$, то это нам не подходит, так как наш кар не может ехать так быстро.
5. Как только v_1 станет меньше или равен 1000, то ответом будет $\min(v2_, 1000)$.
6. Чтобы сравнивать не вещественные числа, а целые, можно переписать неравенства, чтобы вместо делений были умножения.
   Например, вместо `x/t2 <= 1000` будет `x <= 1000 * t2`.

«Светофоры»

Идея решения на 100 баллов

Вещественная арифметика

Сложность $O(1)$

1. Обозначим за `interval` интервал светофора (время продолжения одного цикла «зелёный» – «красный»).
   `interval := a + b`
2. Обозначим за `t` время движения между светофорами на максимальной скорости.
   `t := dist / 1000`. Деление вещественное.
3. Обозначим за `intervalTime` момент в интервале работы второго светофора, когда мы подъезжаем на максимальной скорости.
   `intervalTime := time % interval`, где % — операция взятия вещественного остатка от деления вещественного числа на целое.

«Светофоры»

Идея решения на 100 баллов

Вещественная арифметика

Сложность $O(1)$

1. Обозначим за `interval` интервал светофора (время продолжения одного цикла «зелёный» – «красный»).
   `interval := a + b`
2. Обозначим за `t` время движения между светофорами на максимальной скорости.
   `t := dist / 1000`. Деление вещественное.
3. Обозначим за `intervalTime` момент в интервале работы второго светофора, когда мы подъезжаем на максимальной скорости.
   `intervalTime := time % interval`, где % — операция взятия вещественного остатка от деления вещественного числа на целое.

«Светофоры»

Идея решения на 100 баллов

Вещественная арифметика

Сложность $O(1)$

1. Обозначим за `interval` интервал светофора (время продолжения одного цикла «зелёный» – «красный»).
   `interval := a + b`
2. Обозначим за `t` время движения между светофорами на максимальной скорости.
   `t := dist / 1000`. Деление вещественное.
3. Обозначим за `intervalTime` момент в интервале работы второго светофора, когда мы подъезжаем на максимальной скорости.
   `intervalTime := time % interval`, где % — операция взятия вещественного остатка от деления вещественного числа на целое.

Примеры реализация данной операции % на разных языках

Java

```
intervalTime = time % interval
```

Pascal

```
intervalTime :=
     time - floor(time / (interval)) * (interval)
```

C++

```
intervalTime =
     time - (int)(time / interval) * interval
```

Примеры реализация данной операции % на разных языках

Java

```
intervalTime = time % interval
```

Pascal

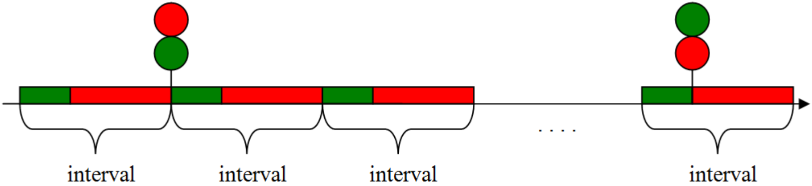
```
intervalTime :=
     time - floor(time / (interval)) * (interval)
```

C++

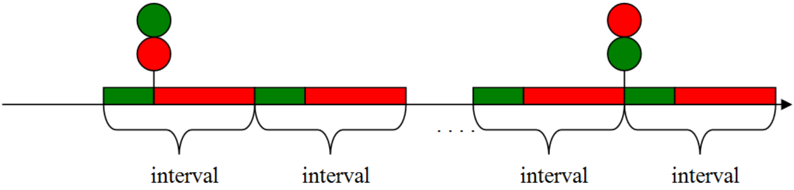
```
intervalTime =
     time - (int)(time / interval) * interval
```

Мы можем проехать на максимальной скорости через второй светофор если выполняется хотя бы одно из условий:

- №1. С первого светофора стартанём в момент перехода «красный» – «зелёный» и приедем ко второму светофору, когда на нём будет зелёный до перехода на красный.
- Это можно записать следующим образом: `intervalTime <= a`.



- №2. С первого светофора стартанём в момент перехода «зелёный» – «красный» и успеем подъехать ко второму светофору, когда на нём будет переход «красный» – «зелёный» и до переключения на красный.
- Это можно записать следующим образом: `intervalTime >= b`.



1. Если мы не успеваем проехать на максимальной скорости, то надо «притормозить» на время, пока продолжает гореть красный как в случае №2 выше:
   `time := time + b - intervalTime`.
2. Найти скорость через вещественное деление пути на время:
   `speed := x / time`.

1. Если мы не успеваем проехать на максимальной скорости, то надо «притормозить» на время, пока продолжает гореть красный как в случае №2 выше:
   `time := time + b - intervalTime`.
2. Найти скорость через вещественное деление пути на время:
   `speed := x / time`.

Вопросы по задаче?

«Кондиционеры»

Идея решения на 100 баллов
Без сортировки
Сложность $O(M * 1000)$

1. Прочитаем данные о классах в массив $classes$, где $classes[i]$ — необходимая мощность кондиционера в i–м класе.
2. Создадим массив $price$, где $price[i]$ — минимальная стоимость кондиционера с мощностью $\geq i$.
3. Прочитав из файла данные об очередном кондиционере с мощностью b и стоимостью c, обновляем элементы $price[1] \ldots price[b]$:

```
if price[i] > c then
  price[i] := c
```

«Кондиционеры»

Идея решения на 100 баллов
Без сортировки
Сложность $O(M * 1000)$

1. Прочитаем данные о классах в массив $classes$, где $classes[i]$ — необходимая мощность кондиционера в i–м класе.
2. Создадим массив $price$, где $price[i]$ — минимальная стоимость кондиционера с мощностью $\geq i$.
3. Прочитав из файла данные об очередном кондиционере с мощностью b и стоимостью c, обновляем элементы $price[1] \ldots price[b]$:

```
if price[i] > c then
  price[i] := c
```

«Кондиционеры»

Идея решения на 100 баллов
Без сортировки
Сложность $O(M * 1000)$

1. Прочитаем данные о классах в массив $classes$, где $classes[i]$ — необходимая мощность кондиционера в i–м класе.
2. Создадим массив $price$, где $price[i]$ — минимальная стоимость кондиционера с мощностью $\geq i$.
3. Прочитав из файла данные об очередном кондиционере с мощностью b и стоимостью c, обновляем элементы $price[1] \dots price[b]$:

```
if price[i] > c then
  price[i] := c
```

1. Ответ для каждого класса i лежит в $price[classes[i]]$.
2. Ответом на задачу будет сумма ответов для всех классов.
3. Количество действий порядка $M * 1000$, так как мощность не превышает 1000.

1. Ответ для каждого класса i лежит в $price[classes[i]]$.
2. Ответом на задачу будет сумма ответов для всех классов.
3. Количество действий порядка $M * 1000$, так как мощность не превышает 1000.

1. Ответ для каждого класса i лежит в $price[classes[i]]$.
2. Ответом на задачу будет сумма ответов для всех классов.
3. Количество действий порядка $M * 1000$, так как мощность не превышает 1000.

«Кондиционеры»

Идея решения на 100 баллов

Сортировка + «жадный алгоритм».
Сложность $O(M * \log(M) + N * \log(N))$

1. Сортируем массив с кондиционерами ($conds$) по возрастанию пары `<стоимость, мощность>`.
2. Сортируем массив классов ($classes$) по возрастанию минимальной требуемой мощности кондиционеров.

«Кондиционеры»

Идея решения на 100 баллов

Сортировка + «жадный алгоритм».

Сложность $O(M * \log(M) + N * \log(N))$

1. Сортируем массив с кондиционерами $(conds)$ по возрастанию пары `<стоимость, мощность>`.
2. Сортируем массив классов $(classes)$ по возрастанию минимальной требуемой мощности кондиционеров.

1. Пусть $lastSelectedIdx$ — индекс минимального выбранного кондиционера.
   Изначально равен 0 (или 1, если массив от 1).
2. Идём по отсортированному массиву классов.
3. Для каждого класса либо подходит кондиционер $conds[lastSelectedIdx]$, либо какой то «ближайший» справа в массиве $conds$ более мощный.
4. Если кондиционер подходит ($conds[lastSelectedIdx].power \geq classes[i]$), то добавляем к ответу его стоимость и переходим к следующему классу.
5. Если слабоват, то
   `lastSelectedIdx := lastSelectedIdx + 1`
   и переходим к следующему кондиционеру, оставаясь в том же классе.
6. По условию задачи всегда существует хотя бы один тип кондиционеров, который можно поставить в любой класс.

1. Пусть $lastSelectedIdx$ — индекс минимального выбранного кондиционера.
   Изначально равен 0 (или 1, если массив от 1).
2. Идём по отсортированному массиву классов.
3. Для каждого класса либо подходит кондиционер $conds[lastSelectedIdx]$, либо какой то «ближайший» справа в массиве $conds$ более мощный.
4. Если кондиционер подходит ($conds[lastSelectedIdx].power \geq classes[i]$), то добавляем к ответу его стоимость и переходим к следующему классу.
5. Если слабоват, то
   `lastSelectedIdx := lastSelectedIdx + 1`
   и переходим к следующему кондиционеру, оставаясь в том же классе.
6. По условию задачи всегда существует хотя бы один тип кондиционеров, который можно поставить в любой класс.

1. Пусть $lastSelectedIdx$ — индекс минимального выбранного кондиционера.
   Изначально равен 0 (или 1, если массив от 1).
2. Идём по отсортированному массиву классов.
3. Для каждого класса либо подходит кондиционер $conds[lastSelectedIdx]$, либо какой то «ближайший» справа в массиве $conds$ более мощный.
4. Если кондиционер подходит ($conds[lastSelectedIdx].power \geq classes[i]$), то добавляем к ответу его стоимость и переходим к следующему классу.
5. Если слабоват, то
   `lastSelectedIdx := lastSelectedIdx + 1`
   и переходим к следующему кондиционеру, оставаясь в том же классе.
6. По условию задачи всегда существует хотя бы один тип кондиционеров, который можно поставить в любой класс.

1. Пусть $lastSelectedIdx$ — индекс минимального выбранного кондиционера.
   Изначально равен 0 (или 1, если массив от 1).
2. Идём по отсортированному массиву классов.
3. Для каждого класса либо подходит кондиционер $conds[lastSelectedIdx]$, либо какой то «ближайший» справа в массиве $conds$ более мощный.
4. Если кондиционер подходит ($conds[lastSelectedIdx].power \geq classes[i]$), то добавляем к ответу его стоимость и переходим к следующему классу.
5. Если слабоват, то
   `lastSelectedIdx := lastSelectedIdx + 1`
   и переходим к следующему кондиционеру, оставаясь в том же классе.
6. По условию задачи всегда существует хотя бы один тип кондиционеров, который можно поставить в любой класс.

Вопросы по задаче?

«Конфеты»

Идея решения на 40 баллов

Сложность $O(N^2)$

1. Если $a + b + c > N$ (не сможем положить ни одну коробку), то подходит любой ящик с суммой размеров $\leq N$.
2. Пусть x, y, z — количество коробок, которых влезает по длине, ширине и высоте в коробку.
3. Перебор всевозможных значений x и y и вычислении z по формуле
$$z = \left\lfloor \frac{n - ax - by}{c} \right\rfloor$$
4. Проверяем, стало ли произведение $x \times y \times z$ больше текущего.

«Конфеты»

Идея решения на 40 баллов
Сложность $O(N^2)$

1. Если $a+b+c>N$ (не сможем положить ни одну коробку), то подходит любой ящик с суммой размеров $\leq N$.
2. Пусть x, y, z — количество коробок, которых влезает по длине, ширине и высоте в коробку.
3. Перебор всевозможных значений x и y и вычислении z по формуле
$$z = \left\lfloor \frac{n-ax-by}{c} \right\rfloor$$
4. Проверяем, стало ли произведение $x \times y \times z$ больше текущего.

«Конфеты»

Идея решения на 70 баллов
Динамическое программирование
Сложность $O(N)$

1. Пусть p, q, r — количество коробок, которое помещается в ящик в длину, ширину и высоту.
2. Нам нужно максимизировать произведение $p \times q \times r$ при ограничении $a \times p + b \times q + c \times r \leq N$.
3. Создадим массив $vars$ длины n, где каждый элемент $vars[i]$ содержит запись с полями p, q, r, такими, что $a \times p + b \times q + c \times r = i$ и при этом $p \times q \times r$ максимально.

«Конфеты»

Идея решения на 70 баллов

Динамическое программирование

Сложность $O(N)$

1. Пусть p, q, r — количество коробок, которое помещается в ящик в длину, ширину и высоту.
2. Нам нужно максимизировать произведение $p \times q \times r$ при ограничении $a \times p + b \times q + c \times r \leq N$.
3. Создадим массив $vars$ длины n, где каждый элемент $vars[i]$ содержит запись с полями p, q, r, такими, что $a \times p + b \times q + c \times r = i$ и при этом $p \times q \times r$ максимально.

Чтобы найти $vars[i]$, рассмотрим элементы $vars[i-a]$, $vars[i-b]$ и $vars[i-c]$ и выберем максимальный.

- Для $vars[i-a]$ мы можем увеличить p на 1, тогда $a * (vars[i-a].p + 1) + b * vars[i-a].q + c * vars[i-a].r$ будет равно i.
- Для $vars[i-b]$ мы можем увеличить q на 1, тогда $a * vars[i-a].p + b * (vars[i-a].q + 1) + c * vars[i-a].r$ будет равно i.
- Для $vars[i-c]$ мы можем увеличить r на 1, тогда $a * vars[i-a].p + b * vars[i-a].q + c * (vars[i-a].r + 1)$ будет равно i.

Заполнив массив $vars$, находим в нём такой элемент k, для которого $p \times q \times r$ максимально.

Чтобы найти $vars[i]$, рассмотрим элементы $vars[i-a]$, $vars[i-b]$ и $vars[i-c]$ и выберем максимальный.

- Для $vars[i-a]$ мы можем увеличить p на 1, тогда $a*(vars[i-a].p+1)+b*vars[i-a].q+c*vars[i-a].r$ будет равно i.
- Для $vars[i-b]$ мы можем увеличить q на 1, тогда $a*vars[i-a].p+b*(vars[i-a].q+1)+c*vars[i-a].r$ будет равно i.
- Для $vars[i-c]$ мы можем увеличить r на 1, тогда $a*vars[i-a].p+b*vars[i-a].q+c*(vars[i-a].r+1)$ будет равно i.

Заполнив массив $vars$, находим в нём такой элемент k, для которого $p \times q \times r$ максимально.

Чтобы найти $vars[i]$, рассмотрим элементы $vars[i-a]$, $vars[i-b]$ и $vars[i-c]$ и выберем максимальный.

- Для $vars[i-a]$ мы можем увеличить p на 1, тогда $a*(vars[i-a].p+1)+b*vars[i-a].q+c*vars[i-a].r$ будет равно i.
- Для $vars[i-b]$ мы можем увеличить q на 1, тогда $a*vars[i-a].p+b*(vars[i-a].q+1)+c*vars[i-a].r$ будет равно i.
- Для $vars[i-c]$ мы можем увеличить r на 1, тогда $a*vars[i-a].p+b*vars[i-a].q+c*(vars[i-a].r+1)$ будет равно i.

Заполнив массив $vars$, находим в нём такой элемент k, для которого $p\times q\times r$ максимально.

«Конфеты»

Идея решения на 100 баллов

Сложность $O(\sqrt[3]{n^2})$

1. Полагаем, что $a \geq b \geq c$.
   Если это не так, то отсортируем их по убыванию, но запомним - кто был каким.
2. Берём x в $\left[\left\lfloor\frac{n}{3a}\right\rfloor - \sqrt[3]{n}, \left\lfloor\frac{n}{3a}\right\rfloor + \sqrt[3]{n}\right]$.
3. Берём y в $\left[\left\lfloor\frac{n-ax}{2b}\right\rfloor - \sqrt[3]{n}, \left\lfloor\frac{n-ax}{2b}\right\rfloor + \sqrt[3]{n}\right]$.
4. Получаем $z = \left\lfloor\frac{n-ax-by}{c}\right\rfloor$ и проверяем, стало ли произведение $x \times y \times z$ больше текущего.

«Конфеты»

Идея решения на 100 баллов

Сложность $O(\sqrt[3]{n^2})$

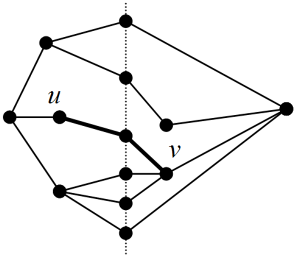
1. Полагаем, что $a \geq b \geq c$.
   Если это не так, то отсортируем их по убыванию, но запомним - кто был каким.
2. Берём x в $\left[\left\lfloor\frac{n}{3a}\right\rfloor - \sqrt[3]{n}, \left\lfloor\frac{n}{3a}\right\rfloor + \sqrt[3]{n}\right]$.
3. Берём y в $\left[\left\lfloor\frac{n-ax}{2b}\right\rfloor - \sqrt[3]{n}, \left\lfloor\frac{n-ax}{2b}\right\rfloor + \sqrt[3]{n}\right]$.
4. Получаем $z = \left\lfloor\frac{n-ax-by}{c}\right\rfloor$ и проверяем, стало ли произведение $x \times y \times z$ больше текущего.

Вопросы по задаче?

«Волонтёры»

Формализация задачи

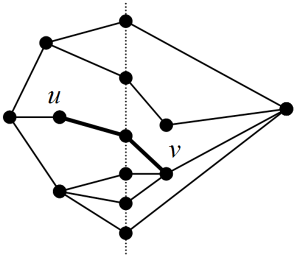
- Члены научного комитета образуют дерево, члены технического комитета — тоже дерево.
  У этих деревьев общие листья — волонтеры.
- Требуется сосчитать количество пар вершин (u, v), где u — вершина из левого дерева, v — из правого, и между u и v есть путь, идущий всегда слева направо.



«Волонтёры»

Формализация задачи

- Члены научного комитета образуют дерево, члены технического комитета — тоже дерево.
  У этих деревьев общие листья — волонтеры.
- Требуется сосчитать количество пар вершин (u, v), где u — вершина из левого дерева, v — из правого, и между u и v есть путь, идущий всегда слева направо.

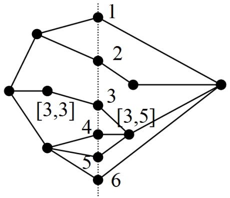


«Волонтёры»

Идея решения на 50 баллов

Сложность $O(M * K + N + M + K)$

1. Для каждой вершины каждого дерева найдём номера листьев, которые ей соответствуют.
2. Любой вершине соответствуют последовательные номера листьев без пропусков.
3. Диапазон листьев будем хранить в каждой вершине как два числа $v1$ и $v2$ — начальный номер листа и конечный.
4. Перебираем все пары вершин (u, v) и определяем, имеют ли отнесенные к ним отрезки общее пересечение.

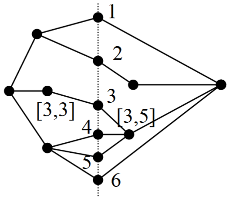


«Волонтёры»

Идея решения на 50 баллов

Сложность $O(M * K + N + M + K)$

1. Для каждой вершины каждого дерева найдём номера листьев, которые ей соответствуют.
2. Любой вершине соответствуют последовательные номера листьев без пропусков.
3. Диапазон листьев будем хранить в каждой вершине как два числа $v1$ и $v2$ — начальный номер листа и конечный.
4. Перебираем все пары вершин (u, v) и определяем, имеют ли отнесенные к ним отрезки общее пересечение.

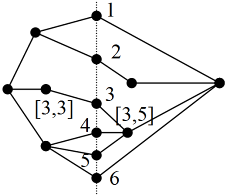


«Волонтёры»

Идея решения на 50 баллов

Сложность $O(M * K + N + M + K)$

1. Для каждой вершины каждого дерева найдём номера листьев, которые ей соответствуют.
2. Любой вершине соответствуют последовательные номера листьев без пропусков.
3. Диапазон листьев будем хранить в каждой вершине как два числа $v1$ и $v2$ — начальный номер листа и конечный.
4. Перебираем все пары вершин (u, v) и определяем, имеют ли отнесенные к ним отрезки общее пересечение.

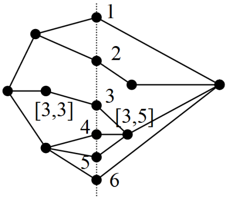


«Волонтёры»

Идея решения на 50 баллов

Сложность $O(M * K + N + M + K)$

1. Для каждой вершины каждого дерева найдём номера листьев, которые ей соответствуют.
2. Любой вершине соответствуют последовательные номера листьев без пропусков.
3. Диапазон листьев будем хранить в каждой вершине как два числа $v1$ и $v2$ — начальный номер листа и конечный.
4. Перебираем все пары вершин (u, v) и определяем, имеют ли отнесенные к ним отрезки общее пересечение.



Поиск $v1$ и $v2$ для вершин дерева

1. Найти этот номера очень легко с помощью поиска в глубину.
2. Пусть мы рассматриваем вершину p.
3. Посетим рекурсивно всех её детей, в результате вычислятся их диапазоны номеров листьев.
4. Возьмём среди этих номеров самый маленький и самый большой — это и будет номера $v1$ и $v2$ для вершины p.

Поиск $v1$ и $v2$ для вершин дерева

1. Найти этот номера очень легко с помощью поиска в глубину.
2. Пусть мы рассматриваем вершину p.
3. Посетим рекурсивно всех её детей, в результате вычислятся их диапазоны номеров листьев.
4. Возьмём среди этих номеров самый маленький и самый большой — это и будет номера $v1$ и $v2$ для вершины p.

Поиск $v1$ и $v2$ для вершин дерева

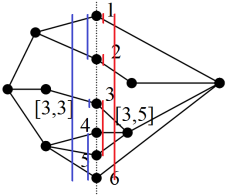
1. Найти этот номера очень легко с помощью поиска в глубину.
2. Пусть мы рассматриваем вершину p.
3. Посетим рекурсивно всех её детей, в результате вычислятся их диапазоны номеров листьев.
4. Возьмём среди этих номеров самый маленький и самый большой — это и будет номера $v1$ и $v2$ для вершины p.

«Волонтёры»

Улучшение решения до 100 баллов

Сложность $O((M+K)*\log(M+K)+N+M+K)$

1. Полученные диапазоны номеров листьев в вершинах можно рассматривать как отрезки на координатной прямой.
2. Обозначим их в левом дереве синими, а в правом - красными.
3. Нам нужно найти количество пар синих и красных отрезков, которые пересекаются.

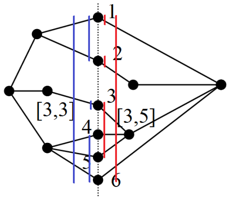


«Волонтёры»

Улучшение решения до 100 баллов

Сложность $O((M+K)*\log(M+K)+N+M+K)$

1. Полученные диапазоны номеров листьев в вершинах можно рассматривать как отрезки на координатной прямой.
2. Обозначим их в левом дереве синими, а в правом - красными.
3. Нам нужно найти количество пар синих и красных отрезков, которые пересекаются.

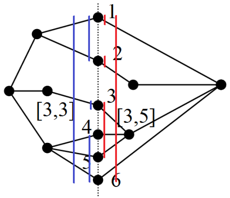


«Волонтёры»

Улучшение решения до 100 баллов

Сложность $O((M+K)*\log(M+K)+N+M+K)$

1. Полученные диапазоны номеров листьев в вершинах можно рассматривать как отрезки на координатной прямой.
2. Обозначим их в левом дереве синими, а в правом - красными.
3. Нам нужно найти количество пар синих и красных отрезков, которые пересекаются.



1. Возьмём координаты начал и концов всех отрезков и занесём их в массив.
   Каждый элемент массива будет хранить три поля:
   - координату
   - признак (начало или конец отрезка)
   - цвет (красный или синий)
2. Отсортируем массив:
   1. Сортируем по возрастанию координаты
   2. Для одинаковых координат сортируем по типу (сначала открывающие, потом закрывающие)
   3. Сортируем по цвету (сначала синие, потом красные)
3. Заведём две переменные:
   red — количество открытых красных отрезков,
   $blue$ — количество открытых синих.

1. Возьмём координаты начал и концов всех отрезков и занесём их в массив.
   Каждый элемент массива будет хранить три поля:
   - координату
   - признак (начало или конец отрезка)
   - цвет (красный или синий)
2. Отсортируем массив:
   1. Сортируем по возрастанию координаты
   2. Для одинаковых координат сортируем по типу (сначала открывающие, потом закрывающие)
   3. Сортируем по цвету (сначала синие, потом красные)
3. Заведём две переменные:
   red — количество открытых красных отрезков,
   $blue$ — количество открытых синих.

1. Возьмём координаты начал и концов всех отрезков и занесём их в массив.
   Каждый элемент массива будет хранить три поля:
   - координату
   - признак (начало или конец отрезка)
   - цвет (красный или синий)
2. Отсортируем массив:
   1. Сортируем по возрастанию координаты
   2. Для одинаковых координат сортируем по типу (сначала открывающие, потом закрывающие)
   3. Сортируем по цвету (сначала синие, потом красные)
3. Заведём две переменные:
   red — количество открытых красных отрезков,
   $blue$ — количество открытых синих.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

1. Обозначим за $answer$ ответ в задаче.
2. Пройдёмся по массиву:
   Встав на очередной элемент, рассмотрим возможные случаи:
   1. Если открывается синий отрезок:
      `blue := blue + 1`
      `answer := answer + red` (потому что этот отрезок конфликтует со всеми открытыми на данный момент красными отрезками).
   2. Если открывается красный отрезок:
      `red := red + 1`
      `answer := answer + blue`
   3. Если отрезок закрывается:
      `blue := blue - 1` либо
      `red := red - 1` соответственно.

Вопросы по задаче?

Спасибо за внимание!