XXVII командный чемпионат школьников Санкт-Петербурга по программированию

27 октября 2019 года

Задача A
Прибытие короля

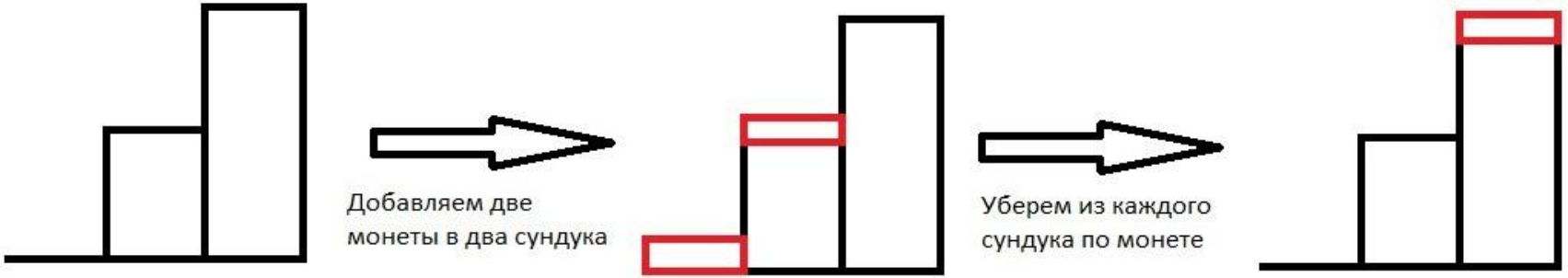
Автор задачи:
Геннадий Короткевич
Разработка задачи:
Дмитрий Гнатюк

Постановка задачи

- Дано три сундука с монетами
- За одну операцию можно кинуть по монете в два сундука
- Нужно найти, сколько операций надо совершить, чтобы уравнять количество монет в сундуках

Решение задачи

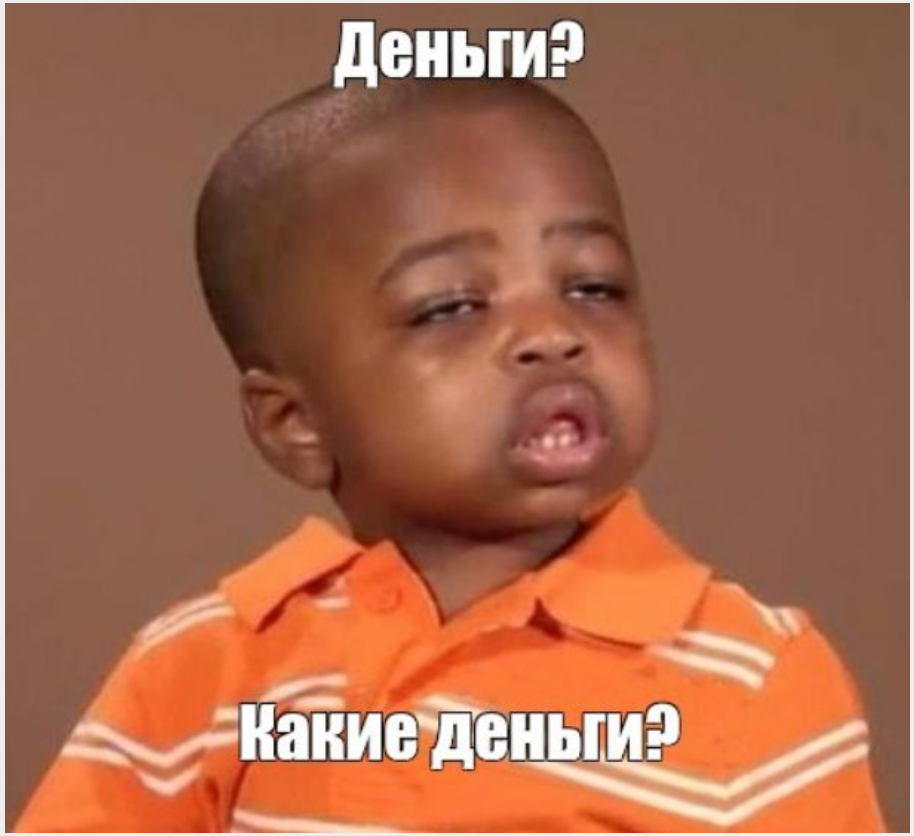
- Отсортируем сундуки
- Кинуть по монете в два сундука это то же самое, что и забрать одну монету из третьего сундука



- Ответ = $(a + b + c) - 3 \cdot \min(a, b, c)$

Задача B Кассовый разрыв

Автор задачи:
Федор Царев
Разработка задачи:
Даниил Орешников

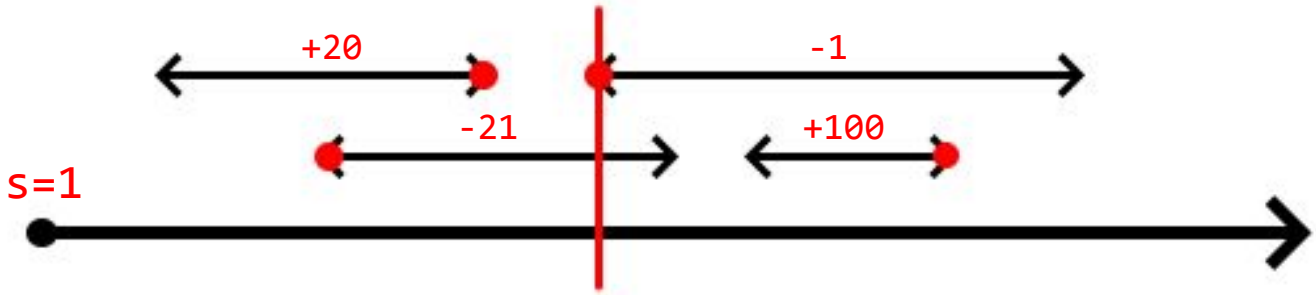


Постановка задачи

- Есть денежный счёт, на котором s денег
- В некоторые интервалы времени могут поступить операции, изменяющие количество денег на счету
- Надо сказать, существует ли такой порядок поступления операций, при котором количество денег на счету в какой-то момент должно стать отрицательным

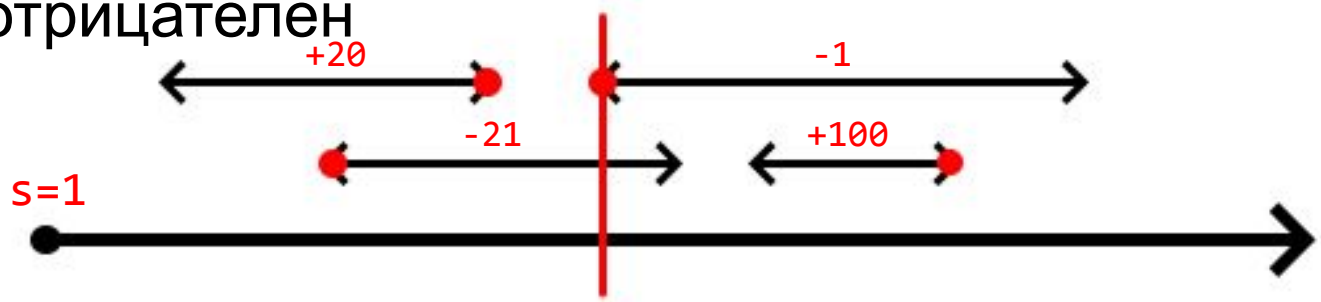
Решение задачи

- Рассмотрим ситуацию, в которой операции, уменьшающие счёт, пришли в первый возможный день, а увеличивающие счет – в последний
- Внутри одного дня расположим операции так, чтобы все уменьшающие шли раньше всех увеличивающих



Решение задачи

- Если возможна ситуация, в которой счёт становится отрицательным, то в описанной выше последовательности счёт также станет отрицательным в какой-то момент
- Достаточно расположить операции таким образом и проверить, что после каждой из них счёт неотрицателен



Решение задачи

- Можно упорядочить события любой сортировкой и пройтись по ним в полученном порядке
- Ограничения задачи позволяли посчитать суммарный счёт до каждой операции даже за время $O(nm)$, просто для каждой операции пройдясь по всем остальным и просуммировав те, у которых время меньше либо равно текущей

Задача C
Реактивные поезда

Автор задачи:
Алексей Плешаков
Разработка задачи:
Алексей Плешаков

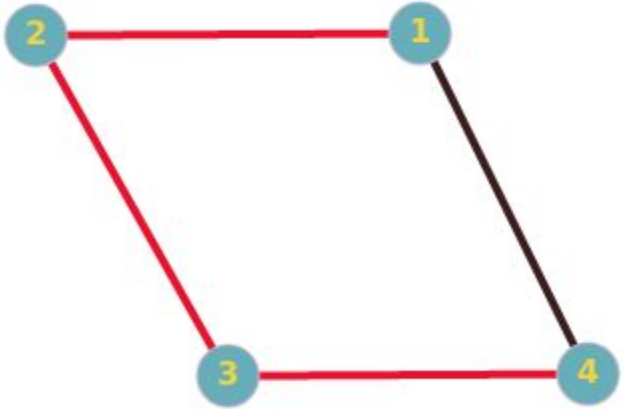
Постановка задачи

- Даны *n* городов. На них определено отношение достижимости
- Некоторые пары городов «дружат»
- Нужно уметь добавлять новые пары друзей, новые пути между городами и отвечать, сколько друзей конкретного города достижимы

Решение задачи

- Давайте формализуем задачу. Скажем, что у нас есть граф на n городов и рёбра двух типов. Пусть дружественную связь (a, b) задаёт чёрное ребро, а рейс (a, b) красное
- Нужно уметь добавлять рёбра и понимать, сколько соседей конкретной вершины по чёрным рёбрам достижимы по красным, то есть лежат в той же компоненте связности по красным рёбрам

Решение задачи



Ответ для первой и четвертой вершины — единица, для остальных — ноль

Решение задачи

- Важное замечание: так как рёбра только добавляются, а не удаляются, очень удобно использовать систему непересекающихся множеств (СНМ) для поддержания компонент связности
- Обозначим компоненту, в которой лежит `v`, за `comp[v]`

Решение задачи

- Для каждой вершины будем хранить текущий ответ `ans[v]`
- На запрос «`? v`» просто выводим `ans[v]`
- Если добавляется пара друзей `(a, b)`, нужно увеличить `ans[a]` и `ans[b]` на единицу, если `comp[a] = comp[b]`
- Самое сложное — научиться добавлять рейс `(a, b)`

Решение задачи

- При добавлении рейса две компоненты x, y могут слиться в одну. Давайте рассмотрим меньшую из них, пусть это компонента x
- Ответ изменится в не более чем $2\sum$`friends[v]` вершинах (`friends[v]` — количество друзей v, v лежит в x)
- У вершин, которые не лежат в x или y, ответ измениться не мог

Решение задачи

- Давайте честно переберём все вершины в x и обновим ответ вдоль каждого чёрного ребра из них.
- Почему это работает быстро?
- Рассмотрим вершину v. Мы будем перебирать всех её соседей не более чем `log(n)` раз, так как каждый раз после такого перебора размер `comp[v]` увеличивается не менее, чем вдвое

Решение задачи

- Итоговая асимптотика:

$$O(q + m + (k+q) \log(n) \cdot \alpha(n))$$

где $\alpha(n)$ — обратная функция Аккермана

Задача D
Нарезка пиццы

Авторы задачи:
Степан Филиппов
Разработка задачи:
Арсений Кириллов

Постановка задачи

- Есть круглая пицца
- *n* человек хотят съесть сектор пиццы с углом ang_i
- Можно делать разрез по радиусу или по диаметру
- Нужно за минимальное количество разрезов получить сектора со всеми нужными углами

Решение задачи

- Можно доказать, что либо не будет диаметральных разрезов, и тогда будет не более одного сектора, который не будет взят ни одним человеком
- Либо будет существовать диаметральный разрез, и тогда таких секторов будет не более двух, и они будут прилегать к одному разрезу по диаметру с разных сторон
- Для подробностей читайте текстовый разбор

Решение задачи

- Если разрезов по диаметру нет, то ответом будет либо n либо $n + 1$
- Иначе используем ДП по подмножествам
- Посчитаем $dp_{mask,sum}$ – минимальное количество разрезов, чтобы отдать секторы людям из маски, и ровно sum градусов заняты по часовой стрелке от диаметрального разреза
- Чтобы подсчитать эту динамику, переберем новый угол, и сторону, на которой будет этот угол

Задача E
Единственное решение

Автор задачи:
Геннадий Короткевич
Разработка задачи:
Рамазан Рахматуллин

Постановка задачи

- Даны n целых чисел a_i от -1 до 1, хотя бы одно не равно 0
- Требуется найти такие целые x_i и $m < 2^n$, чтобы равенство

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \equiv 0 \pmod{m}$$

выполнялось только при данных a_i или при коэффициентах, равных $-a_i$

Решение задачи

- Пусть набор a содержит k нулей, l единиц и $n - k - l$ минус единиц
- Пронумеруем a так, чтобы выполнялось
  - $a_{i\_0} = a_{i\_1} = \ldots = a_{i\_(k-1)} = 0$
  - $a_{i\_k} = a_{i\_(k+1)} = \ldots = a_{i\_(k+l-1)} = 1$
  - $a_{i\_(k+l)} = a_{i\_(k+l+1)} = \ldots = a_{i\_n} = -1$
- Рассмотрим тогда
  - $x_{i\_t} = 2^t$ для всех $t < k + l$
  - $x_{i\_t} = -2^t$ для всех $t \geq k + l$
  - $m = 2^n - 2^k$

Решение задачи

- Заметим, что m делится на 2^k
- Заметим также, что все 2^t при $t < k$ обязаны иметь коэффициент 0 при них, потому что сумма 2^t с любыми коэффициентами при $t \geq k$ будет делиться на 2^k, а суммой степеней двойки, меньших k, нельзя набрать что-то делящееся на 2^k
- Также заметим, что с любыми коэффициентами сумма оставшихся чисел не будет превышать $2^n - 2^k$ по модулю, так как
  - $|2^k \pm \ldots \pm 2^{n-1}| \leq |2^k| + \ldots + |2^{n-1}| = 2^n - 2^k$

Решение задачи

- $|2^k \pm \ldots \pm 2^{n-1}| \leq 2^n - 2^k = m$
- Таким образом, поскольку различными степенями двойки с коэффициентами -1 и 1 нельзя набрать 0, то чтобы в сумме получилось число, делящееся на m, сумма обязана быть равна m или $-m$
- Есть единственный способ получить m и единственный способ получить $-m$ — взять такие a_i, чтобы все $a_i x_i$ были одного знака
- Таким образом, либо при всех x_i обязаны стоять в точности a_i, либо при всех x_i обязаны стоять $-a_i$

Задача F
Метро 2345

Авторы задачи:
Дмитрий Гнатюк,
Андрей Станкевич
Разработка задачи:
Григорий Шовкопляс

Постановка задачи

- Три линии метро
  - Между соседними станциями поезд едет одинаково за время, определённое линией
- Есть пересадки между линиями за время *d*

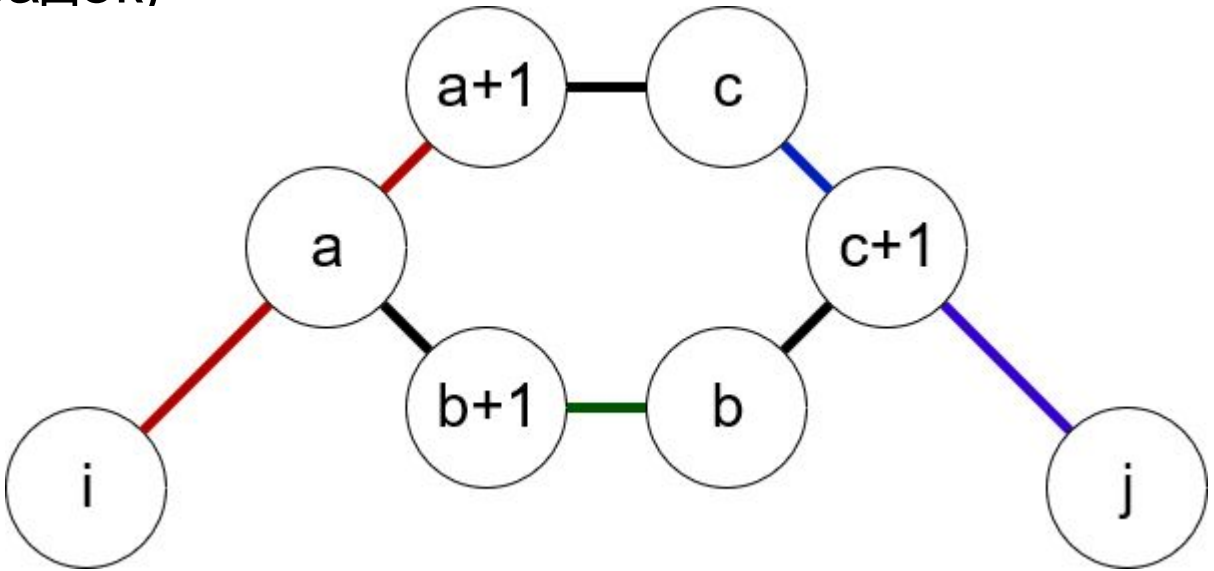
- Нужно доехать от стартовой станции до конечной за минимальное возможное время

Решение задачи

- Два способа:
  - Горе от ума
  - Просто, но аккуратно

Решение задачи («Горе от ума»)

- Ровно восемь «интересных» станций (стартовая и конечная станции, а также двойные станции пересадок)



Решение задачи («Горе от ума»)

- Можно построить граф со следующими рёбрами:
  - Между станциями пересадок
  - Между соседними станциями одной линии
  - Между стартовой вершиной и вершинами пересадок на её линии
  - Между конечной вершиной и вершинами пересадок на её линии
  - Ребро между стартовой и конечной вершинами, если они располагаются на одной линии
- Поиск кратчайшего пути (например алгоритмом Флойда)

Решение задачи («Просто, но аккуратно»)

- Есть всего четыре способа движения:
  - Стартовая и конечная станция на одной линии:
    - Ноль пересадок
    - Три пересадки
  - Стартовая и конечная станция на разных линиях:
    - Одна пересадка
    - Две пересадки
- Остается лишь аккуратно рассмотреть данные случаи и выбрать минимальный

Задача G
Переполнение

Автор задачи:
Даниил Орешников
Разработка задачи:
Даниил Орешников

Постановка задачи

- Есть n чисел a_i и модуль m
- Требуется найти два непересекающихся набора чисел, хотя бы один из которых не пустой, которые дают одинаковый остаток по модулю m
- Или сообщить, что таких наборов не существует

Решение задачи (нет)

- Пусть числа на одной чаше весов берутся в сумму со знаком «+», а на другой — со знаком «-»
- Задача сводится к тому, чтобы получить сумму в таком наборе, делящуюся на m
- Переберём все троичные маски для нашего набора чисел, отвечающие за то, с каким знаком взять число в сумму

1	20	5	17	9	3
1	-1	0	1	0	-1

= 1 - 20 + 17 - 3 = -5

Решение задачи

- Такое решение не проходит в лимит времени
- Вместо этого переберем двоичные маски наборов (взять со знаком «+» или не взять)
- Если у двух наборов одинаковый остаток, можно взять один из них с плюсом, а второй с минусом

1	20	5	17	9	3
1	0	1	1	0	1
0	1	0	0	0	0

$= 1 + 5 + 17 + 3 = 26 \equiv 2 \pmod 6$

$= 20 \equiv 2 \pmod 6$

Решение задачи

- Решение за $O(2^n n + m)$ всё ещё плохо укладывается в ограничения, можно соптимизировать до $O(2^n + m)$
- Для этого достаточно уметь считать сумму, соответствующую маске, за $O(1)$
- Для маски `mask` рассмотрим
  - `prev = mask & (mask - 1)`
  - `last = mask ^ prev`
- Это разбиение маски на её последнюю единицу и всё оставшееся, сумма на маске равна сумме на `prev` и `last`

Альтернативное решение

- Разобьём массив на две половины длины $n / 2$
- В каждой за $O(3^{n/2})$ перебором троичных масок (см. первый слайд) найдем все остатки по модулю m, которые можно получить
- Если в первой половине можно получить остаток x, а во второй $m - x$ для некоторого x (либо если в какой-то можно получить остаток 0), то у нас есть ответ
- Время работы ~ $O(1.73^n n + m)$

Задача H
Проверка теста

Автор задачи:
Андрей Станкевич
Разработка задачи:
Степан Филиппов

Постановка задачи

- Даны правильные ответы на тест и ответы студентов на этот тест
- Две работы похожи, если в каждой из них больше половины правильных ответов совпадают с ответами в другой работе, и больше половины неправильных ответов совпадают с ответами в другой работе.
- Требуется вывести все пары похожих работ

Решение задачи

- Посчитаем количество правильных и неправильных ответов в каждой работе
- Для каждой пары работ посчитаем количество одинаковых правильных ответов и количество одинаковых неправильных ответов
- Используя эту информацию, проверить эту пару работ на похожесть
- Всего пар работ $m(m - 1)/2$, для каждой пары просматриваем все ответы в этих работах
- Асимптотика $O(m^2 * n)$

Задача I
Магический Фокус

Авторы задачи:
Артем Васильев,
Разработка задачи:
Дмитрий Саютин

Постановка задачи

- Загадана перестановка $p_1, p_2, \ldots, p_n$
- Дано множество множеств
  $\{\{p_1,p_2,p_3\}, \{p_2,p_3,p_4\}, \ldots, \{p_{n-1},p_{n-2},p_1\}, \{p_n,p_1,p_2\}\}$

- Всё перемешано, восстановить перестановку

Решение

- Выберем произвольную тройку $\{a, b, c\}$ из входных данных

- Переберём все возможные способы её упорядочить:
- $[a, b, c]$, $[a, c, b]$, $[b, a, c]$, etc

- Попробуем восстановить в каждом случае

Решение

- Пусть уже восстановили $[a, b, c, d, e]$, а оставшаяся часть выглядит как $[x, y, z, ..]$
- Какие тройки содержат $\{d, e\}$?

- $\{c, d, e\}$ и $\{d, e, x\} \rightarrow$ узнали следующий элемент
- Если в какой-то момент не получается продолжить, попробуем другой способ упорядочить изначальную тройку

Решение («для зануд»)

- Сохранить все x, которые присутствуют вместе с $\{d, e\}$?

- `map<pair<int, int>, vector<int>>`

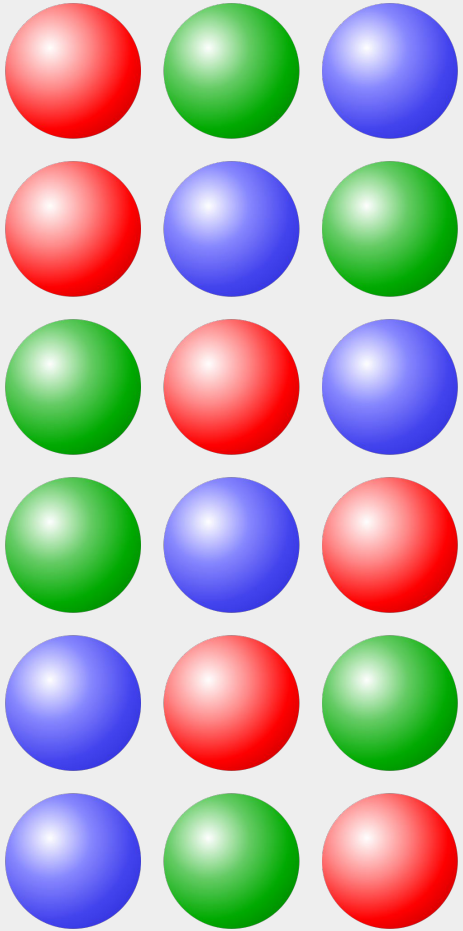
Решение («для зануд»)

- При $n = 3$ и $n = 4$ ответом является любая перестановка.

- Может быть крайним случаем в некоторых решениях.

Задача J
Суперперестановки

Авторы задачи:
Артем Васильев,
Разработка задачи:
Артем Васильев, Роман Коробков



Постановка задачи

- *Суперперестановкой* называется минимальная по длине последовательность, которая содержит все перестановки чисел от 1 до n, как подотрезки
- Описан способ генерации *суперперестановок*
- Нужно найти вхождение заданной перестановки в *суперперестановку*

Решение

- Посмотрим на первые *суперперестановки*:
- S_2 = 1 2 1
- S_3 = 1 2 (3 1 2) 1 (3 2 1)
- S_4 = 1 2 3 (4 1 2 3) 1 (4 2 3 1) 2 (4 3 1 2) 1 3 (4 2 1 3) 2 (4 1 3 2) 1 (4 3 2 1)
- Заметим, что подотрезки длины n - 1 слева и справа от числа n одинаковые. Это значит, что все циклические сдвиги перестановки стоят рядом.

Решение

- S_3 = 1 2 3 1 2 1 3 2 1
- S_4 = 1 2 3 4 1 2 3 1 4 2 3 1 2 4 3 1 2 1 3 4 2 1 3 2 4 1 3 2 1 4 3 2 1
- Чтобы найти позицию перестановки *P*, сдвинем её циклически, чтобы максимум оказался в начале, затем удалим максимум, получим перестановку *Q*
- Найдем позицию *Q* в *суперперестановке* меньшего порядка. Затем будем пересчитывать позицию *P*
- Получим примерную позицию *P* (начало зеленого отрезка), а затем точную позицию *P* определим из расположения числа 4 относительно 1, 2, 3

Решение

- Пусть cnt_n – количество чисел равных N, которые в *суперперестановке* S_n находятся раньше нашей перестановки, $index_x$ — позиция x в полученной перестановке из чисел 1 до x, pos_n – позиция перестановки в *суперперестановке* порядка n
- Тогда
  - $cnt_n = cnt_{n-1} \times n + ((n - 1) - 1 - index_{n-1})$
  - $pos_n = pos_{n-1} + n \times cnt_n + (n - 1 - index_n)$

Решение

- Найти позицию максимального элемента
- Циклически сдвинуть перестановку так, чтобы максимальный элемент стал первым
- Удалить максимальный элемент

- Это можно реализовать явно с помощью Декартова дерева, либо посчитать $\texttt{index}_n$, используя запросы изменения элемента и суммы на отрезке

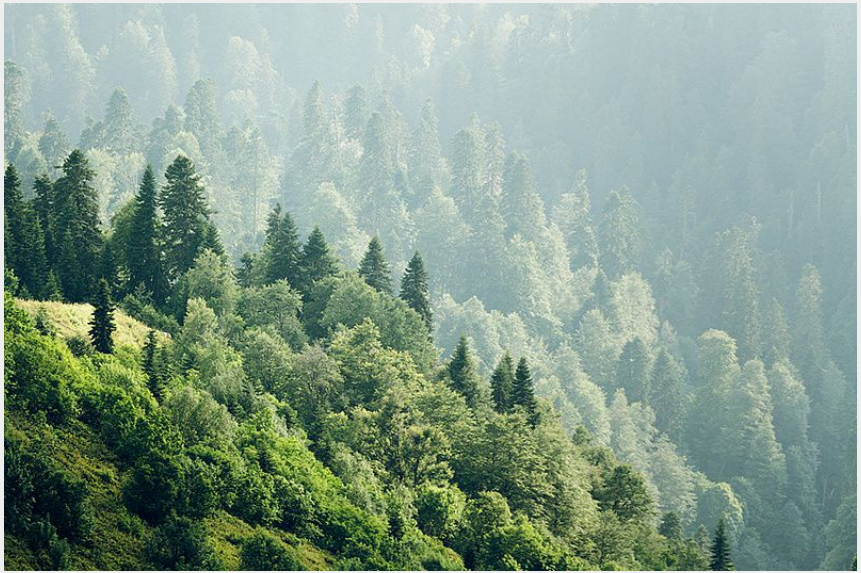
- Итоговое время работы $O(n \log n)$

Больше информации

- по ссылке http://gg.gg/superper

Задача K
Подготовка тестов

Автор задачи:
Николай Будин
Разработка задачи:
Николай Будин



Постановка задачи

- Дан массив
- Нужно найти количество подотрезков, являющихся корректными мультитестами
- Мультитест состоит из нескольких тестов
- Каждый тест имеет вид $m\ a_1\ b_1\ \ldots\ a_m\ b_m$
- Ребра $(a_1, b_1)\ \ldots\ (a_m, b_m)$ задают ацикличный граф

Решение

- Пусть позиция начала теста — i
- Тогда ребра $(a_{i+1}, a_{i+2}) \dots (a_{i+ai \cdot 2-1}, a_{i+ai \cdot 2})$ должны образовывать ацикличный граф
- $e_{0,i} = (a_{i \cdot 2}, a_{i \cdot 2+1})$
- $e_{1,i} = (a_{i \cdot 2+1}, a_{i \cdot 2+2})$
- Если i чётно, то отрезок ребер $e_{1,j}$ должен образовывать ацикличный граф
- Если i нечётно — отрезок $e_{0,j}$

Решение

- Дан массив рёбер и набор l_i, r_i
- Надо проверять, образует ли отрезок рёбер от l_i до r_i ацикличный граф
- Стандартная задача, подробнее в текстовом разборе

Решение

- Для каждой позиции, в ней начинается максимум один корректный тест
- `dp[i] = dp[i + a`$_i$ `· 2 + 1] + 1`, если тест с позиции `i` корректен
- Иначе, `dp[i] = 0`

```
printf("%s\n", "Спасибо за внимание")
```

Материалы олимпиады

http://nerc.itmo.ru/school