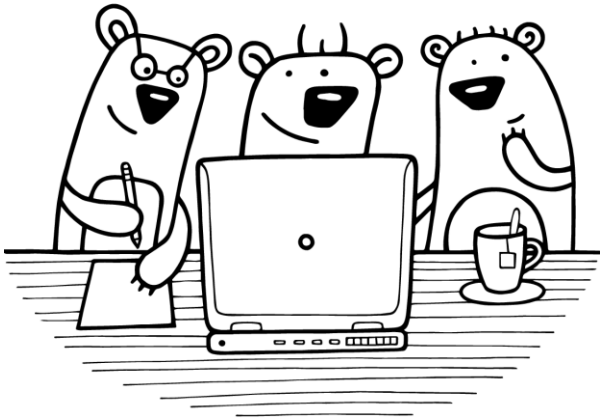


Девятнадцатая открытая Всероссийская командная олимпиада школьников по программированию

9 декабря 2018 года

Задача A
Объединение

Автор задачи:
Игорь Мамай
Разработка задачи:
Арсений Кириллов

Постановка задачи

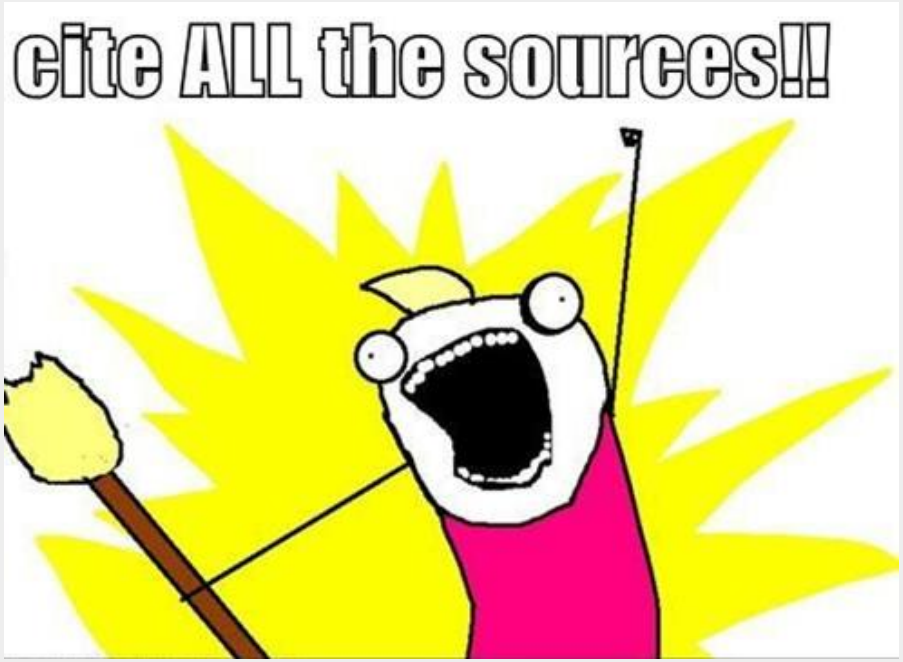
- Есть n компаний, в i-й компании m_i сотрудников
- Можно повысить зарплату всем сотрудникам одной компании
- Нужно сделать так, чтобы во всех компаниях максимальная зарплата была одинаковая

Решение задачи

- Пусть f_i — максимальная зарплата сотрудников i-й компании
- Пусть x — максимальная зарплата во всех компаниях в конце
  - Для всех i: $x \geqslant f_i$
- Всем сотрудникам i-й компании повысят зарплату на $x - f_i$
- Суммарно зарплату повысят на $\sum (x - f_i) \cdot m_i$
  - Нужно сделать x как можно меньше
  - При этом x должен быть не меньше каждого из f_i
  - $x = \max f_i$

Задача B
Эксперт LaTeX

Автор задачи:
Михаил Пядеркин
Разработка задачи:
Григорий Шовкопляс



Постановка задачи

- Дан текст, в котором есть ссылки на использованные источники
  - `\cite{<reference>}`
- Дан список используемых источников
  - `\begin{bibliography}{99}`
  - `\bibitem{<reference>}`
  - `. . .`
  - `\end{bibliography}`
- Проверить корректность списка и составить корректный, если данный неверен

Решение задачи

- Данная задача является представителем задач на реализацию
- Нужно применить навыки простейшей обработки текста и умение пользоваться массивами
- При разборе текста запишем в массив все ссылки в порядке следования
- При проверке списка источников на корректность нужно проверить, что имя `i`-й ссылки в списке совпадает с именем `i`-й ссылки в массиве
- Если хотя бы один раз не совпадает, то список составлен неверно

Решение задачи

- Чтобы составить корректный список, для удобства поиска нужных описаний можно использовать ассоциативный массив
  - `std::map` в C++
  - `TreeMap` в Java
  - . . .
- Однако, ограничения позволяли хранить описания источников в отдельном массиве и каждый раз выполнять линейный поиск по всем значениям

Задача C Новогодние подарки

Автор задачи:
Иван Смирнов
Разработка задачи:
Иван Смирнов

Постановка задачи

- Есть n мешков, в каждом находятся несколько различных подарков одного из m видов
- Можно переложить один подарок из одного мешка в другой
- Требуется за наименьшее число перекладываний сделать так, чтобы размеры самого большого и самого маленького мешка отличались как можно меньше, а каждый мешок содержал различные подарки

Решение задачи

- Пусть во всех мешках суммарно `s` подарков. Поймём, сколько подарков должно в итоге оказаться в каждом мешке.
- Если `s` делится на `n`, то в каждом мешке в итоге должно быть `s / n` подарков
- Иначе в `s mod n` мешках должен быть по `[s / n] + 1` подарку, а в оставшихся по `[s / n]`
- Оптимальная разность — `0` или `1`
- Обозначим последовательность оптимальных размеров за $f_1, \ldots, f_n$

Решение задачи

- Если изначально в мешках было $a_1 ... a_n$ подарков, а должно оказаться $f_1 ... f_n$, то суммарное количество перекладываний не может быть меньше $(\sum |a_i - f_i|) / 2$
- Почему так: для каждого мешка просуммируем разность его исходного и необходимого размера, каждое перекладывание уменьшает эту разность на 1 для двух мешков
- Можно доказать, что значение этой суммы минимально, когда a_i отсортированы по убыванию
- Покажем, как достичь ровно такого количества перекладываний, соблюдая условие про различные подарки

Решение задачи

- Подарки в каждом мешке будут различны всё время работы.
- Будем поддерживать два указателя `l`, `r` и перекладывать подарки из `l`-го мешка в `r`-й. Изначально `l = 0`, `r = n`
- Если в `l`-м мешке осталось f_l подарков, увеличим `l` на `1`, если в `r`-м мешке осталось f_r подарков, уменьшим `r` на `1`
- Иначе нужно переложить один подарок из `l`-го мешка в `r`-й. Это возможно, потому что в `l`-м мешке больше подарков, чем в `r`-м, значит, в нём найдётся подарок, которого в `r`-м нет
- Чтобы эффективно находить этот подарок, будем поддерживать хештаблицу со списком подарков для каждого мешка

Решение задачи

- Рассматривая `l`-й и `r`-й мешок, пройдёмся по хештаблице `l`-го и будем перекладывать найденный подарок, если его нет в хештаблице `r`-го, пока один из мешков не станет нужного размера
- Это приблизит размеры обоих мешков к желаемым
- Покажем, что суммарно мы совершим `O(s)` действий
- Для одной пары мешков рассмотрим `moves(l,r) + misses(l,r)` подарков, где `moves(l,r)` — количество перекладываний, а `misses(l,r)` — количество подарков, которые были просмотрены, но не переложены, поскольку присутствовали в `r`-м мешке

Решение задачи

- Если r-й мешок достиг нужного размера, то оценим $misses(l,r) \leqslant size(r)$ и сдвинем указатель r
- Если l-й мешок достиг нужного размера, то оценим $misses(l,r) \leqslant size(l)$ и сдвинем указатель l
- Значит, суммарное значение misses по всем парам не более s
- Суммарное значение перекладываний также не превосходит s
- Значит, суммарное время работы есть $O(s + n \log n)$, где второе слагаемое берётся из сортировки мешков по размеру (впрочем, её можно заменить на сортировку подсчётом)

Задача D
Похожие массивы

Автор задачи:
Артём Васильев
Разработка задачи:
Александра Дроздова

Постановка задачи

- Дан список пар позиций элементов массива
- Требуется построить два массива, на которых сравнения пар чисел на позициях из списка выдадут одинаковый результат.
  - первый, являющийся перестановкой чисел от 1 до n
  - второй, содержащий хотя бы одну пару одинаковых чисел

Решение задачи

- Если у нас есть в списке все пары позиций, то таких двух массивов не существует.
- Иначе рассмотрим пару позиций, которой нет в списке. Поставим на ее позиции в первом массиве 1 и 2, а во втором 1 и 1. На оставшиеся позиции в обоих массивах числа от 3 до n.

Решение задачи

- Чтобы найти такую пару позиций, сохраним все пары в ассоциативный массив (`std::unordered_map`, `HashMap`) и будем пробегаться по всем возможным парам, пока не найдём ту, которой нет, проверяя наличие за `O(1)`.
- Поскольку мы пробежим не более `m` пар прежде, чем найдем нужную, эта часть работает за `O(m)`
- Построить ответ мы можем за `O(n)`
- Итоговая асимптотика `O(m + n)`

Задача E
Конный спорт

Автор задачи:
Михаил Дворкин
Разработка задачи:
Даниил Орешников

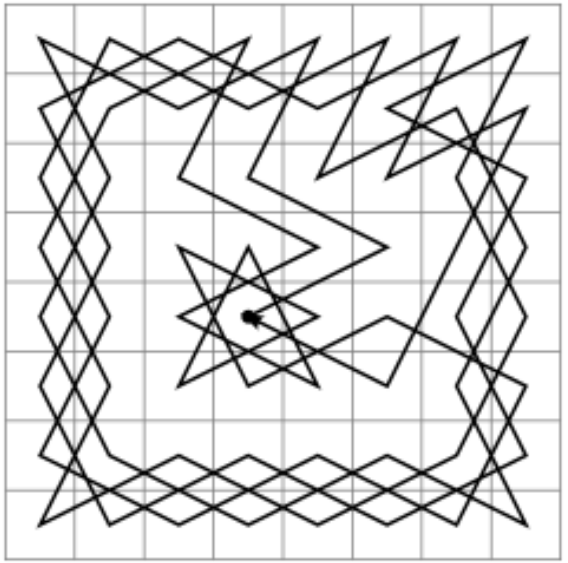
Постановка задачи

- Дано шахматное поле 8x8, и k коней на нем, которые могут ходить как обычные шахматные кони
- Требуется переставить коней так, чтобы в конце
  - [k / 8] нижних строк были заполнены конями полностью
  - k % 8 оставшихся коней стояли в первых столбцах следующей строки
- При этом ни в какой момент времени два коня не могут находиться в одной клетке поля

Решение задачи

- Построим граф с вершинами в клетках поля, соединим вершины ребром, если они достижимы за один ход коня друг из друга

- На таком графе существует Гамильтонов цикл (цикл, проходящий по всем вершинам), пример можно видеть справа
- Выберем любой Гамильтонов путь, отметим $a_1, \ldots, a_k$ — номера в пути позиций коней, $b_1, \ldots, b_k$ — номера стойл



Решение задачи

- Теперь ходом коня можно изменить индекс коня в пути на 1
- Будем перемещать коня на месте a_i на место b_i. Для этого заметим, что если конь уже на своем месте, то “слева” от него на пути поровну коней и стойл, как и “справа”, а значит больше не будет необходимости двигать этого коня
- Если есть конь, у которого a_i меньше b_i, попробуем переместить его на позицию $a_i + 1$, аналогично, если есть конь, у которого $a_i > b_i$
- Если ни одного такого коня нельзя подвинуть, то место $a_i \pm 1$ занято другим конем

Решение задачи

- Не теряя общности, рассматриваем случай, когда $a_i < b_i$, и место $a_i + 1$ занято другим конем, то есть $a_{i+1} = a_i + 1$
- Но поскольку $b_i < b_{i+1}$, то $b_{i+1} \geqslant a_i + 2$, что равносильно тому, что $a_i + 1 = a_{i+1} < b_{i+1}$
- Тогда коня на месте a_{i+1} тоже, как и a_i, надо переместить правее. Если и это невозможно сделать, то $a_{i+2} = a_{i+1} + 1$, повторим для него то же самое рассуждение
- Поскольку коней конечное число, в какой-то момент мы найдем тот, который надо переместить вправо, и при этом следующая за ним клетка свободна

Решение задачи

- Таким образом, чтобы переместить всех коней на нужные места, можно перебирать их по очереди, и пытаться переместить каждого с места a_i на место b_i
- Если место, в которое мы пытаемся переместить коня, уже занято, сдвинем коня оттуда на его место рекурсивно, и продолжим перемещать текущего

Решение задачи

- Таким образом, чтобы переместить всех коней на нужные места, можно перебирать их по очереди, и пытаться переместить каждого с места a_i на место b_i
- Если место, в которое мы пытаемся переместить коня, уже занято, сдвинем коня оттуда на его место рекурсивно, и продолжим перемещать текущего
- Более того, необязательно строить Гамильтонов путь по шахматной доске, можно просто по порядку искать пути между еще не поставленными конями и еще пустыми стойлами

Решение задачи (без Гамильтонова цикла)

- Пока не все кони стоят на местах, есть конь вне стойла, и пустое стойло
- Любым алгоритмом поиска пути в графе найдем путь между ними $a_1, \ldots, a_n$, где a_1 — позиция коня, a_n — позиция стойла
- Пусть в этом пути уже есть кони на позициях $b_1, \ldots, b_t$. Сделаем следующие перемещения:
  - $b_t \rightsquigarrow b_t + 1 \rightsquigarrow \ldots \rightsquigarrow a_n$
  - $b_{t-1} \rightsquigarrow b_{t-1} + 1 \rightsquigarrow \ldots \rightsquigarrow b_t$
  - ...
  - $a_1 \rightsquigarrow a_1 + 1 \rightsquigarrow \ldots \rightsquigarrow b_1$

Решение задачи (без Гамильтонова цикла)

- После одной такой итерации, если кони стояли на местах $a_1, b_1, \ldots, b_t$, то теперь они стоят на местах $b_1, \ldots, b_t, a_n$
- То есть теперь ровно на одного коня больше стоят в стойлах, и при этом всего было сделано $n - 1$ перемещений коней
- Спустя не более, чем k итераций этого алгоритма, все кони окажутся на местах, и суммарно это займет не более, чем $k \cdot \max(n)$ перемещений, где $\max(n)$ — максимальное n (длина пути)
- От любой клетки до любой можно добраться за не более, чем 6 шагов, поэтому ограничение в 1500 дано даже с запасом

Задача F
Как узнать свои баллы

Автор задачи:
Иван Сафонов
Разработка задачи:
Иван Сафонов

Постановка задачи

- Загадан массив $c_1, c_2, ..., c_n$
- За одну операцию можно спросить 3 различных позиции в массиве i, j, h
- В ответ говорят число $\min(c_i, c_j, c_h) + \max(c_i, c_j, c_h)$
- За не более, чем 4n запросов, угадать массив

Решение задачи

- Пусть нам даны 4 различные позиции массива. Как за 4 запроса узнать сумму чисел массива на этих позициях?
- Спросим все 4 тройки из этих позиций. Тогда сумма четырех чисел это сумма минимального и максимального полученных в ответ чисел
- Почему это так? Пусть на этих позициях в каком-то порядке стояли числа $a_1 \leqslant a_2 \leqslant a_3 \leqslant a_4$. Тогда минимальное полученное в ответ число будет $a_1 + a_3$, а максимальное $a_2 + a_4$. Их сумма действительно равна $a_1 + a_2 + a_3 + a_4$

Решение задачи

- Осталось угадать массив, используя n раз операцию суммы четырех чисел.
- Посмотрим на первые 5 чисел. Спросим все 5 четверок этих чисел. Сложим полученные ответы. Если поделить полученную сумму на 4, то мы получим сумму первых 5 чисел. Если вычесть из суммы первых 5 чисел суммы во всех пяти четверках, то получатся первые 5 чисел массива
- Чтобы узнать число на позиции $i \geqslant 6$, узнаем сумму чисел на позициях $1, 2, 3, i$ и вычтем из нее $c_1 + c_2 + c_3$

Решение задачи

- Такое решение будет n раз считать сумму четырех чисел, поэтому будет использовать ровно $4n$ вопросов
- Также существует много других решений этой задачи
- Самые оптимальные из известных жюри решений используют
  - $2n$ и
  - $n + 3 + \log_2(m)$ запросов (где $m = 10^9$ — максимально возможное загаданное число)

Задача G
Комбокамень

Автор задачи:
Илья Збань
Разработка задачи:
Илья Збань

Постановка задачи

- Даны несколько операций вида
  - удвоить число
  - скопировать число
  - вычесть одно число из другого
- Надо промоделировать эти операции
- Много запросов

Проблемы

- В задаче нужно просто промоделировать операции из условия
- Основная проблема — точности стандартных типов данных вроде `long long` или `double` не хватает для решения задачи. Стандартные реализации длинной арифметики также не должны укладываться в ограничения по времени и памяти

Решение

- Значение атаки всегда степень двойки
- Можно хранить двоичную запись значения здоровья в персистентном дереве отрезков с нулями и единицами, каждую операцию на нем довольно просто выразить
- Удвоение здоровья — добавление нуля в конец дерева отрезков, копирование — копирование указателя на корень дерева отрезков

Решение

- Вычитание — найти первую единицу в записи, большую данной, поменять ее на ноль, а соответствующему подотрезку присвоить единицы
- Суммарное время работы — O(nlog(n))

Задача H
Линеаризация

Автор задачи:
Иван Сафонов
Разработка задачи:
Дмитрий Якутов

Постановка задачи

- Дано определение линейной строки из 0 и 1
- За одну операцию можно изменить элементы любой подстроки на противоположные
- Требуется за наименьшее число операций сделать строку линейной
- Много запросов

Решение задачи

- Пусть A — ответ на запрос для строки S длины N
- Пусть B — ответ на запрос для той же строки, но с другими операциями изменения:
  - Можно изменять элементы только на префиксе
- Тогда `A = [(B + 1) / 2]`
  - Пару префиксов можно заменить на отрезок
  - Отрезок можно заменить на пару префиксов
- Будем решать задачу для операций над префиксами

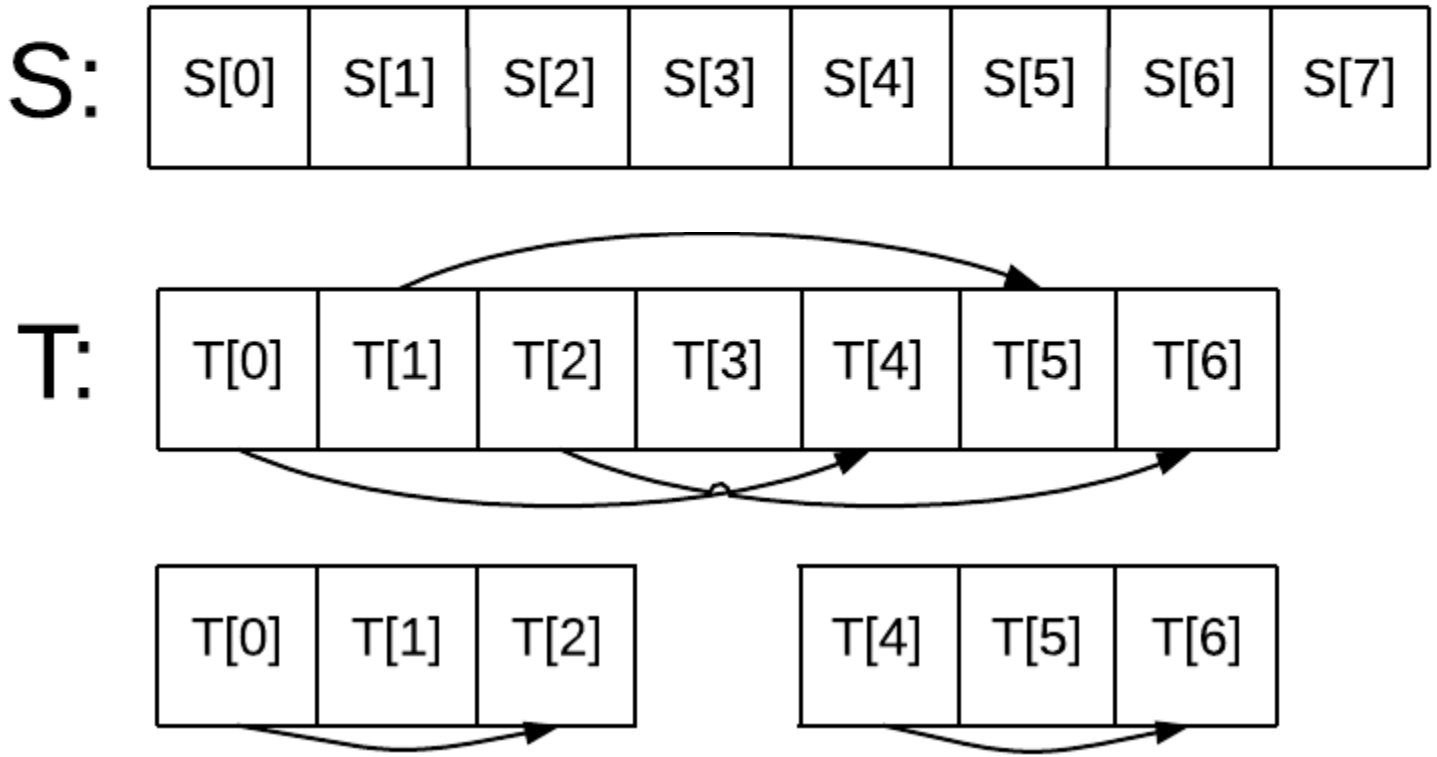
Решение задачи

- Как понять, что строка линейная?
  - Строка длины 1 ⇒ линейная
  - Левая половина линейная, правая половина линейная, половины равны или противоположны ⇒ линейная
- Заменим строку S длины N на строку T длины `N - 1`:
  - `T[i] = S[i] xor S[i + 1]`
  - Половинки T должны быть равны
  - Изменение префикса S ⇒ изменение одного элемента T

Решение задачи

- Решим задачу для одного запроса
- Построим граф
  - Вершины — элементы строки T
  - Ребра между элементами, которые должны быть равны
  - Для строки T рекурсивно строим графы для половинок
  - Проводим ребро между соответствующими элементами половинок — они должны быть равны

Решение задачи

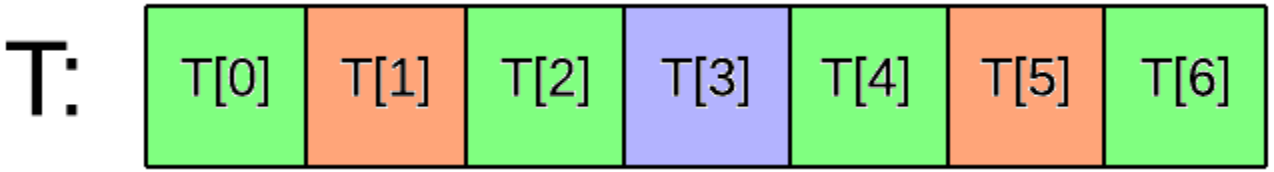


Решение задачи

- Граф распадается на компоненты связности
- В одной компоненте все элементы должны быть равны
- В одной компоненте X элементов строки T равны `0` и Y элементов равны `1`
- Прибавляем к ответу `min(X, Y)`
- Время работы: `nlog(n)` на один запрос
  - Слишком медленно

Решение задачи

- Как выглядят компоненты графа: Один цвет - одна компонента



- 1-я компонента — элементы с остатком 0 при делении на 2
- 2-я компонента — с остатком 1 при делении на 4
- ...
- K компонента — с остатком $2^{K-1} - 1$ при делении на 2^K

Решение задачи

- `t` — исходная строка, `u` — строка, содержащая значения функции `xor` для соседних элементов `t`, `m` — длина `u`
- Для `k = 0...log(m)` и `x = 0...2^k - 1` посчитаем количество единиц на всех префиксах строки `u(2^k, x)`
  - `u(2^k, x) = u[x], u[x + 2^K], ..., u[x + c · 2^K], ...`
  - Время работы `m log(m)`
- Для подстроки-запроса `(L, R)` и k-й компоненты найдем:
  - `C(k)` — количество элементов в k-й компоненте
  - `O(k)` — количество единиц в k-й компоненте
  - Прибавим к ответу `min(O(k), C(k) - O(k))`
  - Время работы: `log(m)` на запрос

Задача I
Минимальное произведение

Автор задачи:
Елена Андреева
Разработка задачи:
Михаил Путилин

Постановка задачи

- Дана последовательность чисел $a_1, \ldots, a_n$
- Найти такие i и j, что:
  - $i < j$
  - $a_i < a_j$
  - $a_i \cdot a_j$ минимально возможное
- Необходимо решение за $O(n)$
- Массив большой, поэтому входные данные нужно сгенерировать

Решение: случай 1

- Пусть $a_j \geqslant 0$
- Зафиксируем j
- Нужно выбрать такое $i < j$, что a_i минимально.
- Перебираем j от 1 до n, поддерживаем минимальное a_i

Решение: случай 2

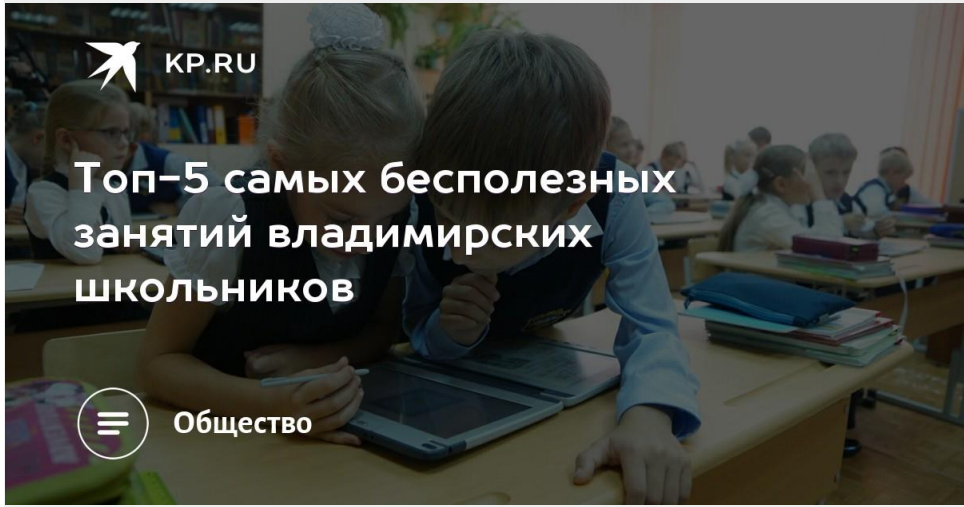
- Пусть $a_i \leqslant 0$. Это симметричный случай
- Зафиксируем i
- Нужно выбрать такое $j > i$, что a_j максимально
- Перебираем i от n до 1, поддерживаем максимальное a_j

Решение

- Кроме этих двух случаев ничего разбирать не надо
- $a_i < a_j$, поэтому не может быть так, что $a_i > 0$ и $a_j < 0$

Задача J
Два префикса

Автор задачи:
Михаил Анопренко
Разработка задачи:
Михаил Анопренко



Постановка задачи

- Даны две строки $s_1s_2...s_n$ и $t_1t_2...t_m$
- Найти количество различных строк вида $s_1s_2...s_at_1t_2...t_b$

Решение

- Всего есть nm комбинаций непустых префиксов строк
- Из различных комбинаций могут получаться разные строки
- Хотим посчитать количество комбинаций, для которых результат совпадает с какой-то другой комбинацией

Решение

- Пусть $s_1...s_at_1...t_b = s_1...s_ct_1...t_d$, причем $a < c$
- Очевидно, что $b > d$
- $t_1...t_b = s_{a+1}..s_ct_1...t_d$
- Таким образом, хотим найти количество префиксов t, у которого префикс длины d совпадает с суффиксом длины d, а префикс длины $b - d$ является подстрокой s

Решение

- Посчитаем префикс-функцию для строки t. Пусть в позиции i она равна p_i
- Для каждой позиции i в строке s найдем максимальную длину l_i подстроки, начинающейся в этой позиции, которая является подстрокой t. Посчитаем для этого z-функцию для строки ts
- Перебираем позицию i в строке s. Для неё посчитаем количество комбинаций, для которых существует комбинация с таким же результатом и большей длиной префикса s

Решение

- В качестве искомой подстроки `s` подходят все подстроки длины не более l_{i+1}
- Хотим для каждого префикса строки `t` понять, подходит ли он
- Это верно для позиции `j` тогда и только тогда, когда $j + 1 - p_j \leqslant l_{i+1}$, причем $p_j \neq 0$
- Почитаем для всех значений `x` от `1` до `m` количество индексов `j`, для которых $p_j \neq 0$ и $j + 1 - p_j \leqslant x$. Это можно сделать одним проходом и префиксными суммами. Назовем эти значения `pref[x]`

Решение

- Для всех позиций `i` в `s` посчитаем значение `m - pref[`l_{i+1}`]`
- `Сумма этих значений и будет ответом`

Задача K
Расширение Сознания

Автор задачи:
Михаил Дворкин
Разработка задачи:
Дмитрий Саютин

Постановка задачи

- Дано n бесконечных строки вида “`s + t + t + ...`“
- Требуется разбить их на минимальное число групп таких, что в каждой группе все строки являются подпоследовательностями друг друга

Эквивалентности

Заметим, что отношение «интересности» двух людей друг другу обладает рядом свойств:

- *Рефлексивность:* любой человек интересен себе
- *Симметричность*: если a и b взаимно интересны, то также взаимно интересны друг другу b и a
- *Транзитивность:* если a и b интересны друг другу, а также интересны друг другу b и c, то a и c тоже интересны друг другу

Эквивалентности

- Такие отношения называются “отношениями эквивалентности”, другой пример таких отношений: связность в неориентированном графе
- Для любого отношения эквивалентности верно, что все элементы можно разбить на множества так, что два элемента из одной группы всегда эквивалентны, а из разных всегда не эквивалентны. Такие множества называются классами эквивалентности.
- Например в случае отношения связности в графе такие классы называются компонентами связности

Решение

- В задаче требовалось разбить все строки на классы эквивалентности
- Для того, чтобы строки были эквивалентны, необходимо, чтобы множества букв в периодичной части совпали, иначе какая-то из букв встречается в одной строке конечное число раз, а в другой нет.
- Но этого недостаточно:

abcmmmmmm

defmmmmmm

Решение

- Давайте преобразуем каждую строку “`s + t + t + ...`” следующим образом: пока последний символ в `s` встречается в `t`, его можно удалить.

abcomnom nom nom nom nom nom…

→ abc nom nom nom nom nom…

- Очевидно, что такая операция не меняет отношения “одна строка — подпоследовательность другой” в силу бесконечности периода

Решение

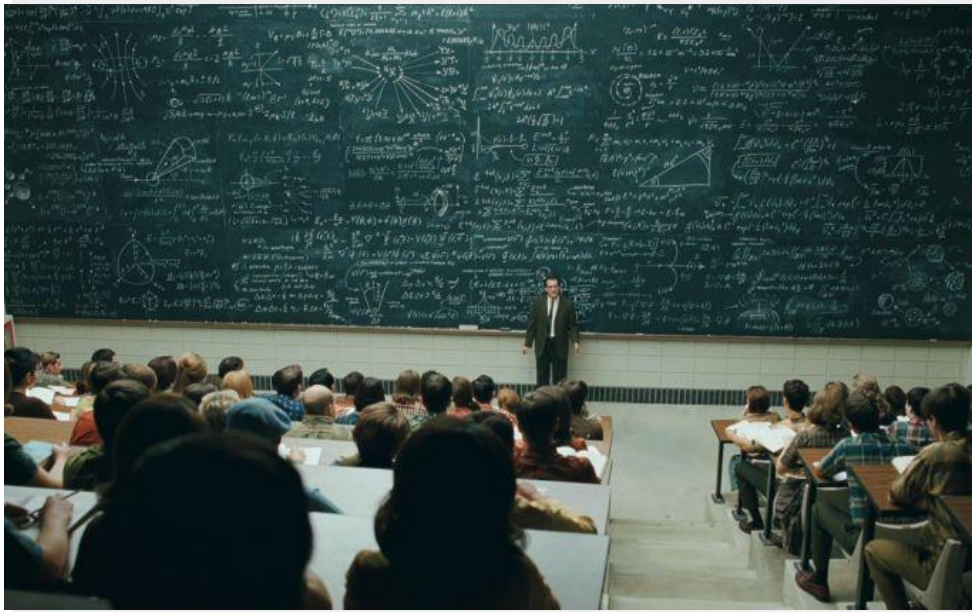
- На самом деле, это является и достаточным условием. То есть чтобы решить задачу достаточно сгруппировать все строки по принципу `pair<string, int>`, где `string` будет отвечать за укороченный предпериод, а `int` будет маской символов, встречающихся в периодической части
- Это можно сделать реализовать как `map<pair<string, int>, vector<int>>`, где последнее — список нужных индексов

Доказательство

- Как доказать достаточность? Пусть “x + y + y + ...” является подпоследовательностью “z + w + w + ...”, при этом x и z укорочены как было описано, а множества символов y и w совпадают
- Тогда x является подпоследовательностью “z + w + w + ...”. В силу того, что x не заканчивается на символ, который есть в w, получаем, что x — подпоследовательность z
- Применив то же рассуждение с другой стороны, получим, что z — подпоследовательность x, а значит z = x

Задача L
Университет Берляндии

Автор задачи:
Ольга Кунявская,
Павел Кунявский
Разработка задачи:
Роман Коробков



Постановка задачи

- Даны две аудитории, одна вмещает A человек, другая — B человек.
- n лекций, аудитории чередуются.
- Студент получает зачет, если посещает хотя бы k лекций.
- Какое максимальное число студентов сможет получить зачет?

Решение задачи

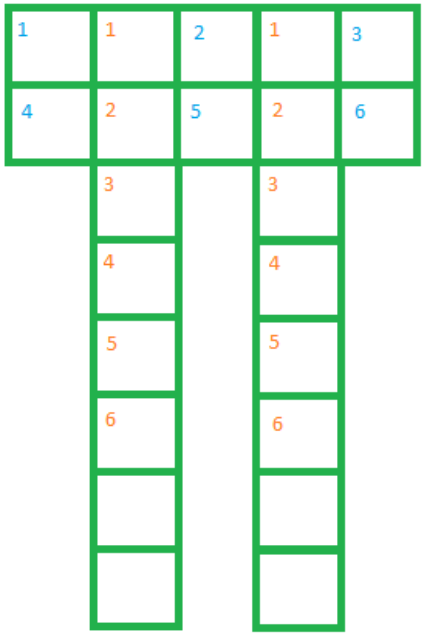
- Если `k > n`, то ни один человек не сможет сдать
- В остальных случаях ответ почти всегда равен

```
(число свободных мест) / k
```

- Ответ не равен в случае, если `k > n / 2`, и в большой аудитории мест сильно больше, чем в маленькой
- Решения, не учитывающие это, получали WA на тесте 11

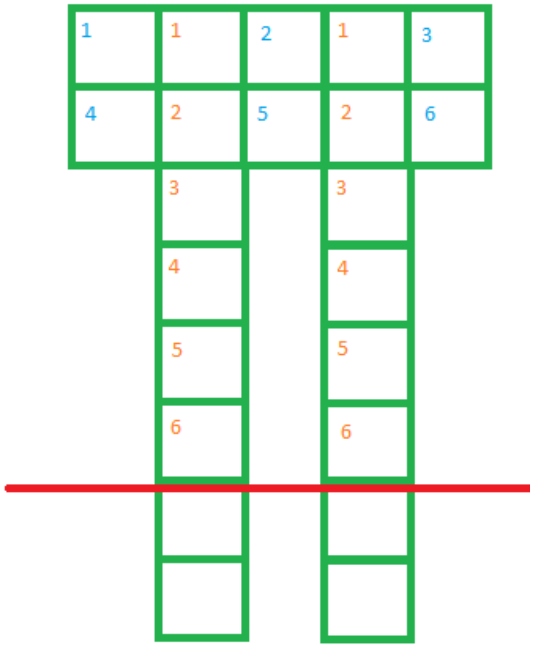
Решение задачи

- Например, `n = 5, k = 3, a = 2, b = 8`
- Нельзя занять все возможные места, потому студенту необходимо выбрать места и в большой, и в маленькой аудиториях



Решение задачи

- Ограничим высоту большой аудитории:
- `b = min(b, количество свободных мест в маленькой аудитории / (k - число лекций в большой аудитории))`



Решение задачи

- После ограничения высоты каждый новый студент может выбирать себе k лекций с максимальным числом свободных мест
- Таким алгоритмом (количество свободных мест / k) студентов смогут сдать зачет

Задача M
Приятная прогулка

Автор задачи:
Андрей Станкевич
Разработка задачи:
Андрей Станкевич

Постановка задачи

- Дана последовательность цветов домов
- Найти самый длинный подотрезок домов, где любые два соседних имеют разный цвет

Решение задачи

- Рассмотрим пары соседних домов одинакового цвета
- Ответ не должен включать никакую такую пару
- Найдем все такие пары, возьмем максимальное расстояние между соседними парами

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

Материалы олимпиады

http://neerc.ifmo.ru/school