XI Открытый кубок школьников

Санкт-Петербургский Государственный Университет

воскресенье, 29 апреля 2018 года

Задача «Разбиение Массива»

- Идея задачи — Евгений Тушканов
- Разработка задачи — Павел Глушень

Постановка задачи

- Требовалось проверить существует ли такое число k, что можно разбить массив ровно на k кусков, в каждом из которых ровно k чисел со значением k.

Основная идея

- Заметим, что достаточно проверить правда ли что существует такое положительное k, что оно встречается в массиве ровно k^2 раз.
- Чтобы найти в массиве такой элемент, нужно, пройдя по массиву, посчитать для каждого положительного элемента количество его вхождений.

Решение

- Хранить информацию можно, например, в словаре: ключом словаря будет элемент, значением — количество вхождений.
- После этого нужно пройти по словарю, проверив есть ли ключ, которому соответствует значение, равное квадрату ключа.
- Время работы: $\mathcal{O}(n \log(n))$ или $\mathcal{O}(n)$, если воспользоваться хеш-таблицей.

Задача «Китайские конфеты»

- Идея задачи — Андрей Райский
- Подготовка тестов — Андрей Райский

Постановка задачи

- Массив из чисел
- Не более одного обмена
- Максимизировать длину отрезка с различными числами

Решение

- Как понять, что отрезок хороший?
  - Все числа различны
  - Все числа различны, кроме одного, которое встречается дважды и его можно свапнуть с чем-нибудь за границей отрезка
- Можно свапать, если число различных на отрезке меньше, чем в массиве
- Два указателя
- Бинпоиск (опционально)

Задача «Удвоение прямоугольников»

- Идея задачи — Андрей Райский
- Подготовка тестов — Андрей Райский

Постановка задачи

- Игра на полоске из w клеточек
- На каждом ходу игрок удваивает свой прямоугольник
- Цель — пересечь чужой прямоугольник

Решение

- Состояние — четверка (l, r, $step$, $turn$)
- Два перехода: удвоиться влево и вправо
- Все переходы выигрышные $\rightarrow$ проигрыш
- Иначе выигрыш
- $O(w^2 \log w)$ состояний?

Решение

- Длина цепочки не более $2 \cdot \log w$
- И два перехода
- $O(w^2)$!
- Все еще долго :(

Решение

- Пошаффлим переходы
- Будем отсекаться при нахождении проигрышного перехода (гарантирует выигрыш)
- $O(w^{1.45})$

Задача «Галактическая служба связи»

- Идея задачи — Иван Казменко
- Подготовка тестов — Иван Казменко

Постановка задачи

- Сообщения из 30 битов передаются по каналу из 50 битов
- Каждый ноль, имеющий единицу слева, может независимо от других превратиться в единицу
- Необходимо уметь кодировать сообщения так, чтобы с учетом изменений раскодирование было точным

Решение 1 (43 бита)

- Кодируем строкой Фибоначчи
- Необходимо уметь искать строку Фибоначчи по номеру и номер по строке
- Найдем строку Фибоначчи длины 50 с номером, равным данному сообщению (в двоичной записи)
- Кодирование: если видим единицу, на следующей позиции окажется ноль
- Раскодирование: если видим единицу, просто пропускаем следующую цифру

Решение 2 (47 битов)

- Посмотрим, сколько среди заданных 30 битов единиц
- Если 15 или меньше, кодируем так: 0 превращается в 0, а 1 в 10
- Если 16 или больше, кодируем наоборот: 1 превращается в 0, а 0 в 10
- В обоих случаях в конце дописываем что угодно: сам код занимает от 30 до 45 битов
- Чтобы различить эти два случая, первым битом закодированного сообщения запишем номер случая
- И ещё один ноль после него на случай, если этот бит окажется единицей.

Решение 2 (47 битов)

- Раскодирование: вначале по первым двум битам узнаём, надо ли инвертировать ответ
- Далее, пока не наберётся 30 битов, 0 превращается в 0, а 1 превращается в 1 и пропускает следующую цифру кода. После этого инвертируем ответ, если требуется

Решение 3: разбиение на кусочки (50 битов)

$000 \to 00000$	$00000 \to 000$
$001 \to 00010$	$0001? \to 001$
$010 \to 00100$	$001?0 \to 010$
$011 \to 01000$	$01?00 \to 011$
$100 \to 01010$	$01?1? \to 100$
$101 \to 10000$	$1?000 \to 101$
$110 \to 10010$	$1?01? \to 110$
$111 \to 10100$	$1?1?0 \to 111$

Задача «Актуальная задача»

- Идея задачи — Олег Евсеев
- Разработка задачи — Александр Логунов

Постановка задачи

- Последовательно приходят запросы двух видов:
- Заблокировать все IP-адреса с заданным префиксом.
- Проверить заблокирован ли заданный IP-адрес.

Решение

- Когда заданный IP-адрес заблокирован?
- Тогда и только тогда, когда какой-то префикс этого IP-адреса был ранее заблокирован.
- Можно перебрать все префиксы и проверить были ли они добавлены в блокировку ранее, их всего 32.

Решение

- Заметим, что IP-адрес — это битовая строка длины 32.
- Значит IP-адрес, а также все фильтры можно хранить в 32-битных беззнаковых числах.
- Заведём 32 `set`-а для фильтров разной длины и будем использовать их для проверки.
- Выделить префикс IP-адреса можно с помощью битового сдвига за $\mathcal{O}(1)$.

Решение

- Итоговая асимптотика: $\mathcal{O}(32n \log n)$ или $\mathcal{O}(32n)$ при использовании хеш-таблицы.

Решение

- Итоговая асимптотика: $\mathcal{O}(32n \log n)$ или $\mathcal{O}(32n)$ при использовании хеш-таблицы.
- Можно также решить задачу с помощью бора: добавляем в бор фильтры как строки, затем при спуске проверяем что нет терминальных состояний.

Задача «K-pop»

- Идея задачи — Александр Марков
- Разработка задачи — Александр Марков

Постановка задачи

- Дан неориентированный граф из музыкальных групп.
- Каждую группу можно пригласить за некоторую сумму денег
- Дополнительно можно собирать много людей на концерт: если набралось хотя бы x_i людей, то для соответствующего ребра приезд одного конца влёчёт автоматический приезд другого.
- В наличии S денег, сколько нужно собрать людей чтобы пригласить всех груп?

Основная идея

- Предположим мы собрали x людей на концерт.
- Рассмотрим в графе компоненты связности по рёбрам веса $\leqslant x$ и компоненты связности ими порождённые.
- В каждой компоненте, очевидно, нужно пригласить самую «дешёвую» группу.
- То есть необходимое количество денег — это сумма по всем компонентам связности минимальной «цены» группы в компоненте.

Основная идея

- Заметим, что достаточно проверять только те x, которые написаны хотя бы на каком-нибудь ребре.
- Отсортируем все рёбра по возрастанию веса, на них написанных, и будем последовательно их добавлять в таком порядке.

Решение

- Изначально у нас есть n компонент, в каждой из которых ровно одна вершина.
- Затем при добавлении ребра компоненты склеиваются.
- Будем поддерживать компоненты в системе непересекающихся множеств.
- Будем внутри каждой компоненты в СНМ можно также хранить минимум в ней.
- Дополнительно будем хранить всеобщую сумму минимов.

Задача «Шифрованный калькулятор»

- Идея задачи — Александр Логунов
- Разработка задачи — Александр Логунов.

Постановка задачи

- Есть калькулятор, который хранит в памяти некоторое число, а на дисплее показывает только сумму его цифр.
- Можно изменить число в памяти с помощью какой-то арифметической операции, соответственно в результате этого изменится и значение на дисплее.
- Проделать некоторый набор операций и узнать какое число было в памяти изначально.


0
ON AC
OFF
000
CE/C
M+
M-
MR C
+/-
7
8
9
:
%
4
5
6
X
−
1
2
3
+
=
0
00
.

Идея решения

- Будем восстанавливать число от младших цифр к старшим.

Решение 1, с помощью вычитания единицы

- Будем последовательно вычитать из числа единицу.
- Рано или поздно мы получим либо нулевую сумму цифр, либо сумма цифр уменьшится больше чем на 1 за один шаг.
- По количеству шагов мы узнаём последнюю цифру, правда возможно портим остальные (они могут уменьшится из-за переноса единицы).

Решение 1-2

- Давайте сделаем += c, чтобы отменить все проделанные операции (c — количество проделанных шагов)
- А затем поделим число нацело на 10 и продолжим решать меньшую задачу.

Решение 1-2

- Давайте сделаем += c, чтобы отменить все проделанные операции (c — количество проделанных шагов)
- А затем поделим число нацело на 10 и продолжим решать меньшую задачу.
- Существует аналоничное решение, которое вместо вычитания единицы её добавляет.
- В обоих случаях количество операций не больше $12 \cdot 19$ действий, что меньше 300.

Решение 3: деление на 10

- Пока сумма цифр не ноль, можно просто делить число нацело на 10.
- Тем самым сумма цифр уменьшается ровно на последнюю цифру.
- Количество действий не больше 19.

Задача «Captcha»

- Идея задачи — Михаил Иванов
- Подготовка тестов — Михаил Иванов

Постановка задачи

- Дан набор цифр
- Составить из него число с минимальным количеством делителей

Решение

- Перебор до некоторой верхней границы
- Предподсчет делителей
- Проверить, что один сет внутри другого
- $M \log M$, где M — верхняя граница

Решение

- Разбор случаев!
- 1, 2, 3, 5, 7 — понятно
- ...
- И еще 15 случаев
- Итоговый лист:
  $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 89, 409, 449, 499, 6469\}$
- Лучше написать перебор :)

Задача «Эстафета»

- Идея задачи — Михаил Иванов
- Подготовка тестов — Михаил Иванов

Постановка задачи

- Даны n, m
- Найти перестановку p из n элементов с порядком m

Основные идеи

- Порядок перестановки — НОК длин ее циклов
- Сумма длин должна быть не больше n
- Минимизируем сумму при данном произведении
- Длина каждого цикла — p^x, где p — простое

Решение

- $m = \prod p_i^{x_i}$
- Раскладываем на простые множители, суммируем
- Строим циклы последовательно, добиваем единичками

Решение

- Факторизуем за $\sqrt{m}$?
- Долго :(
- Простые $> n$ дают сумму больше n
- Перебираем простые до $\min(\sqrt{m}, n)$

Задача «Покер»

- Идея задачи — Александр Логунов
- Разработка задачи — Дмитрий Саютин

Постановка задачи

- Дан массив карт и запросы к его отрезкам.
- Для каждого отрезка нужно вывести размер наибольшего стрита на нём.

Постановка задачи

- Дан массив карт и запросы к его отрезкам.
- Для каждого отрезка нужно вывести размер наибольшего стрита на нём.
- То есть такое наибольшее число k, что $k = b - a + 1$ для некоторых a и b
- И все числа от a до b встречаются на данном отрезке.


FAST RIVER
POKER CHIPS
SNAKE
100
1000
A
A

Идея

- Чтобы ответить на какой-то запрос, достаточно хранить множество рангов карт на этом отрезке в какой-то структуре данных.
- Давайте предложим такую структуру данных, предназначенную для хранения непересекающихся отрезков.

Структура

- Структура состоит из массива go, который для каждой правой границы хранит парную ей левую, а для каждой левой границы — парную правую.
- А также из дополнительнего булевского массив, в котором хранится, покрыта точка отрезком или нет.
- Такая структура позволяет производить операцию добавления точки.

Добавление точки

- Чтобы добавить точку нужно сначала нужно посмотреть в булевский массив, чтобы проверить, не включена ли точка уже.
- Затем завести новый тривиальный отрезок (`go[pos] := pos`)
- и, возможно, склеить его с соседними отрезками слева и справа (посмотрев в `go[pos - 1]` и `go[pos + 1]`).

Решение за длину запроса

- Например, с помощью такой структуры данных мы уже научились отвечать на запросы за их длину (в предположении того, что структура уже создана).
- Для этого нужно просто последовательно добавить в структуру все точки на соответствующем отрезке, попутно поддерживая текущую максимальную длину отрезка.
- Поскольку она только возрастает, это можно делать с помощью одной целочисленной переменной.

Алгоритм МО

- Дальнейшее развитие решения основано на алгоритме Мо.
- Несмотря на то, что в такую структуру очень легко добавлять новые точки (за $\mathcal{O}(1)$), удалить произвольную старую точку непросто.
- С другой стороны, эта структура данных является *откатываемой*: если вести лог всех присваиваний (и предыдущих значений), то достаточно легко откатиться назад до нужной версии.
- Воспользуемся этим фактом.

Решение

- Заведём глобальную структуру данных и будем периодически её откатывать.
- Отдельно решим все короткие запросы (длины не более $\sqrt{n}$).
- Все остальные запросы сгруппируем по левой границе в $\sqrt{n}$ групп: запрос с началом в l положим в группу с номером $\lfloor l/\sqrt{n} \rfloor$.

Решение

- Так как длина запросов хотя бы $\sqrt{n}$, то каждый запрос пересекает границу со следующим блоком массива.
- Будем обрабатывать запросы в порядке увеличения правой границы.

Решение

- Сначала для запроса добавим весь ещё не добавленный кусок от пересечения со следующим блоком до конца правой границы. Так как правая граница монотонна, то суммарно таких добавлений внутри группы не больше n.
- Затем запомним текущее состояние, и добавим все элементы от левой границы до пересечения. Получим ответ и сохраним его. Затем откатим все добавленные *левые* элементы и перейдём к следующему запросу.

Решение

- Так как мы сгруппировали все запросы по левой группе, то на каждый запрос растить левую границу не придётся больше чем $\sqrt{n}$ раз
- А откаты занимают не больше времени, чем исходные операции.
- Итоговая асимптотика: $\mathcal{O}(n\sqrt{n})$, в предположении, что $n \approx q$.

Задача «Фотоувеличение»

- Идея задачи — Артур Рязанов
- Разработка задачи — Артур Рязанов

Постановка задачи

- Формально, задача это классическая задача выполнимости (SAT).
- На фотографиях люди в костюмах — переменные, чёрные и красные костюмы соответствуют `true` и `false` соответственно.
- Нужно найти выполняющий набор.

Существование

- Для начала докажем, что решение существует.
- Рассмотрим математическое ожидание количества клозов при случайном выборе любого набора значений.

Существование

- Для начала докажем, что решение существует.
- Рассмотрим математическое ожидание количества клозов при случайном выборе любого набора значений.
- Оно $\geqslant m \cdot \left(1 - \frac{1}{2^k}\right) > m - 1$.
- Так как эта случайная величина принимает только целые значения, существует набор значений, выполняющий все клозы.

Решение

- Заметим, что можно удалить из каждого клоза все литералы, кроме каких-нибудь k, и формула всё еще будет выполнима.
- Будем действовать жадно: перебирать переменные в любом порядке и присваивать каждой «более удачное значение».
- Чтобы выбрать, какое значение более удачное, будем поддерживать веса клозов $w_1, \ldots, w_m$.
- Изначально (когда ни одно значение не подставлено) все $w_i = 1$.

Инвариант

- Далее будем поддерживать следующий инвариант: $w_i = 0$, если клоз выполнен и $w_i = 2^{k-\ell}$ если в клозе осталось ℓ литералов.
- Будем также поддерживать что $\sum_{i=1}^{m} w_i \leqslant m$ в любой момент времени. Заметим, что если нам удастся так подставлять значения, то итоговый набор будет выполнять все клозы.
- Действительно, если бы какой-то клоз обнулялся нашей подстановкой, то, поскольку в клозе осталось 0 литералов,

Решение

- Как выбирать значение очередной переменной x_i?
- Пусть $P \subseteq \{1, 2, \ldots, m\}$ "— множество клозов, содержащих переменную x_i с положительным знаком, а N "— с отрицательным.
- Тогда, если $\sum_{j \in P} w_j \geqslant \sum_{j \in N} w_j$, подставим $x_i = 1$, а иначе $x_i = 0$.
- Рассмотрим первый случай. Когда мы подставим $x_i = 1$, веса w_j для $j \in N$ удвоятся, а для $j \in P$ обнулятся.
- Тогда суммарный вес не уменьшится и,

Задача «Xor шаров»

- Идея задачи — Вадим Володин
- Подготовка тестов — Вадим Володин

Постановка задачи

- Дан некоторый массив и число X
- Каждый элемент можно не более одного раза изменить на $+1$ или -1
- За минимальное число операций сделать ксор элементов массива равным X

Решение

- Пусть ксор изначального массива равен Q
- Необходимо превратить Q в X за минимальное число действий
- Добавление или вычитание единицы инвертирует некоторый суффикс числа
- Инвертирование суффикса числа влечет инвертирование суффикса результирующего ксора
- Иными словами, необходимо превратить Q^X в 0

Решение

- Заметим, что нет смысла более одного раза инвертировать суффикс одной и той же длины
- Инвертируем суффикс с длиной, равной номеру старшего бита Q^X
- Старший бит становится равным нулю, а в младших битах что-то меняется
- Опять инвертируем старший бит среди оставшихся...
- Получаем, что набор длин суффиксов фиксирован!
- Осталось лишь проверить, можно ли такой набор набрать :)

Решение

- Каждое число в массиве порождает две длины суффикса
- Составим двудольный граф, проведем нужные ребра, найдем совершенное паросочетание
- Достаточно для OK, но можно проще!
- Если число делится на 2, то $+1$ порождает суффикс длины 1
- Если число не делится на 2, то -1 порождает суффикс длины 1
- Наберем суффиксы жадно от большего к меньшему
- Профит!

.end()