Индивидуальная олимпиада по информатике и программированию. Заключительный этап.

19 марта 2017 года

Задача A
Космический корабль

Автор задачи:
Дмитрий Филиппов
Разработка задачи:
Виктория Ерохина

Постановка задачи

- *n* врагов, один из которых босс
- Сила босса равна сумме сил остальных врагов
- Нужно вывести силы в таком порядке, чтобы босс был последним

Решение (20 баллов)

- Переберем врага, который может быть боссом
- Рассмотрим сумму оставшихся врагов, которую будем считать каждый раз заново
- Решение будет работать за $O(n^2)$

Решение (30 баллов)

- В данной подгруппе нет отрицательных чисел
- Босс — максимальный по силе среди всех врагов
- Найти максимум в массиве можно за O(n)

Решение (100 баллов)

- Заметим, что если сила босса равна x, то сумма сил всех врагов равна 2x
- Следовательно, чтобы найти силу босса, необходимо посчитать суммарную силу всех врагов и разделить ее на 2
- Решение будет работать за $O(n)$

Комментарий

- Чтобы восстановить порядок достаточно найти позицию босса и, например, поменять его местами с врагом на последней позиции.

Задача B
Рейнджеры в автобусе

Автор задачи:
Илья Пересадин
Разработка задачи:
Михаил Путилин

Постановка задачи

- Есть пассажиры в автобусе
- Для каждого известно место, куда он садился
- Известны предпочтения рейнджеров
- Нужно для каждого пассажира узнать, каким он мог быть рейнджером

Решение (25 баллов)

- Будем обрабатывать пассажиров по очереди и для каждого определять, кем он мог бы быть. Сразу заметим, что розовый рейнджер может есть куда угодно, поэтому каждый пассажир мог бы быть розовым.
- Для каждого места будем помнить, свободно ли оно
- Используем для этого массив типа *boolean* длины *2n*

Решение (25 баллов)

- Проверим, мог бы очередной пассажир быть красным рейнджером
- Для этого найдём место, на которое сел бы красный
- Переберём циклом ряд от 1 до n, если попался ряд, в котором есть свободное место, выбираем левое, если оно свободно, или правое, если нет — это и есть место, куда сел бы красный.

Решение (25 баллов)

- Поскольку он выбирает место однозначно, мы можем определить, мог бы этот пассажир быть красным — нужно проверить, что он сел именно на это место.
- Таким же образом узнаем, куда сели бы синий, жёлтый и чёрный – разница будет в порядке перебора рядов (от *1* до *n* или наоборот) и в порядке выбора места в ряду (сначала левое или правое).
- После обработки пассажира отметим его место как занятое.

Решение (25 баллов)

- Решение работает за $O(nk)$ и использует $O(n)$ памяти

Решение (50 баллов)

- На самом деле предыдущее решение работает за $O(k^2)$ времени
- Каждый цикл по рядам останавливается, когда находит ряд со свободным местом — а количество занятых рядов не более $k/2$
- Однако оно использует $O(n)$ памяти, а это слишком много для второй подзадачи

Решение (50 баллов)

Есть два способа решения этой проблемы:

- Вместо массива использовать *std::set*, в котором хранить занятые места
- Хранить только $k/2 + 1$ первых рядов и столько же последних.

Тогда решение будет использовать $O(k)$ памяти.

Решение (100 баллов)

- Каждый из циклов останавливается, когда находит ряд, в котором есть свободное место
- Нужно быстро находить первый и последний незанятые ряды
- Будем поддерживать эти значения — *first* и *last*

Решение (100 баллов)

- Изначально *first* = *1*, *last* = *n*
- После каждой итерации цикла обновим эти значения. Будем увеличивать *first* на *1*, пока ряд *first* занят, затем уменьшать *last* на *1*, пока ряд *last* занят
- Занятых рядов не может оказаться больше *k*/2, поэтому *first* и *last* сделают *O(k)* шагов

Таким образом, решение работает за *O(k)*

Задача C
Мегазорды

Автор задачи:
Григорий Шовкопляс
Разработка задачи:
Станислав Наумов

Постановка задачи

- Есть зорды трех цветов
- Есть правила на создание мегазорда
- Нужно узнать количество способов сделать мегазорда

Решение (15 баллов)

- Переберем зорды трех различных цветов и проверим условия
- Для проверки достаточно сравнить первую цифру красной модели с последней цифрой зеленой, а также последнюю цифру красной модели с первой цифрой синей
- Решение будет работать за $O(n^3)$

Решение (25 баллов)

- В этой группе количество различных номеров моделей мало
- Для каждого цвета посчитаем количество моделей с фиксированным номером
- Достаточно перебрать номера моделей для всех трех рейнджеров и прибавить к ответу произведение их количества
- Решение будет работать за $O(90^3 + n)$

Решение (31 балл)

- Предподсчитаем массив *R[a][b]* — количество моделей у красного рейнджера, у которых первая цифра равна *a* и последняя равна *b*
- Переберем номер модели, который взяли зеленый и синий рейнджеры
- Зафиксируется первая и последняя цифра номера красной модели.
- Используя массив *R*, за *O(1)* узнаем количество подходящих моделей красного рейнджера.

Решение (46 баллов)

- В этой подгруппе, существуют модели с одинаковыми номерами
- Заметим, что решение для третьей группы посчитает способы, в которых могут повторяться номера моделей
- Для начала учтем в рассмотрении вариантов, что номера моделей зеленого и синего рейнджера не должны совпадать

Решение (46 баллов)

- Теперь для получения верного ответа вычтем случаи, в которых модели красного рейнджера совпадают либо с зеленым, либо с синим
- Посчитаем количество моделей, которые совпадают у красного и синего, а также красного и зеленого рейнджеров и вычтем эти числа из ответа, если они были посчитаны в нашем решении
- Это можно сделать при помощи структуры *map* или сортировки

Решение (100 баллов)

- Посчитаем массивы $G[a]$ и $B[b]$ — количество моделей у зеленого рейнджера, у которых последняя цифра номера равна a и количество моделей синего рейнджера, у которых первая цифра равна b
- Теперь переберем первую и последнюю цифру номера модели красного рейнджера.
- К ответу прибавим $G[a] \cdot R[a][b] \cdot B[b]$

Решение (100 баллов)

- Не забудем вычесть варианты, в которых у какой-то пары совпадают номера моделей
- Для этого воспользуемся идей из четвертой подгруппы
- Заметим, что вариант, где все три рейнджера имеют одинаковый номер модели будет вычитаться 3 раза
- Поэтому необходимо прибавить это количество увеличенное в 2 раза

Задача D
Объединенная армия

Авторы задачи:
Никита Михайлов
Григорий Шовкопляс
Разработка задачи:
Григорий Шовкопляс

Постановка задачи

- Армии задали два вопроса
- Есть солдаты, которые всегда лгут хотя бы на один вопрос, и которые всегда говорят правду
- Нужно узнать возможные расстановки с наименьшим и наибольшим возможным количеством говорящих правду солдат

Частичные решения (0-100 баллов)

- Случаев, когда в армии могут находится не только солдаты из Глиняного патруля всего 15. Некоторые из них довольно легко рассмотреть на листочке.
- Например, для $x = 0$ и $y = 2$ для $k > 1$ и наименьший, и наибольший ответ будут равны двум, а расстановка будет выглядеть примерно так: $000 \cdots 001$ в первом ряду и $100 \cdots 000$ во втором
- Более того, разбором случаев можно набрать до 100 баллов включительно.

Полное решение (100 баллов)

Будем решать задачу динамическим программированием по профилю.

- *dp[i][mask_prev][mask_cur]* — минимальное/максимальное количество солдат из Зедд патруля, которые могут быть в расстановке из первых *i* столбцов, *mask_cur* — маска *i*-го столбца, а *mask_prev* — *(i − 1)*-го.

Полное решение (100 баллов)

- Переберем маску для (i + 1)-го столбца и проверим выполняются ли ограничения на соседей, для солдат из i-го − столбца, так как теперь известны все их соседи.
- Для первого и последнего столбца ограничения нужно проверять отдельно, потому что у них нет одного из соседей.

Полное решение (100 баллов)

- База: *dp[2][mask_prev][mask_cur]* равно количеству единиц в *mask_prev* и *mask_cur*, если *mask_prev* может стоять слева от *mask_cur*
- Переход: *dp[i+ 1][mask_cur][mask_next]* = *dp[i][mask_prev][mask_cur]+ones(mask_next)*, где *ones(x)* — количество единиц в маске *x*

Полное решение (100 баллов)

- Ответ будет минимумом/максимумом среди всех *dp[k][mask_prev][mask_cur]*, таких что *mask_cur* может находится справа от *mask_prev*.
- Так как размерность маски константа и равна двум, данное решение будет работать за *O(n)*.

Задача E
Тюрьма для Зедда

Автор задачи:
Илья Збань
Разработка задачи:
Илья Збань

Постановка задачи

- Есть прямоугольные листы
- Нужно собрать из них параллелепипед максимального объема

Общие замечания

- Противоположные грани параллелепипеда должны быть равны
- Переведем каждый прямоугольник к виду $a_i \le b_i$, и искать уже три прямоугольника

Общие замечания

- Если длины сторон, исходящих из одной вершины, равны A, B и C, то параллелепипед должен быть составлен из трех прямоугольников со сторонами $A \times B$, $A \times C$, $B \times C$

Есть два аспекта:

- Для каждой грани должно быть хотя бы по два соответствующих прямоугольника
- Два или три размера могут быть равны друг другу

Решение (27 баллов)

- Переберем три прямоугольника, которые войдут в ответ, и явно проверить условие, описанное ранее
- Данное решение работает за $O(n^3)$

Решение (63 балла)

- Можно перебрать один прямоугольник
- Пусть он имеет размер $A \times B$ — у нас уже известно две размерности параллелепипеда,
- Переберем третью размерность параллелепипеда C и проверим, есть ли прямоугольники размера $A \times C$ и $B \times C$.

Решение (63 балла)

- На этом моменте можно допустить ошибку, когда $A = B$, $A = C$ или $B = C$, попытавшись использовать один прямоугольник дважды, хотя его можно взять не больше одного раза
- Для простоты можно считать что A, B и C различны, а случаи, когда какая-то пара совпадает, разобрать отдельно, это несложно сделать за $O(n)$
- Данное решение работает за $O(n^2)$

Решение (64-100 баллов)

- Перейдем к графовой интерпретации задачи.
- Наличие прямоугольников со сторонами $A \times B$, $A \times C$, $B \times C$ эквивалентно нахождению цикла длины 3 в графе, построенном на числах, отвечающих за размерности параллелепипеда — прямоугольник $A \times B$ отвечает в таком графе за ребро AB

Решение (64-100 баллов)

Решение за $O(n^2/64)$

- Перебираем одно из ребер *uv* треугольника, и смотрим на пересечение множества соседей вершин *u* и *v*
- Это можно сделать пересечением bitset-ов, и перебрать третью вершину треугольника как все единички в пересечении *bitset*-ов

Решение (64-100 баллов)

Решение за $O(n\sqrt{n})$

- Упорядочим все вершины по возрастанию их степени.
- Переберем вершину с минимальной степенью, входящую в треугольник
- Переберем ее соседа — вершину, которая будет второй по степени среди вершин нашего цикла
- Переберем третью вершину, как соседа второй, и проверим, есть ли ребро между первой и третьей вершиной

Почему это быстро?

- Если перебирать вершины именно в таком порядке увеличения степени на графе с m ребрами, будет рассмотрено не более $m\sqrt{m}$ троек чисел
- Для этого можно разбить отсортированный массив степеней *deg* на две части: префикс, в котором сумма не больше $\sqrt{n}$, и суффикс
- Когда первая вершина из первой части массива, мы переберем $\sqrt{n}$ кандидатов на вторую вершину и n кандидатов на третью, что дает $n\sqrt{n}$

Почему это быстро?

- Во второй же части не больше $\sqrt{n}$ вершин, поэтому кандидатов на третью вершину для них будет не больше $\sqrt{n}$, что снова приводит нас к тому, что треугольников в графе не больше $O(n\sqrt{n})$

print(’Спасибо за внимание’)