Заключительный этап ИОИП

27 марта 2016 года

Задача A. Смех, да и только!

Задача A. Смех, да и только!

- Идея задачи — Илья Пересадин
- Подготовка тестов — Григорий Шовкопляс
- Разбор задачи — Григорий Шовкопляс

- Есть строка из латинских букв
- Нужно найти длину наибольшей подстроки, названной смехом, состоящей из чередующихся букв «a» и «h»

- Для каждой начальной позиции посчитаем длину смеха.
- Просто идем по строке и проверяем, что содержаться только нужные буквы, а также, что они чередуются.
- Решение работает за n^2 операций — долго, но можно было получить 47 баллов.

- Заметим, что для каждой подстроки, мы найдем и ее подстроки, длина которых точно меньше.
- Например для строки «aha», будут рассмотрены «ha» и «a».
- Тогда можно вместо того, чтобы перебирать начало, пересчитывать его во время выполнения подсчета:
  - Посчитали длину смеха из стартовой позиции
  - Следующая позиция, с которой имеет смысл смотреть, это позиция после текущей.
  - Делаем эту позицию стартовой.
- Таким образом, мы побываем в каждом символе строки не более одного раза, а значит с легкостью получим заслуженные 100 баллов.

Задача B. Путь в никуда

Задача B. Путь в никуда

- Идея задачи — Григорий Шовкопляс
- Подготовка тестов — Григорий Шовкопляс
- Разбор задачи — Григорий Шовкопляс

- Есть клетчатая доска размером $n \times m$
- Из клетки $(x; y)$ начинается «спиралька», которая проходит по всем клеткам и заканчивается, когда выходит за край.
- Нужно посчитать через сколько клеток она прошла.

- Будем хранить доску, как двумерный массив.
- Промоделируем процесс, отметим клетки, в которых побывали.
- Пройдемся по массиву и посчитаем ответ.
- Решение работает за nm операций, и получает 25 баллов.

- Не будем хранить доску.
- Промоделируем процесс, к счетчику каждый раз прибавляем длину шага.
- i-й шаг пройдет по $\lfloor i/2 \rfloor$ клеткам.
- Пересчитываем координату, проверяем, что не вышли за пределы доски.
- Решение работает за n операций.

- Пусть на i-м шаге Колобок пройдет по c_i клеткам, а за пределы доски вышел на k-м шаге.
- Заметим, что все покрытые клетки образуют прямоугольник.
- При этом его площадь равна $c_{k-1} \times c_{k-2}$.
- Чтобы найти k нужно заметить, что:
  - На 1, 5, 9, 13... , шагах колобок может выйти за границу только через правую границу.
  - На 2, 6, 10, 14... , шагах только через нижнюю границу.
  - На 3, 7, 11, 15... , шагах только через левую границу.
  - На 4, 8, 12, 16... , шагах только через верхнюю границу.
- Осталось только рассмотреть расстояние до каждой границы и посмотреть, через какую произойдет выход за пределы поля.

Задача C. Постройка забора

Задача C. Постройка забора

- Идея задачи — Андрей Станкевич, Михаил Анопренко
- Подготовка тестов — Григорий Шовкопляс
- Разбор задачи — Илья Збань

- Дан набор из n отрезков
- Нужно посчитать количество способов составить выпуклый многоугольник, используя некоторые из этих отрезков

- Из набора отрезков длинами $a_1, a_2, \ldots, a_k$ можно построить выпуклый многоугольник тогда и только тогда, когда максимальный отрезок короче чем сумма всех остальных

Решение на 33 балла:

- $n \leqslant 20$, поэтому можно решать за $2^n \cdot n$
- Для этого можно перебрать одно из 2^n подмножеств отрезков и проверить условие леммы

Решение на 73 балла:

- Воспользуемся идей динамического программирования
- Отсортируем отрезки по длине от меньшего к большему
- Посчитаем значения динамики $dp_{i,j}$ — количество способов взять среди первых i в отсортированном порядке отрезков несколько так, чтобы сумма их длин была j
- Перебрав i-й элемент как максимальный, можно добавить к ответу $\sum_{j=a_i+1}^{\infty} dp_{i-1,j}$
- $dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-a_i}$

Решение на 100 баллов:

- Оптимизация предыдущего решения с nL до nL^2
- Наблюдение: если сумма отрезков стала больше, чем максимальная длина отрезка l_n, то не важно, насколько
- Оставляем лишь состояния, в которых второй параметр не больше $l_n + 1$.

Задача D. Карандаши

Задача D. Карандаши

- Идея задачи — Илья Пересадин
- Подготовка тестов — Илья Пересадин
- Разбор задачи — Илья Пересадин

- Дано множество множеств a и число k
- Нужно выбрать из него k множеств
- Требуется, чтобы разность максимального и минимального элемента в их объединении была как можно меньше

- Заметим, что в каждом множестве чисел важны только его минимальный и максимальный элемент
- Оставим только их
- Для i-го множества обозначим их за min_i и max_i

Решение на на 31 балл:

- $k \leqslant n \leqslant 20$, поэтому можно решать за 2^k
- Переберем подмножество, которое мы возьмем
- Посчитаем для него минимум и максимум по всем взятым числам
- Найдем разность
- Выберем множество с лучшей разностью

Решение на 62 балла:

- Пусть мы знаем значение наименьшего элемента min_{opt}, который войдет в финальное множество
- Тогда из всех множеств, для которых выполнено $min_{opt} \leqslant min_i$, нужно выбрать $k - 1$ множество, чтобы наибольший взятый max_i был как можно меньше
- Разность $max_i - min_{opt}$ и будет ответом на задачу

Решение на 62 балла:

- Переберем номер opt
- Возьмем все множества, для которых выполнено $min_{opt} \leqslant min_i$ и $opt \neq i$
- Отсортируем их по неубыванию значения max_i
- Выберем первые $k - 1$ множеств — именно они и войдут в оптимальный ответ
- Ответ для множества с наименьшим элементом min_{opt}: $MAX(max_{k-1}, max_{opt}) - min_{opt}$
- В зависимости от реализации, данное решение работает за n^2 или $n^2 log(n)$, что набирает 62 балла

Решение на 100 баллов:

- Отсортируем все множества по значению min_i
- Будем перебирать множества в порядке не убывания min_i
- Поддерживаем кучу с $k - 1$ наименьшим max_i, из множеств, которые мы уже рассмотрели
- На j-м шаге добавляем max_j в кучу и выкидываем оттуда наибольший элемент

- Для каждого j берем максимум в куче. Разность между ним и min_j и будет ответом для j
- Выбираем минимум по ответам для всех j
- Для решения можно было использовать структуру данных `set` в языке C++ или `TreeSet` в языке Java

Задача E. Рассадка зверей

Задача E. Рассадка зверей

- Идея задачи — Илья Пересадин
- Подготовка тестов — Илья Пересадин
- Разбор задачи — Илья Пересадин

- Был циклический массив a из ноликов и единичек
- Посчитали массив s, s_i равно сумме на отрезке $[i - d; i + d]$ массива a
- Вам дан массив s, требуется восстановить a

Решение первой группы ($n \leqslant 20$):

- Переберем исходный массив за 2^n
- Проверим, что все суммы на отрезках $[i - d; i + d]$ совпадают
- Выведем найденный ответ
- Это решение работает за $2^n n$

Решение второй группы ($n \leqslant 1000, d \leqslant 5$):

- Переберем значение первых $2d + 1$ элементов массива
- Очередное значение массива однозначно восстанавливается из уже восстановленных значений a_i и текущей суммы
- Будем двигать окно $[i - d; i + d]$ и восстанавливать массив a. Если удалось восстановить весь массив — выведем ответ
- Это решение за $2^{2d+1} \cdot n$

Решение третьей и четвертой групп:

- Рассмотрим s_i и s_{i+1}.
- $s_i = a_{i-d} + a_{i-d+1} + \ldots + a_{i+d}$
- $s_{i+1} = a_{i-d+1} + \ldots + a_{i+d} + a_{i+d+1}$
- Существует три случая:
  - $s_i = s_{i+1}$, тогда $a_{i-d} = a_{i+d+1}$
  - $s_i > s_{i+1}$, тогда $a_{i-d} = 1$ и $a_{i+d+1} = 0$
  - $s_i < s_{i+1}$, тогда $a_{i-d} = 0$ и $a_{i+d+1} = 1$
- Используя это восстановим столько значений a_i, сколько сможем

- Для оставшихся значений будет выполнено, что $a_{i-d} = a_{i+d+1}$
- Из предыдущих рассуждений следует, что $a_i = a_j$, если $i \equiv j \mod \text{НОД}(n, 2d+1)$
- Тогда достаточно восстановить значения a_i на позициях от 0 до $\text{НОД}(n, 2d+1) - 1$,
- Каждый элемент a_i относится к классу эквивалетности с номером $i \mod \text{НОД}(n, 2d+1)$
- Для каждого класса эквивалентности j посчитаем количество элементов из него, находящихся на отрезке $[0; 2d+1]$, обозначим это количество за c_j

- Теперь нам нужно набрать значениями c_j сумму s_d (без учета уже найденных значений a_i): после этого мы сможем выбранные классы эквивалетности приравнять к единице, остальные к нулю
- Данная задача решается методом динамического программирования за n^2
- Для решения на 100 баллов нужно было заметить, что все c_j равны между собой, поэтому достаточно проверить, что s_d (без учета уже найденных значений a_i) делится на c_j

Задача F. Воздушные потоки

Задача F. Воздушные потоки

- Идея задачи — Илья Пересадин
- Подготовка тестов — Дмитрий Филиппов
- Разбор задачи — Дмитрий Филиппов

- Дан массив высот h, каждый элемент массива — четное число;
- Для каждого его элемента h_i посчитали $G[i] = \max\{j \mid j < i \text{ и } h[j] > h[i]\}$;
- Ввели функцию $S(h) = \sum_{i=1}^{n}(i - G[i])$;
- Разрешается увеличить один из элементов массива, требуется минимизировать $S(h)$.

Научимся считать G за $O(nlogn)$ с помощью структур данных:

- Построим sparse table на максимуме на массиве h;
- Для каждого i двоичным поиском найдем $\max\{j \mid j < i \text{ и } \max(h_j, \ldots, h_{i-1}) < h_i\}$;
- $G_i = j - 1$.

Научимся считать G за $O(n)$ с помощью стека:

- $while(st.top() < h[i])$
- $\quad st.pop();$
- $G[i] = st.top();$

Введем понятие GR:

- $GR[i] = \min\{j \mid j > i \text{ и } h[j] > h[i]\}$;
- Теперь старое G будем обозначать как GL (то есть GL, GR — ближайшие максимумы слева и справа соответственно);
- Подсчитываем GR аналогично GL;
- Построим лес деревьев по массиву h, где $GR[i]$ — предок i.

- Утверждение: h_i имеет смысл заменять только на $h_{GR[i]} + 1$, $h_{GR[GR[i]]} + 1$, ...;
- То есть для каждого элемента его возможные замены — его предки в дереве;
- После замены h_i на $h_{GR^j[i]}$ (j раз перешли к предку), у всех предков i между i и $h_{GR^j[i]}$ GL стал равен i;
- Для данного i переберем количество предков, для которых мы станем GL;
- Тогда $S(h)$ изменится на $diff = \sum_{x=0}^{j}(i - GL[GR^x[j]) + GL_{new}[i] - i$;

- Чему равно $GL_{new}[i]$?;
- h_i мы заменили на $h_{GR^j[i]} + 1$;
- Но так как все числа четные, получаем: $h_{GR^{j+1}[i]} >= h_{GR^j[i]} + 2 > h_{GR^j[i]} + 1 = h_{new}[i]$;
- Значит $GL_{new}[i] = GR^{j+1}[i]$;
- Подставим в формулу изменения $S(h)$: $diff = \sum_{x=0}^{j}(i - GL[GR^x[j]])$;
- Получили, что $diff$ равен сумме по всем предкам i, включая i;
- Вывод — чем больше j, тем больше $diff$.

Получаем сразу решение первых 4 подгрупп:

- Перебираем i;
- Хотим $diff$ как можно больше, поэтому заменяем h_i на $h_{root[i]}$, где $root[i]$ — корень дерева с вершиной i;
- То есть оптимальная замена h_i — на $\max_{j>i} h[j] + 1$ (если этот максимум больше h_i);

- Чтобы решить задачу на 100 баллов, осталось сделать немного;
- Единственное отличие от 81 балла — не всегда можно заменять h_i на $h_{root[i]} + 1$;
- Чтобы найти $\max\{j \mid \text{можно заменить } h_i \text{ на } h_{GR^j[i]} + 1\}$, используем двоичные подъемы по дереву;
- Дополнительно обойдем dfs-ами все деревья и посчитаем суммы GL на префиксах путей из корня;
- Все остальные подсчеты остаются прежними.

Вопросы?