Разбор задач 1-го этапа Всесибирской открытой олимпиады школьников по информатике 2016

Задача 1. Документы

Поскольку изначальная последовательность была отсортирована по возрастанию, то первые $(l-1)$ число будут отсортированы по возрастанию, с l-го по r-е — будут отсортированы по убыванию, и с $(r+1)$-го по n-ое будут снова отсортированы по возрастанию. Значит можно действовать следующим образом: будем просматривать последовательность слева-направо, первый индекс, в котором $a_i > a_{i+1}$ будет равен l. Аналогично, находим r: просматриваем последовательность справа-налево, как только находим такой индекс i, что $a_i < a_{i-1}$, значит мы нашли r. Время работы такого алгоритма $O(n)$. Есть и другой вариант решения. Пусть изначальная последовательность записана в некоторый массив. Теперь сделаем копию этого массива и отсортируем её. Найдём первый и последний индексы, в которых эти два массива различаются, это будут l и r соответственно. В данном случае имеем сложность алгоритма $O(n \cdot \ln(n))$.

Задача 2. Турнир

Для каждого бойца отметим точку на числовой прямой, соответствующую его силовой характеристике. Рассмотрим четыре точки $a < b < c < d$, отмеченные для каких-то бойцов. Несложно понять, что единственный способ минимизировать суммарную предсказуемость — разбить точки на пары (a, b) и (c, d). В этом случае непредсказуемость будет равна $(b-a)+(d-c)$. Таким образом мы показали, что, если есть два поединка: (a, b) и (c, d), то минимальная предсказуемость достигается, когда отрезки $[a, b]$ и $[c, d]$ не пересекаются. Это условие для набора участников с силами $s_1 < s_2 < ... < s_n$ может быть выполнено единственным способом – когда бойцы разбиты на пары (s_1, s_2), (s_3, s_4), ... , (s_{n-1}, s_n). Понятно, что при таком разбиении на пары участник с силой p выиграет свой поединок тогда и только тогда, когда количество чисел s_i, меньших него, нечетное число.

Теперь задача сводится к нахождению количества чисел s_i, меньших p_j для всех j. Вариантов решения много, рассмотрим один из них. Запишем все числа s_i и p_j в один массив и отсортируем его. Теперь за линейное время можно пройти по массиву и, подсчитывая число встреченных s_i, ответить на вопрос задачи для встреченных p_j.

Задача 3. Сладкоежки и торты

Заметим, что если многоугольник имеет центральную симметрию (существует точка, относительно которой многоугольник симметричен), то прямые линии, разделяющие этот многоугольник пополам — те и только те прямые, которые проходят через его центр симметрии.

Таким образом, задача сводится к рассмотрению центров симметрий прямоугольников. Если все центры совпадают, то возможных разрезов бесконечно много. Если же все центры не совпадают, но лежат на одной прямой, то разрез единственен. В противном случае, провести описанный разрез невозможно.

Задача 4. Арбузы

Пусть n делится на m без остатка. В таком случае, в каждой из коробок в конечном счёте будет ровно $k = n/m$ арбузов. Пусть в i-ой коробке изначально лежит a_i арбузов. Теперь для всех i будем считать, сколько арбузов нам нужно перенести между i-ой и $(i{+}1)$-ой коробкой. Несложно показать, что это значение будет равно $|a_1{+}a_2 \ldots{+}a_i{-}i{\cdot}k|$. Ответом будет сумма по всем i.

Теперь рассмотрим вариант, когда n не делится на m без остатка. В таком случае у нас в $u = n \bmod m$ (остаток от деления n на m) коробках в результате будет находиться $k + 1$ арбуз, в остальных — k арбузов, где $k = n \operatorname{div} m$ (целая часть при делении n на m). Будем решать задачу методом динамического программирования. Пусть $dp[i][j]$ обозначает количество перекладываний между коробками с номерами 1 и 2, 2 и 3, …, $i{-}1$ и i, так, что среди первых i коробок будет j таких, в которых $k{+}1$ арбуз. Заполнение значений массива $dp[i][j]$ происходит аналогично случаю, когда n делится на m.

Задача 5. Бомбослав и дорога в Берляндию

В задаче описан взвешенный граф с N вершинами и M рёбрами, веса которых — целые числа от 1 до 10. Требовалось найти путь из вершины 1 в вершину N, веса рёбер на котором не убывают. Для решения этой задачи можно воспользоваться алгоритмом обхода графа в ширину или глубину. Заметим, что недостаточно просто обойти граф, храня вес ребра, посещённого на предыдущем шаге. Будем выполнять обход на модифицированном графе, вершинами которого являются пары (v, c), где v — вершина исходного графа, в которой мы находимся на данный момент, а c — минимальный допустимый вес ребра, по которому мы можем пойти на данный момент. В таком графе допустимы переходы из (u, c) в $(v, \tilde{c})$, если в исходном графе было ребро из u в v веса $\tilde{c} \geq c$. В полученном графе требуется найти путь из вершины $(1, 1)$ в вершину (N, c) для какого-нибудь c.