Разбор задач финала олимпиады ИнфоТехКвест 2022

Задача A. Турникет на КПП

Организуем удобное хранение данных. Сохраним входные данные в словаре так, чтобы для каждого кадета был свой список событий. Сразу при считывании переведем время из формата hh:mm в минуты по формуле 60hh + mm.

Далее для каждого кадета определим, является ли он нарушителем. Заметим, что в условии гарантируется, что у одного кадета не бывает двух событий одного типа подряд. А, следовательно, если для кадета есть два события выход, то между ними гарантированно есть событие вход. Таким образом, чтобы определить по каждому кадету является он нарушителем или нет, пройдемся в хронологическом порядке по его списку событий, и проверим, есть ли в нем два события типа выход в течение dt минут.

Также важно не забыть отсортировать список нарушителей перед тем, как выводить ответ.

Задача B. Рацион кадета

Решение 1 (конструктивное)

Подсчитаем количество каждого блюда в меню рациона кадет.

Разработаем алгоритм, который бы конструировал оптимальное распределение:

- на первом месте стоит блюдо, которое встречается чаще всего;
- на втором месте стоит блюдо, которое встречается чаще всего среди всех блюд, кроме первого, а также НЕ совпадающее с первым блюдом, если это возможно;
- на третьем месте стоит блюдо, которое встречается чаще всего среди всех блюд, кроме первого и второго, а также не совпадающее со вторым блюдом, если это возможно;
- и так далее.

Таким образом, если уже поставлено k блюд, на k + 1 место будет поставлено блюдо, которое встречается чаще всего среди оставшихся блюд, а также не совпадающее с k-м, если это возможно.

Доказательство оптимальности полученного распределения блюд опустим и предоставим Вам.

После построения оптимального распределения блюд (ограничения задачи по времени позволяют построить это распределение любым способом), проходимся по нему и считаем индекса единообразия меню.

Решение 2 (аналитическое)

Также, как и в первом варианте решения, подсчитаем количество каждого блюда в меню рациона кадет.

Пусть самое «популярное» блюда встречается в меню m раз.

Заметим, что если $n\ div\ 2 + 1 >= m$ для нечётных n и $n\ div\ 2 + 1 >= m$ для нечётных n (div – операция целочисленного деления), то индекс единообразия меню будет равен 0, так как алгоритм, описанные в Решении 1, не приведет к появлению в меню подряд идущих одинаковых блюд.

Если $n\ div\ 2 + 1 < m$, то индекс единообразия меню можно вычислить по формуле $m - (n - m) - 1$.

Задача C. Посадочная площадка

В данное задаче требуется найти оптимальное решение (наибольший квадрат), а внешний вид входных данных (таблица) наводят на мысль, что эта задача может решаться двумерным динамическим программированием. Проверим так ли это.

1. Оптимальная подструктура

Будем двигаться по участку местности с левого верхнего угла сначала по столбцам, далее по строкам, определяя в каждой клетке наибольшую сторону квадрата, правым нижним углом, которого могла бы быть данная клетка.

Пусть мы находимся в клетке выделенной жёлтым цветом и для всех клеток, находящихся в строках выше и в той же строке и правее данной, решения уже найдены (более «мелкие» подзадачи уже решены).

1	1	1	1	1	1	1
1	2	2	2	2	2	0
1	0	1	2	3	3	0
0	0	1	2			

В зависимости от значения, которое находится в данном квадрате возможны два варианта:

а) если квадрат содержит 0, то данный квадрат не может быть правым нижним углом посадочной площадки, и мы поставим в него значение 0 (сторона наибольшей посадочной площадки с правым нижним углом площадки в данной клетке равна нулю).

Б) если квадрат содержит 1, то данная клетка может стать правым нижним углом посадочной площадки, причем эта посадочная площадка может получиться путём расширения одной из уже существующих площадок, поэтому размер новой площадки можно получить прибавлением единицы к размеру площадок, которые находятся выше, левее или левее и выше на одну клетку.

Причем из трёх полученных значений на надо взять минимальное, чтобы площадка точно содержала только безопасные квадраты.

Обозначим за F(i, j) наибольшую сторону посадочной площадки с правым нижним углом в клетке с координатами (i, j). Составим рекуррентное соотношение для вычисления $\mathrm{F}(i,j)$:

$$\mathrm{F}(i,j) = 1 + \min\big(F(i-1,j), F(i,j-1), F(i-1,j-1)\big)$$

Строго доказательство того, что из оптимального решение подзадач получается оптимальное решение текущей задачи, оставим математикам. На олимпиаде на это нет времени)

2. Перекрывающиеся подзадачи

Из приведённого рисунка очевидно, что решение одной подзадачи в дальнейшем понадобиться для решения одной, двух или даже трёх других задач, которые и будут перекрывающимися.

Осталось определить тривиальную задачу: в первой строке местности наибольший размер посадочной площадки равен 1, если квадрат безопасный, и равен 0 в противном случае.

Задача D. Регламент Олимпиады

Ещё одна задача, в которой требуется найти оптимальное (минимальное количество баллов) решение.

Заметим, что для заданного порогового количества баллов достаточно легко определить количество участников Олимпиады, которые должны пройти в финальный этап. То есть зная ответ на задачу, мы сможем быстро понять, является ли он верным. Это замечание сразу подсказывает метод, которым необходимо решать данную задача – двоичный поиск по ответу. Далее вся сложность заключается в корректной реализации алгоритма двоичного поиска.

Задача E. В наряд

Математической моделью данной задачи является задача определения является ли данное множество отрезков (интервал времени, в котором курсант находятся в наряде) минимальным множеством, покрывающим данный отрезок (временные промежутки от 0 до 10000, на которые разделены сутки). Решение данной задачи проводится методом сканирующей прямой.

Будем хранить в массиве структуру с полями: координата точки, тип точки (начало/конец отрезка), номер отрезка по порядку и длину отрезка, которому принадлежит точка.

Отсортируем массив по возрастанию сравнивая точки по координате, в случае равенства координат сравниваем тип точки (сначала должны идти открывающие), в случае равенства типов сравниваем длину отрезка, которому принадлежит точки. Для открывающихся точек сначала должны идти точки с большей длиной отрезка. Для закрывающих точек сначала должны идти точки с меньшей длиной отрезка.

Далее решаем параллельно 2 классические задачи, решаемые методом сканирующей прямой:

1. Определяем покрыт ли данный отрезок множеством отрезков;

2. Принадлежит ли какой-то отрезок другому отрезку.

Если первая задача дает положительный ответ, а вторая отрицательный, то график удовлетворяет требованиям, иначе не удовлетворяет.

Замечание

Задача является весьма кропотливой в реализации. На написание кода и отладку возможных вариантов можно потрать достаточно много времени. Также данная задача является задачей с бинарным ответом YES/NO, что позволяет набрать определенное количество баллов просто выведя любой из ответов. Сама постановка вопроса задачи также наталкивает на мысль, что сложно определить именно случай, когда верным ответом является NO. Возможно, что большинство тестов, как раз проверяют именно этот случай. Посылка в систему тестирования программы, которая всегда выводи просто NO, покажет, что таким образом можно набрать достаточно много баллов. Таким образом, эта задача предусматривает не только проверку знаний участников Олимпиады метода сканирующей прямой, но и компетенций в анализе условия и возможных входных и выходных данных задачи. Иногда выгоднее не тратить время на идеальное решение, а просто выработать удовлетворительное решение за короткое время, а если останется время поработать и над полным решением.