Разбор задачи «Действительно важные вещи»

Понятно, что выполнять непосредственный подсчет на исходной последовательности чисел для каждого значения q_j — дело достаточно долгое ($O(n^2)$). Поэтому проведем предварительную подготовку.

Будем хранить ответы для каждого значения индекса в специальном массиве $ans[]$. Отсортируем пары (*важность дела*, *номер дела*) по неубыванию важности. Если бы важности всех дел были различны, то заполнение массива $ans[]$ свелось бы к сопоставлению номера пары в отсортированной последовательности номеру дела в этой паре. Однако, когда важности в парах повторяются, нужно действовать аккуратнее.

Один из возможных подходов таков. Рассмотрим пару (*важность дела*, *номер дела*) на позиции j в отсортированной последовательности пар, такую, что пара на позиции $(j-1)$ имеет меньшую важность дела. Запомним позицию j и будем двигаться вправо по отсортированной последовательности до тех пор, пока не встретим (первую) пару, у которой значение важности дела окажется отличающимся от значения важности в паре на позиции j. Пусть это пара находится на позиции k ($k > j$). Тогда (полагаем, что нумерация элементов отсортированной последовательности ведется с нуля) число k и будет количеством дел, которые отложит Джефф, если в качестве «эталона» он выберет дела с номерами, встретившимися в парах с j по $k-1$ включительно. Пройдем теперь еще раз от позиции j до позиции $k-1$ (или в обратном порядке, это уже вопросы реализации) и сопоставим всем номерам дел этих пар число k в массиве $ans[]$. Асимптотика сортировки составит $O(n \cdot log\, n)$, заполнения массива $ans[]$ — $O(n)$.

Теперь, когда массив $ans[]$ сформирован, ответить на любой запрос q_j мы можем за $O(1)$. Итоговая асимптотика решения — $O(n \cdot log\, n)$.

Разбор задачи «Организация труда»

Будем решать задачу методом динамического программирования. Обозначим через $dp[i]$ максимальную суммарную эффективность, которую можно получить, если начать работу в минуту $\#i$ (и оптимальным образом работать до истечения времени t).

Понятно, что если Джефф начал работать в момент времени $\#i$, то в следующий раз он начнет работать не раньше, чем в момент времени $\#(i+n+m)$ (т.е. проработает n минут, затем поспит m минут, и сразу после этого приступит к работе). Однако Джефф может лечь спать не сразу по окончании работы. Впрочем, откладывать сон слишком надолго не имеет смысла: если момент времени, в который в следующий раз начнет работать Джефф, больше или равен $\#(i+n+m+n+m)$, то в этот «период непродуктивного ожидания» уже поместится n минут работы и m минут сна, после которых он может вновь совершать выбор, что именно делать. Таким образом, $dp[i] = w[i] + \max\{dp[j] \mid j = (i+n+m)\,..\,(i+n+m+n+m-1)\}$, где $w[i] = \sum_{k=i}^{i+n-1} f[k]$. Значения $w[i]$, разумеется, легко вычислить заранее (в один проход по массиву $f[]$, асимптотика $O(n)$).

Массив $dp[]$ нужно будет заполнять справа налево. Также нетрудно заметить, что нет необходимости каждый раз вычислять максимальное значение $dp[j]$ из нужного диапазона: при движении влево максимальное из доступных значений $dp[j]$ уменьшаться не может. Поэтому при переходе от $dp[i+1]$ к $dp[i]$ достаточно сравнить уже найденный максимум со значением $dp[i+n+m]$ (конечно, при условии, что $i+n+m$ не превосходит t). Поэтому заполнить массив $dp[]$ также можно за один проход. Асимптотика решения составляет $O(n)$ (хотя ограничения допускают и квадратичное решение).

Разбор задачи «Плейлист»

Сопоставим каждому названию композиции число от 0 до $n-1$ — так с ними будет удобнее обращаться. Это можно сделать с помощью какой-либо «словарной» структуры данных (HashMap в Java, map в C++).

Затем воспользуемся методом динамического программирования. Обозначим через $d[mask][i]$ количество таких непротиворечивых вариантов, что на на первых k местах идут только композиции

с номерами, которым соответствуют единичные биты в $mask$, при этом на последнем месте ($\#k$) находится композиция номер i (k — количество единичных битов в $mask$).

База динамики: $d[1 << i][i] = 1\ \forall i \in [1, n]$.

Пересчёт: $d[mask][i] = \sum_{j \neq i} d[mask \oplus (1 << i)][j]$, причём композиция $\#i$, идущая за композицией $\#j$, должна быть допустимой и не нарушать порядок композиций. Это не выполняется только в трёх случаях:

- для композиции $\#j$ известна следующая, и это не композиция $\#i$;
- для композиции $\#i$ известна предыдущая, и это не композиция $\#j$;
- для композиции $\#i$ известна следующая, но она уже была использована (соответствующий ей бит в $mask$ равен единице). (Разумеется, исключаются ситуации, когда композиция $\#i$ — последняя.)

Во всех остальных случаях можно использовать композицию $\#i$ после композиции $\#j$.

Для каждого значения количества единичных битов k в маске (оно равно текущему количеству композиций) посчитаем $count[k] = \sum_{i=1}^{i \leqslant n} d[mask][i]$, где $mask$ содержит ровно k единичных битов. Тогда вероятность, что мы обнаружим «неправильный» порядок композиций, вычисляется так: $res[i] = (1 - res[i-1]) \cdot (1 - count[i]/total[i]))$, где $total[i] = n!/i!$, $i > 1$. Заметим, что $res[1] = 0$ всегда, поскольку невозможно сделать вывод о том, в какой последовательности исполняются композиции, прослушав только одну.

Итоговая асимптотика составит $O(2^n \cdot n^2)$: $2^n \cdot n$ состояний динамики, каждое из которых пересчитывается за $O(n)$.

Разбор задачи «Салаты»

Формально, мы стремимся максимизировать $x+y$ при следующих ограничениях: $f_i \cdot x + s_i \cdot y \leqslant d_i$, $\forall i = 1, 2, \ldots, n$, $x, y \geqslant 0$.

Заметим, что каждое ограничение задаёт некую полуплоскость, ограниченную прямой, на плоскости XY, а их пересечение образует выпуклый многоугольник F (он лежит в одной полуплоскости относительно прямой, проходящей вдоль каждой его стороны).

Запустим бинарный поиск по величине ответа $x + y$: найдем максимальное значение величины c, при котором значение функции $f(c) = 1$, если прямая пересекает многоугольник F, и 0 — в противном случае.

Применение бинпоиска корректно на отрезке $[0, \max d_i + 1]$, так как выполняются следующие условия относительно $f(c)$ на данном отрезке:

- $f(c)$ является монотонно неубывающей в силу выпуклости F;
- $f(0) = 1$;
- $f(d + 1) = 0$.

Чтобы упростить отыскание значения $f(x + y)$, преобразуем исходные ограничения следующим образом:

- $f_i \cdot x + s_i \cdot y \leqslant d_i$;
- $s_i \cdot y \leqslant d_i - f_i \cdot x$;
- $(s_i - f_i) \cdot y \leqslant d_i - f_i \cdot (x + y)$.

Здесь достаточно рассмотреть три случая:

1. $s_i - f_i = 0$ — тогда ограничение не выполняется только в случае $0 > d_i - f_i \cdot (x + y)$;
2. $s_i - f_i > 0$ — каждое такое ограничение можно преобразовать к виду $y \leqslant (d_i - f_i \cdot (x+y))/(s_i - f_i)$;

3. $s_i - f_i < 0$ — каждое такое ограничение можно преобразовать к виду $y \geqslant (d_i - f_i \cdot (x+y))/(s_i - f_i)$.

В такой формулировке $f(x+y) = 1$, если выполнились все ограничения 1 типа, а минимальное значение y по ограничениям 3 типа меньше или равно максимального значения y по ограничениям 2 типа.

Проверка всех ограничений в одной итерации бинпоиска происходит за $O(n)$, итоговая сложность решения $O(n \cdot \log(\max d_i))$

Разбор задачи «Шары»

Для решения задачи можно разобрать четыре случая.

В каждом из этих случаев мы будем учитывать два момента. Во-первых, какого бы цвета ни оказался первый шар, Джефф всегда помещает его на елку за одну единицу времени. Во-вторых, как можно большее количество остальных шаров мы должны размещать на елке за две единицы времени.

Пусть t — нечетное. Возможны два варианта: $t = 2 \cdot (2 \cdot k + 1) + 1$ и $t = 2 \cdot (2 \cdot k) + 1$.

Первый вариант наиболее простой. Понятно, что $t = 2 \cdot (2 \cdot k + 1) + 1$ «раскладывается» так: 1 единица времени тратится на помещение на елку первого шара, после чего по две единицы времени будут израсходованы на помещение на елку количества шаров, равного $(2 \cdot k + 1)$. Это число нечетное, и поэтому вместе с первым шаром можно построить чередующуюся (цветами) последовательность шаров, в которой будет равное количество шаров обоих цветов. Пример такой последовательности для $t = 7$: $RBRBRB$. Как легко видеть, шаров каждого цвета будет $k + 1$.

Второй вариант несколько сложнее. Опять же ясно, что 1 единица времени будет истрачена на помещение на елку первого шара, но далее, если мы будем строить последовательность так же, как в первом варианте, количество шаров разных цветов у нас окажется разным. Поэтому поступим следующим образом: представим $t = 2 \cdot (2 \cdot k - 2) + 4 + 1$. Иными словами, мы построим «цветочередующуюся» последовательность из $k - 1$ пары шаров разных цветов. При этом у нас останется 4 единицы времени, за которые мы должны разместить на елке три шара (понятно, что для размещения одного шара 4 единицы времени — это слишком много).

Если мы будем считать, что достали первым красный шар (R), то затем у нас будет $(k - 1)$ пара BR. Тогда из трех шаров два должны быть синими и один — красным. Это можно сделать следующим образом: сначала вытащить красный шар (и потратить на это 1 единицу времени), затем синий (который придется вернуть и снова вытащить, потратив суммарно 2 единицы времени), после чего еще один синий (за 1 единицу времени). Пример последовательности для $t = 9$: $RBRRBB$. Вновь несложно увидеть, что шаров каждого цвета будет $k + 1$.

Теперь рассмотрим случай, когда t — четное. Здесь также будет два варианта, которые мы, поддерживая сквозную нумерацию, будем называть третьим и четвертым.

Третий вариант — $t = 2 \cdot (2 \cdot k + 1) = 2 \cdot (2 \cdot k) + 1 + 1$. Как понятно, 1 единица времени будет истрачена на размещение первого шара, затем мы можем создать k «цветочередующихся» пар, после чего у нас останется 1 единица времени, за которую мы никак не сможем поместить шар нужного цвета. Поэтому придется применить уже знакомый по второму варианту «трюк»: $t = 2 \cdot (2 \cdot k - 2) + 5 + 1$. В этом случае мы уже должны будем попробовать поместить на елку за 5 единиц времени 3 шара, соблюдая баланс по цветам.

Если мы вновь предположим, что первым на елку был помещен красный шар (R), то за ним будут следовать $(k - 1)$ пар вида BR, после чего будет нужно поместить за 5 единиц времени 2 синих и 1 красный шар. Это несложно сделать: сначала нужно вытащить синий шар (который придется вернуть, а затем достать вновь и повесить на елку; на это уйдет 2 единицы времени), затем еще один синий шар (и повесить его на елку за 1 единицу времени), и, наконец, достать последний красный шар (на его размещение будет потрачено 2 единицы времени). Пример последовательности для $t = 10$: $RBRBBR$. Снова нужно отметить, что шаров каждого цвета будет $k + 1$.

Последний, четвертый, вариант — $t = 2 \cdot (2 \cdot k)$. Попробовав выделить из него 1 единицу времени на помещение на елку первого шара, можно достаточно быстро понять, что придется разложить t следующим образом: $t = 2 \cdot (2 \cdot k - 4) + 7 + 1$. Т.е. мы повесим первый шар, получим $(k - 2)$ «цветочередующиеся» пары, после чего попробуем за 7 единиц времени разместить 5 шаров (очевидно, что для 3 шаров это время слишком большое).

Опять же, полагая, что первым на елке был размещен красный шар (R), за которым следовало $k-2$ пары вида BR, за 7 единиц времени будет нужно разместить на елке 3 синих шара и 2 красных. Это можно сделать так: сначала вытащить красный шар (и за 1 единицу времени повесить его на елку), затем подряд три синих (на первый из них будет потрачено 2 единицы времени, а на второй и третий — по одной единице времени), и еще один красный шар (который будет размещен на елке за 2 единицы времени). Пример последовательности для $t = 8$: $RRBBBR$. И опять же, шаров каждого цвета будет $k + 1$.

Разбор задачи «Точный расчет»

Ограничения позволяют решать эту задачу симуляцией описанного в условии процесса.

Представим входные данные более удобным образом: сформируем список пар (*цвет*, *количество*). Переберём все пары цвета c в новом списке.

Для каждой пары с соответствующим цветом сначала проверяем, что (*количество* + 1) окажется не меньше, чем k. Если это так, то фрагмент цепочки, соответствующий этой паре, самоуничтожится, и мы перейдем к левому и правому соседям этой пары в списке пар. После этого, пока соседние пары с обеих сторон существуют и могут самоуничтожиться в случае слияния (т.е. имеют одинаковый цвет и суммарную длину, не меньшую k), наращиваем счетчик количества уничтоженных шариков.

Полученное для очередной пары цвета c значение сравним с уже имеющимся максимальным и при необходимости обновим последнее.

Разбор задачи «Набор паролей»

Формально необходимо найти набор чисел $a = [a_0, a_1, \ldots, a_k]$, таких, что:

$\sum_{i=0}^{k} a_i = n;$

$\sum_{i=0}^{k} {a_i}^2 = m;$

$1 \leqslant a_i \leqslant 9, \quad \forall i = 0 \ldots k.$

Преобразуем вторую формулу и получим следующее:

$\sum_{i=0}^{k} a_i = n \qquad (1)$

$\sum_{i=0}^{k} ({a_i}^2 - a_i) = m - n \qquad (2)$

$1 \leqslant a_i \leqslant 9, \quad \forall i = 0 \ldots k.$

Для каждого набора чисел определим функцию $S(a) = (x, y)$, где x и y — суммы, вычисленные по формулам (1) и (2). Заметим, что если $S(a_0, a_1, \ldots, a_k) = (x, y)$, то $S(a_0, a_1, \ldots, a_k, 1) = (x + 1, y)$.

Пользуясь данным фактом, мы можем решить задачу в два этапа следующим образом:

1. найдём набор чисел a такой, что $S(a) = (x,\ m - n), x \leqslant n$;
2. получим набор b добавлением к набору a единиц, так что $S(b) = (n,\ m - n)$.

Реализация второго этапа достаточно проста. Для решения первого этапа воспользуемся динамическим программированием. Пусть $dp[y]$ — минимальное среди x, таких, что существует подходящий под условие задачи набор a: $S(a) = (x, y)$. Базовое состояние и переходы для данной динамики следующие:

$dp[0] = 0$

$dp[i] = \min(dp[i - (j^2 - j)] + j), \quad \forall j = 2 \ldots 9.$

В случае, если $dp[m - n] > n$, искомого множества не существует. В противном случае нам остаётся только восстановить ответ (достаточно хранить для каждого состояния динамики цифру, которая даёт оптимальный ответ).

Итоговая сложность решения составляет $O(k \cdot m + n)$, где $k = 8$ — количество возможных цифр-переходов в динамике.

Разбор задачи «Здание»

Четырёхугольник является квадратом, если он является одновременно ромбом с равными диагоналями, поэтому достаточно проверить выполнение следующих условий:

- Все стороны четырехугольника равны — в этом случае он является ромбом.
- Обе диагонали четырехугольника равны между собой.

Во избежание проблем с точностью достаточно сравнивать квадраты расстояний между точками (при данных ограничениях можно использовать целочисленный тип).

Также необходимо проверить, что квадрат невырожденный, т.е. удостовериться, что длины сторон и диагоналей больше 0.

Разумеется, это не единственно возможное решение. Можно было также вычислить векторное произведение для каждой пары сторон квадрата, взятых в порядке обхода.

Разбор задачи «Лестницы»

Для каждого цвета квадрата найдём наименьшее и наибольшее количество ступенек, после которого он был получен. Теперь для каждого цвета j посчитаем min_j — минимальное значение x, такое, что верхняя граница для этого цвета всё ещё включает наибольшее количество ступенек для этого цвета во входных данных. Это можно сделать с помощью бинпоиска (или вывести формулу). Аналогично посчитаем max_j — максимально возможное значение x, при котором нижняя граница текущего цвета включает в себя наименьшее количество ступенек для этого цвета во входных данных.

Далее, найдём $lower = \max(min_j)$ и $upper = \min(max_j)$. Если $lower \leqslant upper$, то эти числа и являются ответом, иначе ответа не существует. Заметим, что изначально нужно принять $lower$ за 1, а $upper$ за $2 \cdot 10^9$.

Итоговая асимптотика: $O(n)$.

Разбор задачи «Пропущенный вызов»

Ограничения позволяют решить данную задачу эмуляцией процесса.

Будем хранить список номеров, с которых уже поступали звонки. В случае поступления звонка с номера, уже содержащегося в списке, перенесем его на первую позицию и сдвинем оставшуюся часть списка за $O(m)$.

Итоговая сложность решения $O(n \cdot m)$. Разумеется, можно было написать более эффективное решение, использующее структуры данных.

Разбор задачи «Окно возможностей»

В этой задаче фактически требовалось аккуратно найти максимум, и главная сложность состояла в том, чтобы учесть все условия. Нужно было последовательно рассматривать пары дней (идущих подряд), которые с точки зрения Джеффа являлись благоприятными, и выбирать наибольшее значение параметра благоприятности среди подходящих пар.

Разбор задачи «Обстоятельства»

Вероятно, наиболее простое решение выглядит следующим образом. Джефф может заниматься подготовкой к проекту от одного до n дней. Поскольку в каждый из дней он будет заниматься проектом не менее $2 \cdot m$ и не более $3 \cdot m$ минут, то если он может подготовиться к встрече за k дней, для величины p должно выполняться неравенство: $2 \cdot m \cdot k \leqslant p \leqslant 3 \cdot m \cdot k$. Выполнение этого неравенства и нужно проверить для всех k от 1 до n.