Разбор задачи «Продукты в экспедиции»

Автор задачи: Дмитрий Саютин

В первой подзадаче у есть только один тип продуктов. Нужно просто проверить, можно ли съесть этот тип продукта. За время t_1 все участники экспедиции смогут съесть $c \cdot t_1$ порций, поэтому если $k_1 \leqslant c \cdot t_1$, то ответ 1, иначе ответ 0. Время работы этого решения $O(1)$.

Как для множества типов продуктов проверить, можно ли его съесть полностью? Пусть множество состоит из m типов продуктов, таких что время, за которое они портятся это $t_1, t_2, \ldots, t_m$, количество порций продуктов $k_1, k_2, \ldots, k_m$, соответственно. Отсортируем их по возрастанию времени, то есть пусть $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_m$. Заметим, что для всех $1 \leqslant i \leqslant m$ должно быть выполнено, что $k_1 + k_2 + \ldots + k_i \leqslant c \cdot t_i$. Это так, потому что время, за которое мы сможем есть первые i типов продуктов это t_i, при этом за это время мы должны будем съесть как минимум $k_1 + k_2 + \ldots + k_i$ порций. Докажем, что это также является достаточным условием того, что мы сможем все съесть. Заметим, что $k_1 \leqslant c \cdot t_1$, поэтому за первые t_1 дней мы сможем съесть все порции продукта первого типа. Затем, так как $k_1 + k_2 \leqslant c \cdot t_2$, мы за первые t_2 дней сможем съесть все порции первого и второго типа, так как на продукты первого типа мы потратили ровно k_1 возможностей съесть одну порцию. И так будет для любого i: поскольку $k_1 + k_2 + \ldots + k_i \leqslant c \cdot t_i$, мы сможем съесть за t_i дней все продукты первых i типов, потому что мы съели все продукты первых $i - 1$ типов, потратив на это ровно $k_1 + k_2 + \ldots + k_{i-1}$ возможностей съесть оджну порцию.

Во второй подзадаче переберем все 2^n подмножеств типов продуктов. Для каждого из них, за время $O(n \log n)$ или $O(n)$ (если предварительно отсортировать все t_i) мы сможем определить, можно ли съесть все порции выбранного множества типов продуктов. Среди всех таких подмножеств найдем подмножество максимального размера, это будет ответом. Время работы этого решения $O(2^n n)$.

Для того, чтобы решить третью подзадачу, отсортируем все t_i по возрастанию. Мы должны выбрать некоторое подмножество типов продуктов, которое удовлетворяет полученному нами критерию. Обозначим за $T = t_n$ — максимальное значение t_i. По ограничениям этой подзадачи $T \leqslant 2000$. Посчитаем $dp[i][s]$ — максимальное количество типов продуктов, которое можно выбрать, так чтобы его можно было съесть, сумма значений k_j по всем продуктам из множества равна s и продукт i это максимальный номер продукта, взятый в множество. Эти значения мы будем считать для $1 \leqslant i \leqslant n$ и $1 \leqslant s \leqslant T$. Тогда $dp[i][s] = \max\limits_{1 \leqslant j < i} dp[j][s - k_i] + 1$, если $s \leqslant t_i$ и $dp[i][s] = 0$, иначе. Это позволяет нам вычислить все значения за время $O(n^2 T)$, что недостаточно чтобы решить задачу. Для того, чтобы ускорить вычисление, обозначим $pref[i][s] \max\limits_{1 \leqslant j \leqslant i} dp[j][s]$. Тогда $dp[i][s] = pref[i-1][s - k_i] + 1$ и $pref[i][s] = \max(pref[i-1][s], dp[i][s])$. Вычислим значения за время $O(nT)$ и ответом будет являться максимум $dp[i][s]$ по всем $1 \leqslant i \leqslant n, 1 \leqslant s \leqslant T$.

Пропустим четвертую подзадачу и разберем сначала пятую. В этой подзадаче все значения t_i совпадают. Обозначим их за t. Тогда в этом случае критерием того, что мы можем съесть все продукты, количества которых $k_1, k_2, \ldots, k_m$ является одно неравенство, а именно $k_1 + k_2 + \ldots + k_m \leqslant c \cdot t$. То есть мы хотим взять как можно больше типов продуктов, таких, что сумма их количеств не превышает $c \cdot t$. Для этого можно воспользоваться жадным алгоритмом: отсортируем все значения k_i и будем набирать их от меньшего к большему, пока сумма не превысит $c \cdot t$. Асимптотика решения этой подзадачи $O(n \log n)$.

Далее приведем концепцию полного решения. Отсортируем все типы продуктов по убыванию времени, за которое они портятся, то есть $t_1 \geqslant t_2 \geqslant \ldots \geqslant t_n$. Будем перебирать i от 1 до n и смотреть, какие продукты надо есть в обратном порядке времени. Будем хранить множество типов продуктов и для каждого типа сколько еще порций этого типа осталось. Перебирая i добавим тип продукта i и то, что осталось k_i порций этого типа. Теперь до следующего добавления типа продукта мы можем съесть $c \cdot (t_i - t_{i-1})$ порций (положим $t_0 = 0$). Заметим, что теперь выгодно начать есть те типы продуктов, которых осталось меньше всего, потому что разные типы продуктов к этому моменту равны между собой и невыгодно есть тип, которого больше. Поэтому будем есть продукты в порядке возрастания оставшихся порций этого типа, пока количество порций, которые мы можем сейчас съесть не будет равно 0 или пока все типы продуктов не кончатся. Если мы доедаем какой-

то тип продуктов, то добавим его индекс в ответ. Таким образом мы получим множество типов продуктов максимального размера, которое можно съесть.

Для решения четвертой подгруппы можно наивно реализовать это решение — будем хранить массив типов продуктов и количество оставшихся порций. При выборе типа продукта, который надо съесть, линейным поиском найдем минимальное количество и будем есть порцию этого типа. Тогда реализация этого решения будет работать за время $O(n^2)$.

Для полного решения задачи будем хранить типы продуктов в структуре данных s типа, например, `std::set`, сортируя их по количеству оставшихся порций этого типа. При добавлении нового типа, просто положим его в s, если мы хотим съесть $c \cdot (t_i - t_{i-1})$ порций, будем вынимать из s тип продукта с минимальным количеством оставшихся порций и есть. Если количество оставшихся порций будет равно 0, то не будем возвращать этот тип в s, иначе вернем с новым количеством оставшихся порций. Суммарное время работы такого решения будет равно $O(n \log n)$, потому что каждый тип продуктов будет удален из s не более одного раза и мы n раз будем добавлять тип в s и n раз обновлять количество оставшихся порций.