Задача A. Нужно больше менеджеров!

Каждый сотрудник — это двоичное число. Расстояние между сотрудниками — это количество бит, в которых они отличаются.

Переформулируем задачу. Дан полный граф из m вершин. Каждой вершине соответсвует число. Стоимость ребра между парой вершин определена выше. В этом графе нужно найти стоимость минимального остовного дерева `http://e-maxx.ru/algo/mst_kruskal`.

Первая подзадача — нужно аккуратно написать примерно то же, что делает алгоритм Крускала, только сортировать подсчетом, так как вес каждого ребра не больше 20.

Идея полного решения точно такая же, как у задачи «Палитра» с республиканской олимпиады 2016. Возьмем новый граф, в котором 2^n вершин, а ребро между двумя вершинами будет только если они отличаются ровно в одном бите.

Для нахождения минимального остовного дерева будем использовать алгоритм Прима, а не Крускала (можно почитать на емаксе). Алгоритм Прима каждый раз выбирает вершину из тех, что не были выбраны раньше, такую, что у нее минимален вес входящего в нее ребра, второй конец которого — уже выбранная вершина. Как на очередной итерации (которых m) алгоритма Прима находить вершину с минимальным веса ребра, и из нее релаксировать расстояния до остальных вершин? Чтобы быстро найти вершину, опять используем идею сортировки подсчетом — так как каждый вес ребра не более 20, то храним 20 векторов, i-й из которых содержит все вершины v, что до них есть есть ребро веса i от одной из заданных вершин. Чтобы выбрать вершину на очередной итерации, нужно взять ее из самого маленького по номеру вектора, в котором хоть что-то есть. После этого релаксируем все расстояния с помощью `BFS`, точно так же, как в задаче «палитра» и аккуратно обновляем расстояния до вершин и наши 20 векторов.

Итого решение за $O(2^n \cdot n^2)$. Если туго с пониманием, как это написать, можно попробовать посмотреть и разобраться в моей реализации: `https://ideone.com/CkClFQ`

Задача C. Subsequence queries

- Решение за $O(nm)$ на запрос.

  Будем рассматривать все элементы с l по r, можем выписать их явно (в массив). Будем считать $dp_{i,j}$ $(0 \leqslant i \leqslant r-l+1, 0 \leqslant j < m)$— количество подпоследовательностей из первых i элементов, таких, что, остаток их суммы при делении на m равен j.

  База: $dp_{0,0} = 1$, $dp_{0,j} = 0$

  Переход: $dp_{i,j} = dp_{i-1,j} + dp_{i-1,(j-a_i) \mod m}$

  Ответ: $dp_{r-l+1,0}$.

- Для четвертой подзадачи можно было предпосчитать все значения ДП для всех отрезков за $O(n^2)$ и отвечать на запрос за $O(1)$.

- Ускоряем с деревом отрезков, решение за $O(m^2 \log n)$ на запрос. 75 или может даже 87 баллов.

  Храним в вершине дерева отрезков массив длины m, i-й элемент — количество подмножеств с суммой i. Для листа с индексом num этот массив выглядит как $[1, 0, 0, \ldots, 0, 1, 0, 0, \ldots, 0]$, где вторая единица стоит в позиции a_{num}.

  Пусть нужно объединить две вершины a, b. Тогда новые значения вершины c получаются следующим образом:

  $c_i = \sum_{j=0}^{m-1} a_j \cdot b_{i-j}$, где $\sum_{l}^{r}$ — обозначение суммы с l по r.

  то есть если в левом поддереве мы возьмем подмножество с суммой j, то в правом нужно взять подмножество с суммой $i - j$, и так нужно просуммировать по всем $j = 0..m-1$.

  Таким образом обновление по двум вершинам за $O(m^2)$, на запрос надо объединить $O(\log n)$ пар вершин.

- 100 баллов — разделяй и властвуй!

  Посмотрим на элемент в середине массива: $m = (1 + n)/2$. Все запросы, у которых $l \leqslant m \leqslant r$, обязательно «проходят» через него.

  Насчитаем две ДПшки как были выше - одну влево от m, а одну вправо. Пусть первая дпшка $dpL_{i,j}$ $(1 \leqslant i \leqslant m)$ — количество подмножеств с суммой j среди элементов $[i..m]$. Аналогично $dpR_{i,j}$ $(m \leqslant i \leqslant n)$ — количество подмножеств с суммой j среди элементов $[m..i]$.

  Левая ДПшка считается за $m \cdot \frac{n}{2}$, за столько же правая, суммарно — $O(nm)$.

  Используя эти дпшки, можем ответить на все запросы l r, которые проходят через m. Ответ на них получается с помощью элементов $dpL_{l,*}$ и $dpR_{r,*}$, он равен $\sum_{j=0}^{m-1} dpL_{l,j} \cdot dpR_{r,(m-j) \mod m}$.

  Таким образом ответ на запрос получается за $O(m)$.

  Рекурсивно разделимся на две половины и там сделаем то же самое. В каждой половине насчитаем две дпшки за $m \cdot \frac{n}{4}$, суммарно 4 дпшки, суммарное время на ДПшки будет снова $4 \cdot m \cdot \frac{n}{4} = O(nm)$. Ответим снова на все запросы, которые можем, а потом снова запустимся рекурсивно и поделимся на 8 частей.

  Разделяться можно не больше $\log_2 n$ раз, поэтому суммарно дп считается $O(nm \log n)$ времени. Каждый запрос может побывать на каждой «глубине» рекурсии не более одного раза, значит суммарно ответы на запрос занимают $O(q(m + \log n))$.

  Итого $O((n \log n + q)m + q \log n)$.