Разбор задачи «Капли»

Посчитаем, сколько раз каждая капля упадет между двумя последовательными очистками трубы и просуммируем эти значения по всем каплям: $sum = \sum_{i=1}^{n} \lfloor \frac{k}{p_i} \rfloor$. Также посчитаем, сколько капель упадет с каждой трубы за последние $t \bmod k$ секунд сбора капель и просуммируем эти значения: $rem = \sum_{i=1}^{n} \lfloor \frac{t \bmod k}{p_i} \rfloor$. Затем вычислим, сколько очисток трубы произойдет за t секунд: $cnt = \lfloor \frac{t}{k} \rfloor$.

Ответом является следующая сумма: $cnt \cdot sum + rem$.

Разбор задачи «Газорпазорп»

- Если k нечетное, то всегда выигрышной является стратегия «назвать предыдущее число». Действительно, тогда мы всегда будем называть его последними, и когда мы назовем число $k-1$ раз, первому игроку придется назвать какое-то другое число, а мы продолжим нашу стратегию.

- Если k четное, то сначала предподсчитаем массив выигрышных-проигрышных позиций (тут нам наоборот называть когда-либо предыдущее число нет смысла, потому что по четности мы проиграем). Если хотя бы одна из позиций $x_{prev} - 1..x_{prev} - m$ — проигрышная, назовем это число и выиграем. Если нет, мы точно проиграем при оптимальной игре первого игрока и можно сразу же сказать «`I'm giving up`». Замечание: если первый игрок после нашего хода называет то же самое число, ответим ему тем же — теперь он в итоге проиграет по четности.

Разбор задачи «Морти и пароль»

Находим каждый элемент максимально возможной перестановки по очереди, начиная с самого первого. На его место может встать один из первых трех элементов. Берем из них максимальный, перемещаем его влево, если нужно. Считаем, сколько раз каждый из элементов уже был перемещен, и, опираясь на это, рассматриваем возможные варианты для элемента, который должен располагаться на следующей позиции и так далее. Заметим, что на каждом шаге будут израсходованы операции не более, чем у двух следующих элементов.

Разбор задачи «Матрица Рика»

Матрица, которая должна получиться, однозначно задаётся своей верхней левой подматрицей размера $\lceil \frac{n}{2} \rceil$ на $\lceil \frac{m}{2} \rceil$.

Переберём клетку из этой подматрицы. Рассмотрим те клетки, которые должны быть ей равны, то есть такие, которые получаются из неё отражениями относительно центральной горизонтали или вертикали. Таких клеток может быть четрыре, две или одна. Возьмём значение, которое встречается среди них большее число раз, и присвоим его остальным.

Разбор задачи «Кошелёк»

Заметим, что для того, что бы положить купюру типа b_j, то надо либо переложить все купюры типа меньшего, чем b_j в стопку s_l, либо переложить все купюры типа большего, чем b_j в стопку s_r. Значит минимальное количество действий, которое надо для этого сделать — минимум из количества купюр, типа меньшего чем b_j, и количества купюр, типа большего чем b_j.

Заведем дерево отрезков на сумму, которое будет считать сумму количеств имеющихся купюр каждого типа. Тогда для ответа на запрос, нужно посчитать сумму первых $b_j - 1$ элементов, и сумму последних $n - b_j$ элементов, и взять из них минимум.

Разбор задачи «Безопасное путешествие»

Существует всего 3 возможных вида графов, удовлетворяющих условиям задачи:

- Полный граф — $\frac{n \cdot (n-1)}{2}$ ребер

- Цикл — n ребер

- Полный симметричный двудольный граф — $(\frac{n}{2})^2$ ребер

Подробнее о том, почему только такие варианты подходят, можно прочитать по ссылке http://stu.alnam.ru/book_grnet-24 (стр. 59).

Проверим, подходят ли заданные в условии n и m хотя бы один из этих вариантов и выведем ответ, если совпадение найдено.

Разбор задачи «Морти и подпоследовательности»

Для решение данной задачи нам понадобится сделать небольшой предподсчет. Вычислим следующую динамику $lis_{l,r}$ — длина максимальной возрастающей подпоследовательности в исходном массиве на отрезке с l по r заканчивающаяся в элементе с индексом r. Сделаем это следующим образом: зафиксируем левую границу l, после чего начнем перебирать правую границу r принимая во внимание факт, что r-й элемент будет последним в данной подпоследовательности. Теперь нужно перебрать предыдущий элемент входящий в данную подпоследовательности j, такой, что $l \leqslant j < r$, и обновить значение $lis_{l,r} = max(lis_{l,r}, lis_{l,j} + 1)$.

После того как массив lis посчитан, нужно посчитать $f_{k,r}$ — максимальное число элементов, из которого могут состоять возрастающие подпоследовательности длины не менее k на префиксе исходного массива длины r. Пересчет следующим образом: фиксируем k, затем постепенно увеличиваем r. Для фиксированных k и r перебираем j — если на отрезке с $k+1$ до j существует возрастающая подпоследовательность длины не менее k, то мы можем обновить значение $f_{k,j}$ следующим образом: $f_{k,j} = max(f_{k,j}, f_{k,r-1} + lis_{r,j})$. Так не забываем про переход вида $f_{k,j} = max(f_{k,j}, f_{k,j-1})$.

Разбор задачи «Морти покупает продукты»

Давайте представим все продукты, как многочлен P, степень которого равна максимальной стоимости продукта. Коэффициентом, стоящим при i-й степени будет cnt_i, где cnt_i — количество продуктов, стоимость которых равна i. Теперь давайте поймем, что же представляет из себя покупка. Если $k = 1$, то P_i — количество способов купить один товар со стоимостью i. Если $k = 2$, $P2 = P^2$ то $P2_i$ — количество способов купить два товара, чтобы их суммарная стоимость была i и т.д. Нам нужно вычислить Pk. Для того, чтобы сделать это, воспользуемся бинарным возведением в степень. За кадром остается лишь вопрос о перемножении многочленов. Для того, чтобы быстро перемножать два многочлена, нужно написать быстрое преобразование Фурье с целыми числами (обратите внимание на модуль), так как очень вероятно, что быстрое преобразование Фурье, использующее вещественные числа превысит лимит времени исполнения на ограничениях в данной задаче.

После того, как многочлен Pk вычислен, остается предпосчитать префиксные суммы sum_i, где $sum_i = sum_{i-1} + Pk_i$. Теперь ответом на запрос lr является разность $sum_r - sum_{l-1}$, взятая по модулю.

Разбор задачи «Портальная пушка»

Переберем индекс i массива A, тем самым зафиксировав a_i. Для фиксированного i нам нужно посчитать $\sum_j (i-j) \cdot |a_i - b_j|$. Теперь раскроем $|a_i - b_j|$, рассмотрев два случая:

- $a_i \geqslant b_j$. Тогда $\sum_j (i-j) \cdot |a_i - b_j| = \sum_j (i \cdot a_i - i \cdot b_j - j \cdot a_i + j \cdot b_j)$;
- $a_i < b_j$. Тогда $\sum_j (i-j) \cdot |a_i - b_j| = \sum_j (i \cdot b_j - i \cdot a_i - j \cdot b_j + j \cdot a_i)$.

Выражения в обоих случаях одинаковы с точностью до знака, поэтому мы рассмотрим подсчет только первого из них. $\sum_j (i \cdot a_i - i \cdot b_j - j \cdot a_i + j \cdot b_j) = \sum_j i \cdot a_i - \sum_j i \cdot b_j - \sum_j j \cdot a_i + \sum_j j \cdot b_j$. Отсортируем массив пар (b_i, i) по возрастанию и двоичным поиском найдем минимальный индекс min_j, такой, что $a_i \geqslant b_{min_j}$.

- При фиксированном i, $i \cdot a_i$ — константа для всех j, поэтому $\sum_j i \cdot a_i = i \cdot a_i \cdot (|b| - min_j + 1)$;
- $\sum_j i \cdot b_j = i \cdot \sum_{j=min_j..|b|} b_j$;
- $\sum_j j \cdot a_i = a_i \cdot \sum_{j=min_j..|b|} b_j.index$, где $b_j.index$ — индекс b_j в начальном неотсортированном массиве (то есть второе число в паре);

Заметим, что чтобы посчитать сумму $\sum_{j=min_j..|b|} b_j$, достаточно посчитать префиксные суммы $pref_i = b_1 + b_2 + ... + b_i$ на отсортированном массиве. Чтобы посчитать суммы $\sum_{j=min_j..|b|} b_j.index$ и $\sum_{j=min_j..|b|} j \cdot b_j$, также достаточно посчитать другие префиксные суммы на отсортированном массиве. Таким образом, мы научились для фиксированного i за $O(\log |b|)$ вычислять сумму $\sum_j (i-j) \cdot |a_i - b_j|$. Суммарное время работы — $O(|a| \cdot \log(|b|))$.

Разбор задачи «Спасите Землю»

Рассмотрим кратчайшие отрезки от точки до одного круга и до другого. Если отрезок задевает оба круга, обновим ответ.

Теперь посмотрим, как выглядит решение. Сначала из точки нужно провести отрезок до какой-то точки на одной окружности, потом из этой точки провести отрезок до точки на второй окружности по направлению к центру этой окружности. Второй отрезок надо проводит в направлении центра второй окружности, потому что требуется из какой-то точки за минимальное расстояние дойти до окружности. Заметим, что ответом будет $|a - p| + |c - p| - r_c$, где a — начальная точка, p — точка на первой окружности, c — центр второй окружности, r_c — ее радиус. Следовательно, нужно минимизировать $|a - p| + |c - p|$, потому что r_c — константа.

Рассмотрим порядок, в котором будут посещены окружности. Найдем минимум требуемой функции. Это можно сделать с помощью тернарного поиска по углу точки p из b, где b — центр первой окружности, выбрав в качестве границ тернарного поиска углы точек a и c из b.