Московская городская командная олимпиада 2021 года
Лига A

Разбор задач

Задача «Уничтожение массива»

Автор и разработчик задачи: Иван Сафонов

Формальная постановка

- Дан массив из n целых чисел $a_1, a_2, \ldots, a_n$
- Фиксируется целое число k
- За одну операцию можно выбирать k индексов массива и вычитать из элементов на них битовый *AND* этих чисел
- Найти все возможные k для которых можно обнулить элементы массива

Решение

- Посчитаем для каждого из 30-ти битов количество раз, которое он встречается в массиве
- Заметим, что если все числа массива можно сделать равными 0, то все эти количества делятся на k (потому что при каждой операции вычитаем либо 0, либо k из количества данного бита в массиве)
- Заметим, что обратное верно - если все количества делятся на k, то массив обнулить можно

Решение

- Таким образом ответом являются все делители наибольшего общего делителя посчитанных количеств (*30*-ти чисел)
- Просто перебираем возможные k и проверяем, что они являются делителями
- Асимптотика: $O(n \log C)$

Задача «Детский садик "Тормозок"»

Автор задачи: Дарья Крохина

Разработчики: Дарья Крохина, Евгений Колодин, Егор Чунаев

Формальная постановка

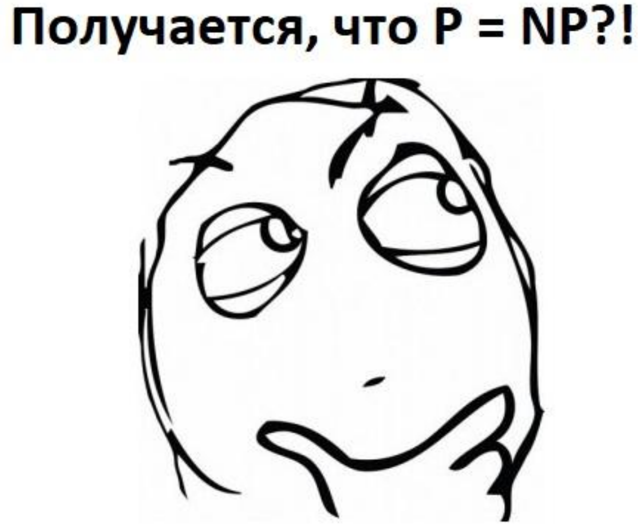
- Дано n гирек с массами m_1, m_2, …, m_n
- Надо уравновесить качели положив каждую гирю на **уникальную** позицию

Решение

- Если $n = 1$, то ответа нет
- Иначе ответ всегда есть, например, всегда подойдет следующий набор позиций:

$$-\sum_{i=2}^{n} i * m_i,\ 2 * m_1,\ 3 * m_1,\ \ldots,\ n * m_1$$

Задача «Трудная задача»



Автор и разработчик задачи: Степан Стёпкин

Формальная постановка

- Дан граф состоящий из $3n$ вершин и $3n$ рёбер
- Надо найти независимое множества размера n ~~или определить, что такого множества нет~~

Решение

- Возьмём все вершины в независимое множество
- Пока в независимом множестве есть вершина, соединённая хотя бы с двумя вершинами множества, удалим её из множества
- Теперь каждая вершина в множестве соединена не более чем с одной вершиной из множества
- Будем удалять вершины множества, соединённые с другими вершинами множества, пока такие есть

Доказательство корректности

- Пусть после первого шага алгоритма в множестве осталось F вершин, между которыми проведены M рёбер
- С одной стороны, $2 * (3n - F) \le 3n - M$, поскольку при удалении каждой вершины было удалено хотя бы два ребра
- С другой стороны $2M \le F$, поскольку каждая вершина из множества соединена не более, чем с одной вершиной из множества

Доказательство корректности (продолжение)

- Сложим два ранее полученных неравенства и получим $n \leq F - M$
- Заметим, что $F - M$ — это в точности количество вершин в независимом множестве в конце работы алгоритма, поскольку на второй фазе каждое удаление вершины происходит одновременно с удалением ровно одного ребра
- Данный алгоритм нетрудно реализовать на $O(n)$ или $O(nlogn)$

Задача «Лягушка-путешественница»

Автор и разработчик задачи: Константин Амеличев

Формальная постановка

- Дан набор из $n + 1$ состояний, каждое состояние характеризуется параметрами a_i, b_i
- Из состояния i можно перейти в любое состояние из отрезка $[i - a_i, i]$
- При переходе в состояние j происходит автоматическая смена состояния на $j + b_j$
- Назовем два вышеописанных действия *прыжком*
- Требуется за минимальное количество *прыжков* добраться из состояния n в состояние 0

Ключевые идеи

- Воспользуемся динамическим программированием dp_i — минимальное количество ходов, нужное для достижения 0, если начать в i
- $dp_i = 1 + min\ dp_{j + b[j]}, \quad i - a_i \leq j \leq i$
- Если из состояния достижим 0, то его dp равна 1. Если 0 не достижим, но достижимо состояние, из которого достижим 0, то dp равно 2, и так далее
- Таким образом, наша динамика симулирует ни что иное, как *поиск в ширину*

Пересчет динамики

- Пусть мы обработали состояния с расстоянием от 0 до d
- Возьмем состояние v $(dp_v = d)$ и все такие u, что $u + b_u = v$
- Тогда для всех j, таких что $j - a_j \le u \le j$ имеем $dp_j = d + 1$
- Сохраним в дереве отрезков для каждой позиции i значение $i - a_i$
- Тогда надо на суффиксе ДО перебрать все элементы со значением меньше u
- Перебирать элементы можно явно, так как использованные элементы второй раз смотреть не нужно
- Получаем асимптотику $O(n \log n)$

Линейное решение*

- Назовем прообразом состояния i такие j, что $j + b_j = i$
- Также заметим, что самые глубокие прообразы для каждого следующего расстояния имеют монотонно убывающие индексы
- Глубина n также встретится в этой последовательности, так что будем восстанавливать последовательность
- Для пересчета следующего прообраза нужно найти самый нижний i_1, из которого достижим текущий прообраз j
- Отсортируем глубины по величине $i - a_i$ подсчетом
- Теперь можно пересчитывать оптимальные прообразы линейным проходом

Задача «Онлайн-курс по физкультуре»

Автор задачи: Жюри
Разработчик: Тимофей Федосеев

Формальная постановка

- Дан массив из n целых чисел $a_1, a_2, \ldots, a_n$
- Приходят запросы l, r
- Требуется просуммировать минимумы на префиксах подотрезка $[l, r]$, длины которых сравнимы с 1 по модулю k

Заметим что

- Обозначим за b_i минимум на отрезке $[i - k, i - k + 1, ..., i]$
- Все b_i можно посчитать за линейное время с помощью алгоритма поиска минимума в окне
- Тогда ответом на запрос будет $a_l + b_{l + k} + min(b_{l + k}, b_{l + 2k}) + ... + min(b_{l + k}, b_{l + 2k} + ... + b_{l + t\_k}$, где $l + t_k \leq r$
- Заметим, что такая сумма независима по остатку при делении l на k, а значит наша задача свелась к такой: дан массив c, нужно уметь суммировать префиксные минимумы на отрезке

Решение

- Для этого посчитаем величину nxt_i - ближайшая справа позиция к i, такая что $c_{nxt\_i} < c_i$
- Посчитаем динамику dp_i - сумма минимумов на всех префиксах i-го суффикса. Заметим, что $dp_i = dp_{nxt\_i} + c_i * (nxt_i - i)$
- Чтобы посчитать сумму префиксных минимумов на отрезке l, r, найдем позицию минимума на отрезке, обозначим ее за p, тогда ответ на запрос равен $dp_l - dp_p + (r - p + 1) * c_p$

Конец

- Получаем решение за $O(n + q\alpha^{-1}(n))$, если воспользоваться алгоритмом Тарьяна для поиска минимума на отрезке в офлайне
- Для сдачи задачи можно было воспользоваться любой логарифмической структурой

Задача «Еще одна задача с фишками»

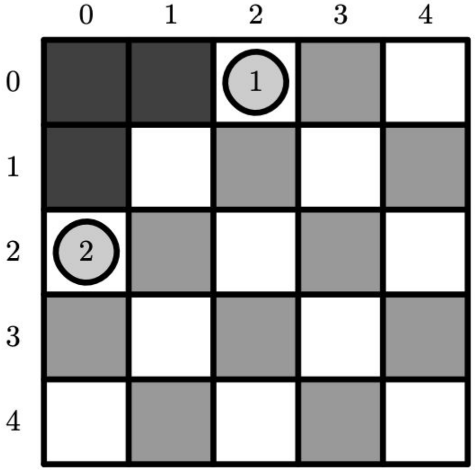
Автор и разработчик задачи: Владимир Романов

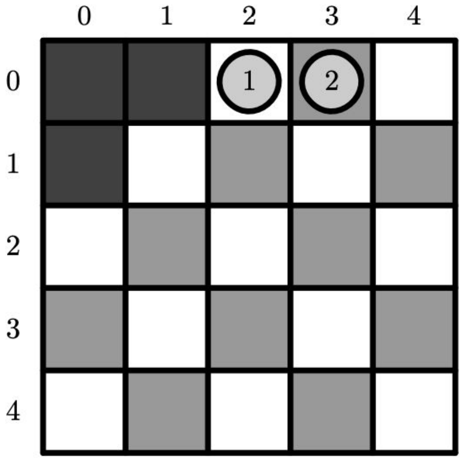
Формальная постановка

- Дано поле размером n на n с заблокированным верхним левым углом размером k
- На поле расположены две фишки
- Два игрока ходят по очереди
- За один ход нужно переместить одну из фишек вверх или влево в свободную клетку
- Проигрывает тот, кто не может сделать ход
- Требуется для t расположений фишек определить, кто победит

Заметим что

- Фишки в конце игры будут либо на одноцветных клетках, либо на разноцветных, в последнем случае они будут стоять рядом



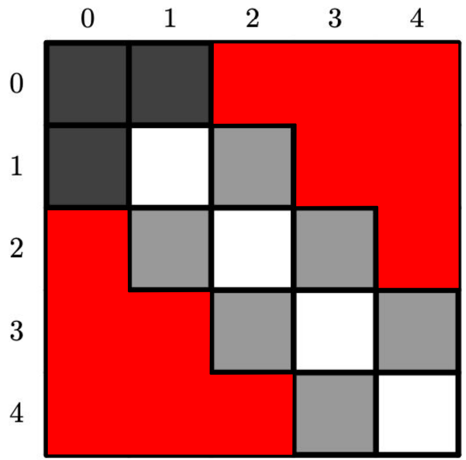


Решение

- Решим задачу, когда первым ходит тот игрок, у которого фишки стоят на клетка разного цвета
- Если первым ходит другой игрок, то просто переберем его варианты хода, сведя задачу к предыдущей

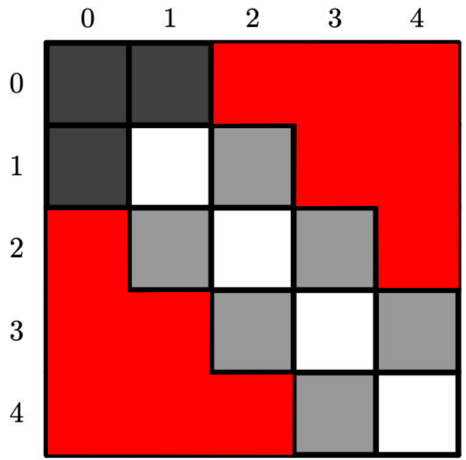
Решение

- Обозначим за диагональ i клетки, для которых $x - y$ равно i
- Назовем диагонали главными, если на них есть заблокированная клетка
- Иными словами диагональ i главная, если $|i| < k$

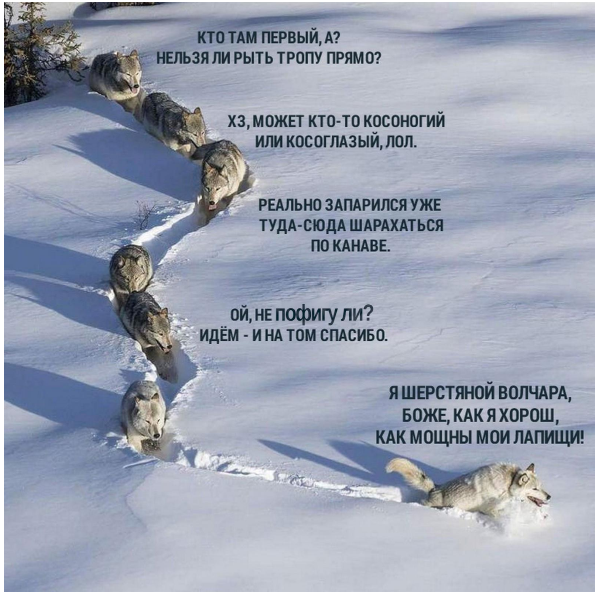


Решение

- Хотя бы одна фишка на главных диагоналях – первый побеждает
- Фишки на разных красных половинах – снова первый
- Иначе второй



Задача «Сложная гора»



Автор и разработчик задачи: Тихон Евтеев

Формальная постановка

- Дано n пар чисел (s_i , a_i) и число d
- Требуется найти максимальное число k, такое что существует последовательность $Q = q_1 \dots q_k$, такая что:
  - $1 \le q_i \le n$
  - Если $i \ne j$, то $q_i \ne q_j$
  - Для любого $i : s_{qi} \ge max(d, a_{q1,} a_{q2} \dots a_{qi-1})$

Первые шаги

- Для начала отбросим из рассмотрения все пары (s_i, a_i), такие что $s_i < d$. Они нам точно не пригодятся
- Будем называть пару индексов i, j противоречивой, если
  - $i \neq j$ и $s_j < a_i$ и $s_i < a_j$. Такие индексы не могут одновременно входить в ответ (последовательность Q)
- Заметим, что если нет противоречивой пары индексов, то $n = k$
- В качестве Q подойдет сортировка $(1, 2 \ldots n)$ по возрастанию пар $\{min(s_i, a_i), max(s_i, a_i)\}$

Простая подзадача

- Научимся решать частный случай задачи:
  - Пусть для каждого $i : s_i < a_i$, В таком случае можно воспользоваться следующим жадным решением:
  - Пусть $D = d$, найдем среди пар (s_i, a_i) такие, что $D \le s_i$, а среди таких — пару с наименьшим a_i — она и будет следующей в нашем порядке
  - Заменим D на a_i, увеличим k (ответ) на 1 и повторим алгоритм
  - Если пары с $D \le s_i$ не существует, завершим алгоритм
- Корректность такого алгоритма доказывается по индукции
- Чтобы эффективно реализовать это решение, отсортируем все пары (s_i, a_i) по возрастанию a_i
  - Пройдем по индексам i от 1 до n
  - Если $D \le s_i$, то добавим к ответу 1 и заменим D на a_i

Вернёмся к полной задаче

- Рассмотрим такую пару индексов i, j, что
  - $s_i < a_j \leq s_j < a_i$
- Такая пара индексов противоречива, более того, если в последовательности $Q : q_x = i$, то при замене q_x на j последовательность всё ещё будет удовлетворять необходимым условиям и являться ответом
- Тогда для всех таких пар можно удалить из рассматриваемого множества пару с индексом i и ответ не ухудшится

Реализация

- Чтобы эффективно удалить все такие пары (s_i, a_i) воспользуемся методом двух указателей:
- Вынесем все такие пары, что $a_i \le s_i$, в массив b. Пусть оставшиеся пары находятся в массиве c. Отсортируем массив b по возрастанию a_i и массив c по возрастанию s_i
- Пусть M - упорядоченное множество, которое хранит пары (s_i, a_i) по убыванию a_i
- Заведём указатель $j = 0$
- Пройдем по элементам массива b с индексом i.
    - Пока $c_j.s \le b_i.a$ будем добавлять c_j в множество M
    - Теперь пока $M_1.a > b_i.s$ будем удалять первый элемент M
- Среди элементов массива b, множества M и оставшихся элементов в массиве c не осталось искомых пар

Сейчас всё встанет на свои места

- Заметим, что среди оставшихся пар (s_i, a_i) любая пара индексов i, j, такая что $a_i \le s_i$ или $a_j \le s_j$ не противоречива
- Давайте решим задачу для пар с $a_i < s_i$, используя жадный алгоритм, и добавим к полученному ответу все пары с $s_i \le a_i$
- В получившемся множестве нет противоречивой пары индексов, а также по очевидным причинам оно максимального размера
- Следовательно размер полученного множества и есть ответ на оригинальную задачу
- Итоговая асимптотика $O(n \log n)$

Задача «Полёт над озером»

geometry, the process:

- geome***try***
- geome***cry***
- geome***why***
- geome***bye***

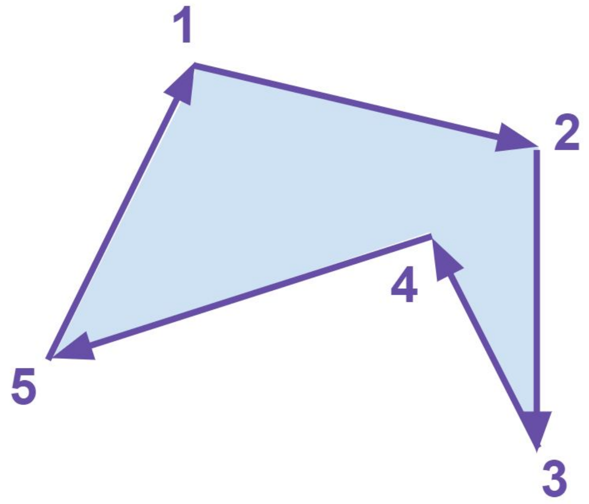
Автор задачи: Михаил Пядеркин
Разработчик: Алеся Иванова

Формальная постановка

- Дан невыпуклый многоугольник и набор горизонтальных прямых
- Нужно вычислить суммарную площадь частей многоугольника под каждой прямой

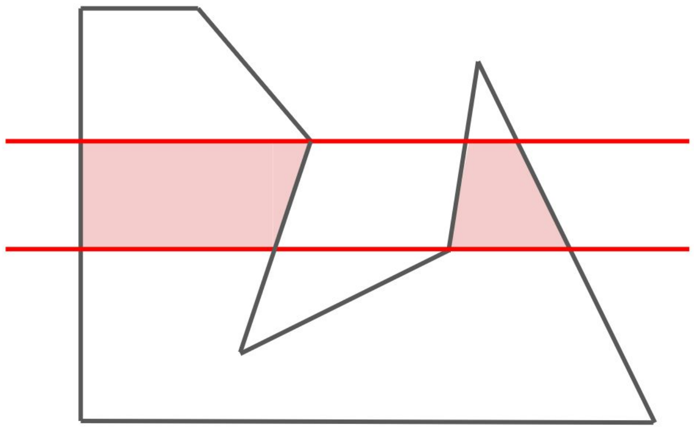
Решение

- Перенумеруем вершины так, чтобы многоугольник был записан в порядке обхода по часовой стрелке
- Порядок обхода можно определить, посчитав ориентированную площадь многоугольника
- Теперь для каждой стороны внутренняя часть многоугольника находится справа, если встать в вершину с меньшим номером и смотреть по направлению вершины с большим номером



Решение

- Мысленно проведём горизонтальные прямые через каждую вершину многоугольника
- Эти прямые делят плоскость на полосы между соседними прямыми, при этом пересечение каждой полосы и исходного многоугольника представляет из себя несколько трапеций



Решение

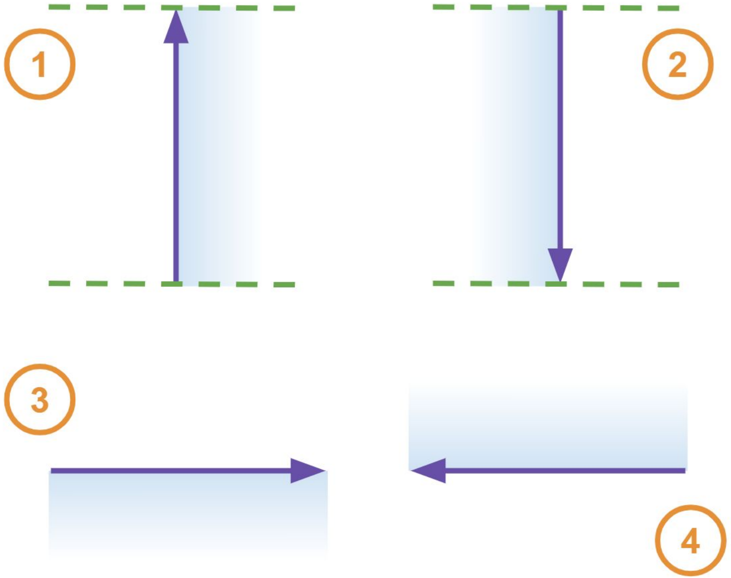
- Пусть $f(y)$ - суммарная длина отрезков горизонтальной прямой на высоте y, лежащих внутри многоугольника
- Пусть соседние прямые находятся на высоте y_1 и y_2 $(y_1 < y_2)$, тогда суммарная площадь трапеций внутри соответствующей полосы может быть посчитана как $(f(y_1) + f(y_2)) * (y_2 - y_1) / 2$
- Между двумя соседними прямыми $f(y)$ изменяется линейно. Для каждой полосы найдём коэффициенты k и b такие, что для этой полосы выполняется $f(y) = k * y + b$

Решение

- Будем решать задачу при помощи сканирующей прямой
- При прохождении сканирующей прямой через вершину многоугольника функция будет изменяться в зависимости от наклона сторон, соответствующих этой вершине
- Для каждой стороны определим, как она меняет коэффициенты линейной функции при прохождении сканирующей прямой через её концы

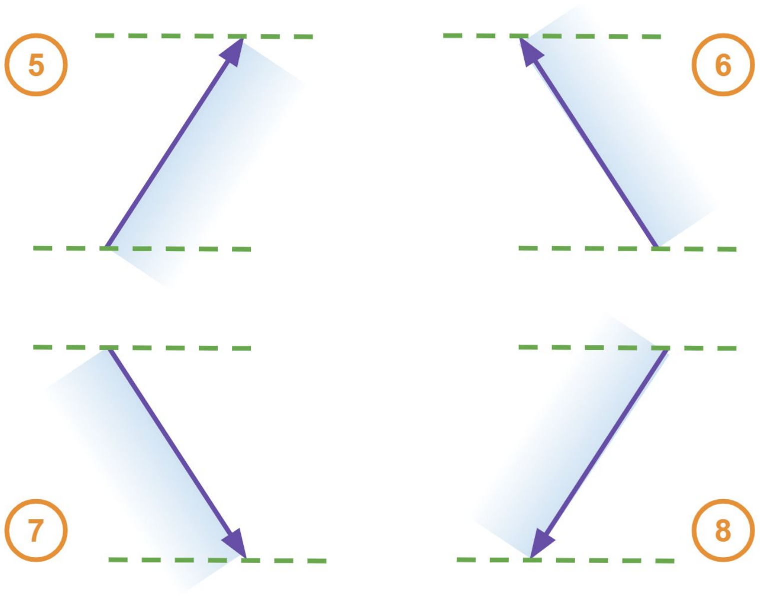
Решение

- Если сторона вертикальная, то она никак не изменяет функцию.
- Если сторона горизонтальная, то она изменяет функцию на константу. Если внутренняя часть многоугольника лежит сверху, то она прибавляет к b свою длину, иначе вычитает



Решение

- В остальных случаях у функции изменяется коэффициент k
- В конца отрезка он изменяется на одну одинаковые по модулю, но противоположные по знаку величины.
- При прохождении сканирующей прямой через нижнюю вершину отрезка в случаях 5 и 7 k уменьшается на $|(x_2 - x_1) / (y_2 - y_1)|$, а в случаях 6 и 8 увеличивается на такую же величину



Решение

- Задача свелась к следующей: на каких-то y-координатах есть события, которые показывают, как надо изменить функцию при прохождении сканирующей прямой через эту координату
- При этом каждый отрезок создаёт 0, 1 или 2 события, в зависимости от своего типа
- Зная функцию на каждом полосе, можно посчитать суммарную площадь пересечения многоугольника и этой полосы
- Также можно посчитать для каждого y, являющегося y-координатой какой-то вершины, площадь частей многоугольника под прямой на высоте y

Конец

- Будем хранить все такие прямые, площади под ними и линейные функции на полосах
- Чтобы узнать площадь под произвольной горизонтальной прямой на высоте y', надо бинарным поиском найти ближайшую снизу к ней прямую (пусть её высота y), вычислить $f(y')$ с помощью соответствующей линейной функции
- Ответом будет являться сумма площади под прямой на высоте y и $(f(y) + f(y')) * (y' - y) / 2$
- Асимптотика решения $O((n + q) * (\log n + \log C))$, где $\log C$ возникает из-за операции деления по модулю

Задача «Оптимальная вставка»

Автор и разработчик задачи: Иван Сафонов

Формальная постановка

- Даны массивы из целых чисел a_1, a_2, …, a_n и b_1, b_2, …, b_m
- Нужно вставить элементы b_i в произвольные места массива a в любом порядке
- Какое минимальное количество инверсий может быть в полученном массиве длины $n + m$?

Решение

- Отсортируем массив b
- Пусть p_i это индекс массива a, перед которым вставляется элемент b_i ($p_i = n + 1$, если в конец)
- Заметим, что $p_1 \le p_2 \le \ldots \le p_m$
- Если это не так, то можно поменять местами вставленные элементы b_i, из-за чего количество инверсий только уменьшится

Решение

- Будем искать p_i с помощью “Разделяй и властвуй”
- Напишем рекурсивную функцию $solve(l_i, r_i, l_p, r_p)$
- Она будет искать все p_i для $i \in [l_i, r_i)$, если известно, что $p_i \in [l_p, r_p]$
- Чтобы найти все p_i надо вызвать $solve(1, n + 1, 1, m + 1)$

Реализация функции *solve*

- Если $l_i \geq r_i$, то ничего не нужно делать
- Пусть $mid = (l_i + r_i) / 2$
- Найдем p_{mid}. Заметим, что b_{mid} приносит количество инверсий $\#(a_i > b_{mid}$ при $i < p_{mid}) + \#(a_i < b_{mid}$ при $i \geq p_{mid})$
- Это отличается на константу от $\#(a_i > b_{mid}$ при $i \in [l_p, p_{mid})) + \#(a_i < b_{mid}$ при $i \in [p_{mid}, r_p))$
- Такое оптимальное p_{mid} можем найти линейным проходом за $O(r_p - l_p)$

Реализация функции *solve*

- p_{mid} нашли
- Вызовем $solve(l_i, mid, l_p, p_{mid})$ и $solve(mid + 1, r_i, p_{mid}, r_p)$, чтобы найти остальные p_i
- Заметим, что суммарное время работы *solve* равно $O((n + m) \log m)$, потому что у нас $\log m$ уровней рекурсии, на каждом мы суммарно решаем за $O(n + m)$

Конец

- Используя найденные p_i получим массив после вставки
- Найдем количество инверсий в нем
- Время работы решения: $O((n + m) \log (n + m))$

- Существуют также другие решения с такой же асимптотикой, использующие дерево отрезков

Задача «Две сортировки»

Автор задачи: Дмитрий Саютин
Разработчики: Михаил Ипатов, Дмитрий Саютин

Формальная постановка

- Целые числа от 1 до n отсортировали лексикографически, сравнивая их как в строки в десятичной системе счисления, назовем это перестановкой $a_1, a_2, \ldots, a_n$
- Дан модуль $p = 998244353$
- Требуется найти $sum_{i=1..n} ((i - a_i) \bmod p)$

О суммах

- Суммировать $sum_{i=1..n}((i - a_i) \bmod p)$ не так удобно, сделаем замену
- Пусть b это обратная перестановка к a, т.е. $a[b[i]] = b[a[i]] = i$
- $b[i]$ это позиция числа i в сортировке лексикографически.
- Тогда исходная сумма равна $sum_{i=1..n}((b[i] - a[b[i]]) \bmod p)$, из-за того что слагаемые такие же
- Из соображений $a[b[i]] = i$, получается, что искомая сумма равна $sum_{i=1..n}((b[i] - i) \bmod p)$

К вопросу о позиции в списке

- $sum_{i=1..n}$ $((b[i] - i) \bmod p)$
- $b[i]$ это позиция числа i в сортировке лексикографически.
- Это число чисел, которые лексикографически меньше нашего
- Такие сравнения определяются первой отличающейся цифрой:

$$12\mathbf{0}4 <_{lex} 12\mathbf{2}$$

- **Вопрос:** какие числа меньше какого-то конкретного числа *122*? Как их посчитать?

К вопросу о позиции в списке

- **Вопрос:** какие числа меньше какого-то конкретного числа *122*? Как их посчитать?
- **Ответ:** перебрать длину числа, которое нас меньше, перебрать позицию первой отличающейся цифры, саму первую отличающуюся цифру

*1**2**2 > 1**1**???*

- А дальше просто посчитать число чисел подходящие под такое описание
- Обратите внимание: если “*11???*” равно по длине *N*, где доступны числа от *1...N*, то формула будет чуть-чуть особой, пример “*11???*”, *N=11456*

Back to business

- Сгруппируем вместе слагаемые, которые удобно суммировать вместе
- Переберём:
    - Длину i
    - Общий префикс с N, а значит и позицию первой различной цифры (числа, которые образуют префикс с N надо разобрать отдельно)
    - первую цифру после

Почти Конец

- Вспомним формулу $sum_{i=1..n} ((b[i] - i) \bmod p)$
- Обозначим оставшиеся цифры переменными $i = 123xyz$
- Несложно заметить, что i равно линейной комбинации x, y, z, и что не менее важно, $b[i]$ тоже
- $i = ax + by + cz + d$
- $b[i] = a'x + b'y + c'z + d' \Rightarrow b[i] - i = a''x + b''y + c''z + d''$

Конец

- Осталось просуммировать выражение вида $(a''x + b''y + c''z + d'') \bmod p$ по всем $x, y, z \leftarrow \{0, 1, 2, \ldots, 9\}$
- Можно сделать meet-in-the-middle: переберём все варианты на первой половине переменных, отсортируем остатки и сохраним в массив. Аналогично для второй половины
- Осталось вычислить $sum_{i=1..|a|}\ sum_{j=1..|b|}\ [(a_i + b_j) \bmod p]$, что тоже несложно сделать
- Итоговая асимптотика $O(10^{[len/2]}\ poly(len) = 10^{[log10(n)/2]}\ poly(log(n)) = sqrt(n)\ poly(log(n)))$