Разбор задачи «Лягушка-путешественница»

Автор и разработчик задачи: Константин Амеличев

Назовем прыжком комбинацию переходов $i \to i - a_i \dashrightarrow i - a_i + b_{i-a_i}$.

Введем динамическое программирование dp_i — минимальное количество ходов, нужное для достижения 0, если начать в i.

Мы знаем, что $dp_i = 1 + \min(dp_j + b_j)$, для $i - a_i \leqslant j \leqslant i$. Эту динамику можно считать в обратную сторону. А именно, 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, такой пересчет симулирует ни что иное, как поиск в ширину.

Рассмотрим пересчет bfs на стадии, когда все расстояния от 0 до d посчитаны. Возьмем состояние v ($dp_v = d$) и все такие u, что $u + b_u = v$. Тогда для всех j, таких что $j - a_j \leqslant u \leqslant j$ имеем $dp_j = d + 1$. Сохраним в дереве отрезков на минимум для каждой позиции i значение $i - a_i$. Тогда надо на суффиксе массива перебрать все элементы со значением меньше u.

Перебирать элементы можно явно, так как использованные элементы второй раз смотреть не нужно — их можно заменить на нейтральный элемент.

Итоговая асимптотика $O(n \log n)$

Бонус. Попробуйте решить задачу за линейное время :)

Разбор задачи «Уничтожение массива»

Автор и разработчик задачи: Иван Сафонов

Заметим, что при одном уничтожении для каждого бита i ($0 \leqslant i < 30$) мы либо заменяем все k единичных i-х битов на нулевые биты, либо ничего не происходит. Таким образом, из количества i-х единичных битов в массиве вычитается либо k, либо 0. В итоге все количества станут равными 0. Значит, чтобы массив можно было уничтожить, все количества изначально должны делиться на k. Докажем, что это критерий.

Действительно, если все количества делятся на k, то будем делать операции пока можем. Если еще не все элементы массива нулевые, то есть хотя бы один бит i, такой что есть хотя бы k чисел массива, i-й бит которых единичный. Тогда выберем эти числа для очередной операции уничтожения, тем самым уменьшив сумму чисел в массиве. Так мы сделаем все элементы массива нулями.

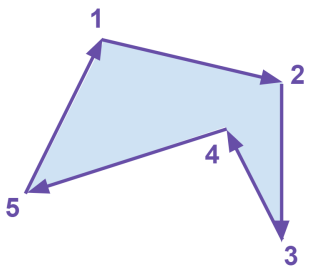
В итоге алгоритм такой: найдем для каждого бита i ($0 \leqslant i < 30$) количество чисел в массиве с единичным i-м битом. Найдем все общие делители k ($1 \leqslant k \leqslant n$) всех полученных чисел.

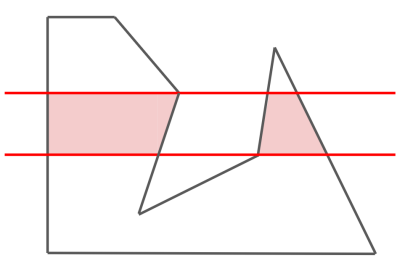
Решение работает за $O(n \log C)$, где $C = 10^9$ — верхнее ограничение на все числа в массиве.

Разбор задачи «Полёт над озером»

Автор задачи: Михаил Пядеркин, разработчик: Алеся Иванова

Будем считать, что многоугольник задан в порядке обхода по часовой стрелке. Порядок обхода можно определить, посчитав ориентированную площадь многоугольника. Если порядок обхода против часовой стрелки, то перенумеруем вершины. Теперь для каждой стороны внутренняя часть многоугольника находится справа, если встать в вершину с меньшим номером и смотреть по направлению вершины с большим номером.



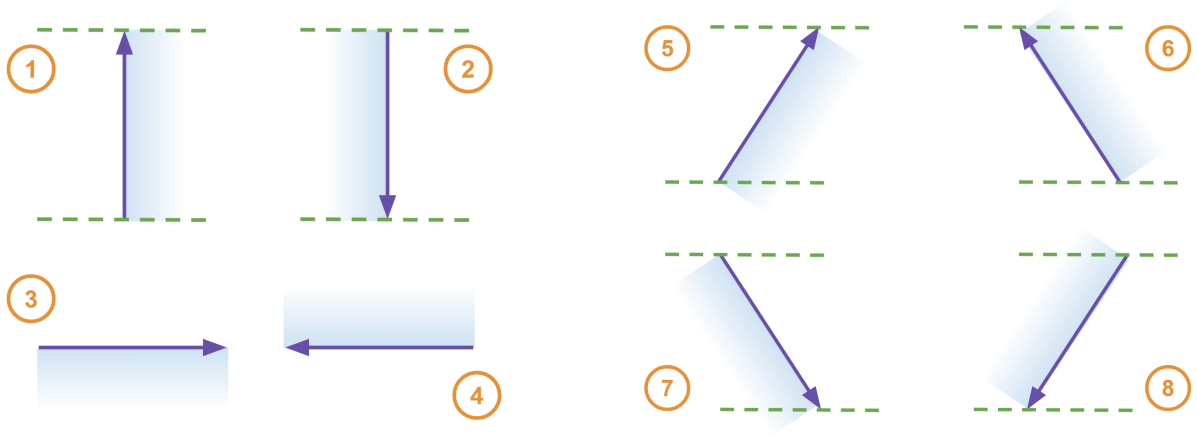


Мысленно проведём горизонтальные прямые через каждую вершину многоугольника. Эти прямые делят плоскость на полосы между соседними прямыми, при этом пересечение каждой полосы и исходного многоугольника представляет собой несколько трапеций.

Пусть $f(y)$ — суммарная длина отрезков горизонтальной прямой на высоте y, лежащих внутри многоугольника (пока будем считать, что в многоугольнике нет горизонтальных сторон). 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 $\frac{f(y_1)+f(y_2)}{2} \cdot (y_2 - y_1)$.

Между двумя соседними прямыми $f(y)$ изменяется линейно. Для каждой полосы найдём коэффициенты k и b такие, что для этой полосы выполняется $f(y) = k \cdot y + b$.

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. Рассмотрим все случаи:



- 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.
- 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 $|\frac{x_2-x_1}{y_2-y_1}|$, а в случаях 6 и 8 увеличивается на такую же величину.

То есть 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. 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 $\frac{f(y)+f(y')}{2} \cdot (y'-y)$. Асимптотика решения $O((n+q) \cdot (\log n + \log C))$, где $\log C$ возникает из-за операции деления по модулю.

Разбор задачи «Оптимальная вставка»

Автор и разработчик задачи: Иван Сафонов

Отсортируем массив b.

Обозначим за p_i индекс массива a, перед которым нужно вставить b_i. Если элемент вставляется после всех элементов массива a, то $p_i = n + 1$.

Заметим, что вставленные элементы массива b должны идти в порядке сортировки в оптимальном ответе. Если это не так и есть $p_i > p_j$ для $i < j$, то поменяем в ответе b_i и b_j местами. Количество инверсий уменьшится.

Значит $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_m$. Нам нужно найти их значения, тогда мы сможем восстановить массив после вставки.

Воспользуемся техникой «Разделяй и властвуй». Напишем рекурсивную функцию $solve(l_i, r_i, l_p, r_p)$, которая находит p_i для всех $l_i \leqslant i < r_i$ и при этом известно, что $p_i \in [l_p, r_p]$ для всех таких i. Для того, чтобы найти все значения массива p нам нужно запустить функцию $solve(1, m+1, 1, n+1)$.

Реализация функции $solve(l_i, r_i, l_p, r_p)$ будет следующей:

- Если $l_i \geqslant r_i$, то ничего искать не нужно.
- Пусть $mid = \lfloor \frac{l_i + r_i}{2} \rfloor$. Тогда найдем p_{mid}. Заметим, что количество инверсий, которое принесет то, что мы поставим b_{mid} перед p_{mid} будет равно (количество $a_i > b_{mid}$ для $i < p_{mid}$) + (количество $a_i < b_{mid}$ для $i \geqslant p_{mid}$). Эта сумма отличается на константу от суммы: (количество $a_i > b_{mid}$ для $l_p \leqslant i < p_{mid}$) + (количество $a_i < b_{mid}$ для $p_{mid} \leqslant i < r_p$). Для этой суммы мы можем легко найти минимальное оптимальное p_{mid} за время $O(r_p - l_p)$.
- Запустим два рекурсивных вызова $solve(l_i, mid-1, l_p, p_{mid})$, $solve(mid+1, r_i, p_{mid}, r_p)$, которые найдут оставшиеся значения p.

Время работы этой процедуры будет $O((n+m)\log m)$, потому что всего будет $O(\log m)$ уровней рекурсии, на каждом из которых мы сделаем суммарно $O(n+m)$ операций.

В конце, когда мы нашли все p_i, получим оптимальный массив и найдем количество инверсий в нем.

Итоговая асимптотика: $O((n+m)\log(n+m))$.

Также существуют также другие правильные решения с такой же асимптотикой, использующие дерево отрезков.

Разбор задачи «Онлайн-курс по физкультуре»

Автор задачи: Жюри, разработчик: Тимофей Федосеев

Заметим, что нужно купить сертификат в первый день, затем нужно купить минимальный сертификат среди первых k+1-го дня, затем минимальный среди первых $2k$+1-го дня, ..., минимальный среди первых $tk+1$-го дня. 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 $a_l + b_{l+k} + \min(b_{l+k}, b_{l+2k}) + ... + \min(b_{l+k}, b_{l+2k} + ... + b_{l+tk})$, где $l + tk \leqslant r$.

Заметим, что такая сумма независима по отстатку при делении l на k, а значит наша задача свелась к такой: дан массив c, нужно уметь суммировать префиксные минимумы на отрезке. 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 $dp_i = dp_{nxt_i} + c_i \cdot (nxt_i - i)$. Чтобы посчитать сумму префиксных минимумов на отрезке $[l, r]$, найдем позицию минимума на отрезке, обозначим ее за p, тогда ответ на запрос равен $dp_l - dp_p + (r - p + 1) \cdot c_p$.

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

Разбор задачи «Сложная гора»

Автор и разработчик задачи: Тихон Евтеев

Для начала отбросим все i, такие что $s_i < d$.

Вместо альпинистов будем рассматривать пары (s_i, a_i), а также говорить, что множество пар корректно, если существует перестановка $p_1 \ldots p_n$, такая что для любого $i : \max(d, a_{p_1}, \ldots a_{p_{i-1}}) \leqslant s_{p_i}$, что соответствует тому, что существует порядок, в котором все альпинисты смогут забраться.

Будем называть пару индексов i, j противоречивой, если $i \neq j$ и $s_j < a_i$ и $s_i < a_j$ — это значит, что i-й альпинист не может забраться после j-го и наоборот. Заметим, что если в множестве нет противоречивой пары индексов, то оно корректно. В качестве порядка для пар (s_i, a_i) подойдет сортировка по возрастанию пар $min(s_i, a_i), \max(s_i, a_i)$, после этого окажется, что i-й в такой сортировке может забраться после $i-1$-го или пара $(i-1, i)$ противоречива.

Теперь научимся решать частный случай задачи, а именно, когда 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 $D \leqslant s_i$, а среди таких — пару с наименьшим a_i — она и будет следующей в нашем порядке. Заменим D на a_i, увеличим ответ на 1 и повторим алгоритм. Если пары с $D \leqslant s_i$ 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 $D \leqslant s_i$, то добавим к ответу 1 и заменим D на a_i.

Вернёмся к полной задаче:

Рассмотрим такую пару индексов i, j, что $s_i < a_j \leqslant s_j < a_i$.

Такая пара индексов противоречива, причём если в оптимальном ответе присутствует i, то её можно заменить на j и последовательность не сломается.

$s_i < s_j \Rightarrow$ для любого занчения D подходящего для i оно подходит для j.

$a_j < a_i \Rightarrow$ для любого $D : max(D, a_j) \leqslant max(D, a_i)$

Следовательно для любой такой пары i, j из множества альпинистов можно исключить i-го и ответ не ухудшится.

Чтобы эффективно удалить все такие пары (s_i, a_i) воспользуемся методом двух указателей:

Вынесем все такие пары, что $a_i \leqslant s_i$, в массив b. Пусть оставшиеся пары находятся в массиве c. Отсортируем массив b по возрастанию a_i и массив c по возрастанию s_i. Заведем упорядоченное множество M, которое сможет хранить пары (s_i, a_i) по убыванию a_i. Заведём указатель $j = 0$.

Пройдем по элементам массива b с индексом i.

Для данного элемента:

Пока $c_j.s < b_i.a$ будем добавлять c_j в множество M.

Теперь пока $b_i.s < M_1.a$ будем удалять первый элемент M.

Среди элементов массива b, множества M и оставшихся элементов в массиве c не осталось искомых пар.

Заметим, что среди оставшихся пар (s_i, a_i) любая пара индексов i, j, такая что $a_i \leqslant s_i$ или $a_j \leqslant s_j$ не противоречива.

Теперь если найти максимальное корректное подмножество из пар (s_i, a_i), таких что $s_i < a_i$ и объеденить его с множеством пар (s_i, a_i), таких что $a_i \leqslant s_i$ мы получим корректное множество, причём по очевидным причинам — максимальное по размеру. Следовательно, мы получим ответ на задачу.

Разбор задачи «Две сортировки»

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

Пусть b это обратная перестановка к a, т.е. $a_{b_i} = b_{a_i} = i$, то есть b_i это позиция числа i в сортировке лексикографически. Перепишем желаемую сумму используя замену $i \to b_i$: $\sum_{i=1\ldots n}((i - a_i) \bmod p) = \sum_{i=1\ldots n}((b_i - a_{b_i}) \bmod p) = \sum_{i=1\ldots n}((b_i - i) \bmod p)$.

Поймём как считать величину b_i. Заметим, что b_i равно 1 плюс количество чисел x $(1 \leqslant x \leqslant n)$, что $x <_{lex} i$. Такие числа делятся на две категории: собственные префиксы i (их количество зависит только от длины i) и числа x имеющие общий префикс с i какой-то длины t, и меньшую цифру c в $(t + 1)$-й позиции.

Если зафиксировать величины t, c, длину x, то у нас останется "маска" следующего вида: "123???", и нужно посчитать сколько чисел под неё подходят. Почти всегда это зависит только от количества "?", но к сожалению есть особые случаи связанные с n: например рассмотрите $n = 123456$ для примера выше.

Поэтому в требуемой сумме сгруппируем слагаемые по следующим признакам i (переберём значения этих признаков).

- Длина i,
- Позиция первой цифры отличающей i от n (случаи где i является строгим префиксом n разберём отдельно),

- Значение этой цифры.

Т.е. мы знаем описание i следующего вида: $i = c_1, c_2, \ldots, c_k x_1 \ldots, x_l$, где c_j фиксированные цифры, а x_1,...,x_l произвольные переменные в $[0, 9]$. Заметим, что b_i и i являются линейными комбинациями относительно переменных x_j и единицы, так что $b_i - i$ тоже является линейной комбинацией относительно этих величин. Осталось научиться считать $b_i - i$ по модулю p.

Чтобы просуммировать по всем возможным значениям $x_1, \ldots, x_l$ воспользуемся методом meet-in-the-middle: отдельно переберём все значения для первой половины x_j, и для второй, и совместим одно с другим.

Если $n \leqslant 10^L$, то решение работает за $\mathcal{O}(10^{\frac{L}{2}} poly(L))$, или $\mathcal{O}(\sqrt{n}\, poly(log(n)))$.

Разбор задачи «Еще одна игра с фишками»

Автор и разработчик задачи: Владимир Романов

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

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

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 $|i| < k$.

Второй выигрывает только тогда, когда в конце фишка будет блокировать другую. Тогда сделаем предположение, что второй побеждает, если фишки находятся по одну сторону от главных диагоналей. Действительно, как бы ни походил первый, второй всегда сможет сделать ход, чтобы вернуть инвариант.

Следующее предположение, первый побеждает в остальных случаях. Представим, когда второй побеждает (делает так, чтобы фишки были на одной половине) за один ход. Утверждается, что если первый походит фишкой с минимальной по модулю разницей координат в сторону противоположную диагонали другой фишки, то нельзя получить представленные ранее позиции.

Разбор задачи «Трудная задача»

Автор и разработчик задачи: Степан Стёпкин

Воспользуемся следующим алгоритмом. Изначально добавим все вершины в независимое множество. На первом шаге будем удалять вершины из независимого множества пока они соединены хотя бы с двумя вершинами множества. На втором шаге будем удалять все вершины, которые соединены хотя бы с одной вершиной независимого множества.

Очевидно, что множество, полученное в конце работы алгоритма, действительно является независимым. Покажем, что оно имеет размер хотя бы n.

Предположим, что после первого шага работы алгоритма в множестве осталось F вершин, соединённых M рёбрами. Во-первых, $2(3n - F) \leqslant 3n - M$, поскольку при удалении каждой вершины было удалено хотя бы два ребра. Во-вторых, $2M \leqslant F$, поскольку каждая вершина из множества соединена не более, чем с одной вершиной из множества.

Сложим эти неравенства; после упрощения получим $F - M \geqslant n$. Заметим, что $F - M$ — это в точности размер независимого множества в конце, поскольку при удалении каждой вершины на втором шаге удаляется ровно одно ребро.

Данный алгоритм нетрудно реализовать так, чтобы он имел сложность $O(n)$ или $O(n \log n)$.

Разбор задачи «Детский садик «Тормозок»»

Автор задачи: Дарья Крохина, разработчики: Дарья Крохина, Евгений Колодин, Егор Чунаев

Если $n = 1$, то ответа не существует, в любом другом случае ответ можно построить.

Например, всегда подойдет следующий набор позиций:

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