Задача A. Аборигены

Автор и разработчик: Андрей Станкевич

Заметим, что отдать следует наименее ценные сокровища. Отсортируем их по ценности и отдадим первые $\lceil n/2 \rceil$ сокровищ. Для вычисления количества сокровищ k, которые следует отдать можно использовать формулу `k = (n + 1) / 2`, если «/» означает целочисленное деление.

Приведем пример кода на языках C++:

```
sort(a.begin(), a.end());
int sum = 0;
for (int i = (n + 1) / 2; i < n; ++i){
    sum += a[i];
}
cout << sum << '\n';
```

и Python:

```
a.sort()
print(sum(a[(n + 1) // 2:]))
```

Задача B. Сбалансированная иллюминация

Предложение и разработка задачи: Андрей Станкевич

Объект, описанный в условии называется сбалансированным кодом Грея, он был описан в работах Адама, Бакоса, Кона, Поллака, Робинсона, Тутилла и других в 1950-е годы.

Приведем конструктивный алгоритм, который строит искомый код.

Коды Грея для $n = 1$ и $n = 2$ подходят в любом случае. Коды Грея для $n = 3$ и $n = 4$ приведены в качестве примера. Научимся переходить от кода для n к коду для $n + 2$.

Для начала заметим, что код однозначно характеризуется последовательностью из 2^n чисел от 1 до n — номерами битов, которые меняются при переходе к очередному элементу. Например, для кода, приведенного в условии для $n = 3$ искомая последовательность будет $[2, 3, 1, 3, 2, 3, 1, 3]$. Число c_i — количество вхождений числа i в эту последовательность. Будем дальше рассматривать эти последовательности.

Также отметим, что циклический сдвиг последовательности также является корректной последовательностью для кода Грея с теми же значениями c_i. Кроме того, можно применить произвольную подстановку π, заменив вхождение i в последовательность на $\pi[i]$.

Пусть есть код для n, выделим в нем l позиций, включая последнюю, где l — нечетное число. Обозначим фрагмент до i-й выделенной позиции как α_i, а элемент на i-й выделенной позиции как j_i. Будем обозначать как β^R развернутую последовательность β.

Тогда если «$\alpha_1 j_1 \alpha_2 j_2 \dots \alpha_l j_l$» — код Грея для n, то код «$\alpha_1 (n+2) \alpha_1^R (n+1) \alpha_1 j_1 \alpha_2 (n+1) \alpha_2^R (n+2) \alpha_2 j_2 \alpha_3 (n+2) \alpha_3^R (n+1) \alpha_3 j_3 \dots j_l \alpha_l (n+2) \alpha_l^R (n+1) \alpha_l (n+2) j_{l-1} \alpha_{l-1}^R j_{l-2} \alpha_{l-2}^R \dots j_1 \alpha_1^R (n+1)$» является кодом Грея для $n + 2$.

Осталось выбрать, какие позиции выделить, чтобы из сбалансированного кода для n получать сбалансированный код для $n + 2$. Утверждается, что это достигается следующим алгоритмом. Применим подстановку, чтобы значения c_i шли по возрастанию. Затем сдвинем код циклически, чтобы число 1 находилось на последнем месте. Выберем числа c'_i как четные числа, равные либо $2\lfloor 2^{n+1}/(n+2) \rfloor$, либо $2\lceil 2^{n+1}/(n+2) \rceil$ с суммой 2^{n+2}, отсортированные по возрастанию. Выберем $b_i = 2c_i - c'_{i+2}/2$. Уменьшим b_1 на один. Отметим b_i вхождения числа i, причем обязательно отметим последнее вхождение единицы. Теперь применим трансформацию из предыдущего абзаца, получим сбалансированный код Грея для $n + 2$.

К сожалению, эта, уже сама по себе трудно придумываемая конструкция, не работает для $n = 3$. Впрочем, многие способы отметить 5 позиций в коде Грея для $n = 3$ приводят к успеху, например, отметить одну единицу и четыре двойки.

Задача C. Сколько строк меньше

Автор задачи: Евгений Карпович, разработчики: Григорий Шовкопляс, Евгений Карпович

Для решения данной задачи воспользуемся структурой данных бор. Положим все строки набора D в бор.

Каждая вершина бора соответствует некоторому префиксу хотя бы одной строки из набора. Мы хотим поддерживать актуальную вершину — наибольший префикс текущей версии строки s, который есть в боре.

Далее воспользуемся динамическим программированием. Посчитаем для каждой вершины бора v величину `dp[v][c]` — какая вершина бора станет актуальной, если вершина v соответствовала некоторому префиксу строки s, а затем со следующего символа произошла модификация символом c. Также посчитаем для каждой величину `ans[v][c]` — сколько строк в боре меньше префикса, соответствующего v, если следующий символ будет c.

Как тогда обрабатывать модификации? Мы поддерживаем вершину в боре и знаем следующий символ текущей версии строки s.

- Если модификация не затрагивает префикс и следующий символ за ним, то такие модификации можно игнорировать, они не повлияют на ответ.
- Иначе поднимемся в боре в вершину, для которой префикс будет совпадать с новой версией строки s, а затем с помощью величины `dp[v][c]` найдем новую актуальную вершину, следующий символ, очевидно, будет c.

Итоговая сложность решения будет $O(\alpha \cdot \sum |D_i| + q)$, где α — размер алфавита, $\sum |D_i|$ — суммарная длина строк набора.

Задача D. Запись на экзамен

Автор задачи: Сергей Копелиович, разработчик задачи: Григорий Хлытин

Для решения этой задачи воспользуемся бинарным поиском по ответу.

Пусть ans — текущий ответ, для которого мы хотим проверить свойство: можно ли добиться, чтобы недовольство любого студента не превышало ans (заметим, что эта функция монотонна, то есть, если мы можем добиться, что недовольство любого студента не превышает ans, то оно не превышает и любое k, где $k > ans$).

Чтобы проверить это свойство для ответа ans, для каждого студента попробуем сместить его на максимальное количество дней w влево, такое что $w \leqslant ans$.

Это можно сделать за $O(n)$, так как достаточно пройтись слева-направо по всем дням i, на которые записаны студенты, и поддерживать позицию j, указывающую на самый левый «свободный» день (на который еще остались свободные места). Осталось проверять, если j находится от i не дальше, чем на ans дней, и при этом мы можем переместить x студентов со дня i на день j — то сделаем это для наибольшего x.

Если задача имеет ответ, отличный от -1, то этот ответ будет лежать в пределах от 0 до $n - 1$ включительно (так как при перемещении студента с первого на последний день, его недовольство будет равно $n - 1$). Исходя из этого факта, бинарный поиск отработает за $O(\log n)$.

Итоговая асимптотика решения будет $O(n \cdot \log n)$.

Задача E. Справедливое ограбление

Автор задачи и разработчик: Даниил Орешников

Рассмотрим произвольное k. Заметим, что ответ зависит только от минимального и максимального состояний жителей в левой и правой частях улицы. В левую часть относим всех людей, состояние которых не меняется, а в правую — тех, кто теряет долю t от состояния. Также помимо этого заметим, что условие на максимизацию украденного при прочих равных равносильно условию на максимизацию t, что можно увидеть из данной в условии формулы для b.

Введем следующие обозначения: $\min_{\text{left}} = \min(a_1, \ldots, a_{k-1})$, $\max_{\text{left}} = \max(a_1, \ldots, a_{k-1})$ и, аналогично, $\min_{\text{right}} = \min(a_k, \ldots, a_n)$, $\max_{\text{right}} = \max(a_k, \ldots, a_n)$. Отдельно рассмотрим случай $k = 1$, в случае которого left-значения не определены. В этом случае наименее несправедливый план будет задаваться числом $t = 1$, так как, если все жители потеряют свои состояния целиком, искомая разность $\max(a^{\text{new}}) - \min(a^{\text{new}})$ будет равна минимально возможному значению 0, а украдено в сумме будет максимально возможное значение.

При $k > 1$ же достаточно рассмотреть четыре введенные величины. Глобальный максимум после ограбления будет равен $\max(\max_{\text{left}}, (1-t)\max_{\text{right}})$, тогда как минимум будет равен $\min(\min_{\text{left}}, (1-t)\min_{\text{right}})$. Достаточно рассмотреть случаи взаимного расположения этих величин, чтобы определить оптимальные значения t:

1. $\max_{\text{left}} \geqslant \max_{\text{right}}$ и $\min_{\text{left}} \geqslant \min_{\text{right}}$
   В таком случае $\max(a^{\text{new}}) = \max_{\text{left}}$, так как максимум в правой части не увеличится, и аналогично $\min(a^{\text{new}}) = (1-t)\min_{\text{right}}$. Разность между этими значениями минимальна при $t = 0$.

2. $\max_{\text{left}} \geqslant \max_{\text{right}}$ и $\min_{\text{left}} < \min_{\text{right}}$
   В этом случае максимум также фиксирован и равен $\max_{\text{left}}$, а тогда для минимизации несправедливости минимум должен быть как можно больше. Но он гарантированно будет не больше $\min_{\text{left}}$, а максимальное t, при котором достигается $\min(a^{\text{new}}) = \min_{\text{left}}$, удовлетворяет равенству $\min_{\text{left}} = (1-t)\min_{\text{right}}$, то есть $t = 1 - \frac{\min_{\text{left}}}{\min_{\text{right}}}$.

3. $\max_{\text{left}} < \max_{\text{right}}$ и $\min_{\text{left}} \geqslant \min_{\text{right}}$
   Аналогично первому случаю, $\min(a^{\text{new}})$ гарантированно равен $(1-t)\min_{\text{right}}$. А максимум будет зависеть от t: при $t \leqslant 1 - \frac{\max_{\text{left}}}{\max_{\text{right}}}$ он равен $(1-t)\max_{\text{right}}$, достигая значения $\max_{\text{left}}$ в $t = 1 - \frac{\max_{\text{left}}}{\max_{\text{right}}}$, после чего он перестает уменьшаться. Так как минимум при этом продолжает уменьшаться, большие значения t приведут к большей несправедливости. А при t в данном интервале разность между максимумом и минимумом равна $(1-t)(\max_{\text{right}} - \min_{\text{right}})$, что минимально при максимально возможном $t = 1 - \frac{\max_{\text{left}}}{\max_{\text{right}}}$.

4. $\max_{\text{left}} < \max_{\text{right}}$ и $\min_{\text{left}} < \min_{\text{right}}$
   Объединяя рассмотренные выше идеи, делаем вывод, что до порога $t = 1 - \frac{\max_{\text{left}}}{\max_{\text{right}}}$ максимальное состояние убывает, и либо минимальное равно $\min_{\text{left}}$, тогда искомая разность убывает, либо эта искомая разность равна $(1-t)(\max_{\text{right}} - \min_{\text{right}})$ и тоже убывает.
   Если при данном t оказывается выполнено $\min_{\text{left}} < (1-t)\min_{\text{right}}$, можно поднять t до $1 - \frac{\min_{\text{left}}}{\min_{\text{right}}}$, тем самым не изменив искомую разность, но увеличив количество награбленного. Таким образом, ответ равен $t = 1 - \min\left(\frac{\max_{\text{left}}}{\max_{\text{right}}}, \frac{\min_{\text{left}}}{\min_{\text{right}}}\right)$.

Заметим, что последний ответ так же совпадает с полученными нами ответами во втором и третьем пунктах. Чтобы получить универсальный ответ, достаточно взять

$$t = \max\left\{0; 1 - \frac{\max_{\text{left}}}{\max_{\text{right}}}; 1 - \frac{\min_{\text{left}}}{\min_{\text{right}}}\right\}.$$

К тому же ответу можно было прийти, заметив, что t невыгодно увеличивать только в случае, когда $\max(a^{\text{new}}) = \max_{\text{left}}$, а $\min(a^{\text{new}}) = (1-t)\min_{\text{right}}$. Описанный выше ответ получался объединением ограничений на t, необходимых для достижения такого случая.

Таким образом, для фиксированного k ответ может быть найден за $\mathcal{O}(1)$ времени, если известны $[\min \mid \max]_{\text{left}|\text{right}}$. Эти же величины могут быть предподсчитаны за $\mathcal{O}(n)$ с помощью линейного прохода по массиву a в обе стороны (префиксные и суффиксные минимум и максимум).

Задача F. Количество антител

Автор задачи и разработчик: Рита Саблина

Заметим, что тяжёлых сетей всего может быть $V_h \cdot D_h \cdot J_h$ — для каждой цепи выбирается по одному варианту из V_h, D_h и J_h для из участков V, D и J, соответственно.

Аналогично, лёгких цепей типа κ может быть $V_\kappa \cdot J_\kappa$, а лёгких цепей типа λ может быть $V_\lambda \cdot J_\lambda$.

Каждая тяжёлая цепь соединяется или с лёгкой типа κ, или с лёгкой типа λ. В молекуле иммуноглобулина по паре тяжёлых и лёгких цепей, но цепи в каждой паре одинаковые. Тогда общее количество возможных молекул иммуноглобулинов считается по формуле $(V_\kappa \cdot J_\kappa + V_\lambda \cdot J_\lambda) \cdot V_h \cdot D_h \cdot J_h$

Этой задачей мы хотели показать, как сложно бывает программисту разобраться в непрофильной для него узкой предметной области.

Задача G. Парусная математика

Автор задачи и разработчик: Даниил Орешников

Рассмотрим равенство маневренности и стабильности: $a_1a_4 + a_2 + a_3 = a_1 + a_4 + a_2a_3$. Заметим, что xy может быть выражен как $(x-1)(y-1) + x + y - 1$, таким образом, если обозначить за A сумму $a_1 + a_2 + a_3 + a_4$, получаем, что маневренность равна

$$m = (a_1 - 1)(a_4 - 1) + a_1 + a_4 - 1 + a_2 + a_3 = A - 1 + (a_1 - 1)(a_4 - 1),$$

а стабильность, аналогично, равна

$$s = A - 1 + (a_2 - 1)(a_3 - 1).$$

Следовательно, необходимо добиться равенства

$$m - A + 1 = \Big[(a_1 - 1)(a_4 - 1) = (a_2 - 1)(a_3 - 1)\Big] = s - A + 1.$$

Поскольку в остальном порядок парусов не имеет значения, можно не теряя общности выбрать в ответ в качестве a_1 минимальный из четырех парусов, а в качестве a_2 — минимальный из парусов центральной мачты. Таким образом, достаточно найти ответ, в котором $a_1 \leqslant a_2 \leqslant a_3 \leqslant a_4$.

Рассмотрим исходные размеры кусков ткани в порядке неубывания: $t_{p_1} \leqslant t_{p_2} \leqslant t_{p_3} \leqslant t_{p_4}$. Эта перестановка p затем станет первой строчкой ответа. Обозначим за q_i величину $t_{p_i} - 1$, то есть i-й по величине размер куска ткани, уменьшенный на 1. Тогда заметим, что $(a_1-1)(a_4-1) = (a_2-1)(a_3-1)$ ограничены сверху $\min(q_1 \cdot q_4, q_2 \cdot q_3)$. Действительно, если эти два произведения равны, а каждый из множителей не превосходит некоторого q_i по условию задачи, то оба произведения не превосходят минимального из произведений двух q_i. Переберем, какое q будет стоять «в паре» с q_4: если это q_1, то эти величины не превосходят $q_1 \cdot q_4$ и $q_2 \cdot q_3$ соответственно, что и требовалось доказать. Если же q_4 стоит в паре с q_2 или q_3, то произведение во второй паре будет не превосходить $q_1 \cdot q_2$ или $q_1 \cdot q_3$, каждое из которых, в свою очередь, не превосходит как $q_2 \cdot q_3$, так и $q_1 \cdot q_4$.

Мы доказали оценку на $m - A + 1 = s - A + 1$ сверху, равную $\min(q_1 \cdot q_4, q_2 \cdot q_3)$. Заметим, что она достигается. Более того, для ее достижения, как окажется дальше, понадобится обрезать не более одного куска ткани, что позволяет нам одновременно с максимизацией этого произведения максимизировать и сумму размеров парусов A. Если обе эти величины принимают максимально возможные значения, итоговые m и s тоже максимальны.

Если $q_1 \cdot q_4 < q_2 \cdot q_3$, можно уменьшить произведение $q_2 \cdot q_3$ до $q_1 \cdot q_4$, не меняя q_1 и q_4, и наоборот. Осталось понять, каким именно образом следует приводить большее произведение к меньшему. Не теряя общности, рассмотрим случай $q_1 \cdot q_4 < q_2 \cdot q_3$. Тогда мы гарантированно берем $a_1 = q_1 + 1$ и $a_4 = q_4 + 1$. Чтобы понять, какие следует взять a_2 и a_3, вспомним, что $s = A - 1 + (a_2 - 1)(a_3 - 1)$, где $A = a_1 + a_2 + a_3 + a_4$. А в таком случае от нас требуется при фиксированном произведении $(a_2 - 1)(a_3 - 1)$ максимизировать сумму $a_1 + a_2 + a_3 + a_4$. А так как a_1 и a_4 тоже уже выбраны, достаточно максимизировать сумму значений (a_2-1) и (a_3-1) при фиксированном их произведении. Известно, что сумма двух чисел с фиксированным произведением растет с отдалением этих чисел друг от друга, так что следует уменьшить минимальный из двух парусов.

Итоговые размеры парусов тогда будут равны $q_1 + 1, \frac{q_1 q_4}{q_3} + 1, q_3 + 1, q_4 + 1$. Аналогично в случае $q_1 \cdot q_4 \geqslant q_2 \cdot q_3$ оптимальным ответом будет $\frac{q_2 q_3}{q_4} + 1, q_2 + 1, q_3 + 1, q_4 + 1$. В качестве перестановки p при этом следует вывести перестановку, рассмотренную выше для упорядочивания t_i по неубыванию. Время работы решения равно $\mathcal{O}(1)$.

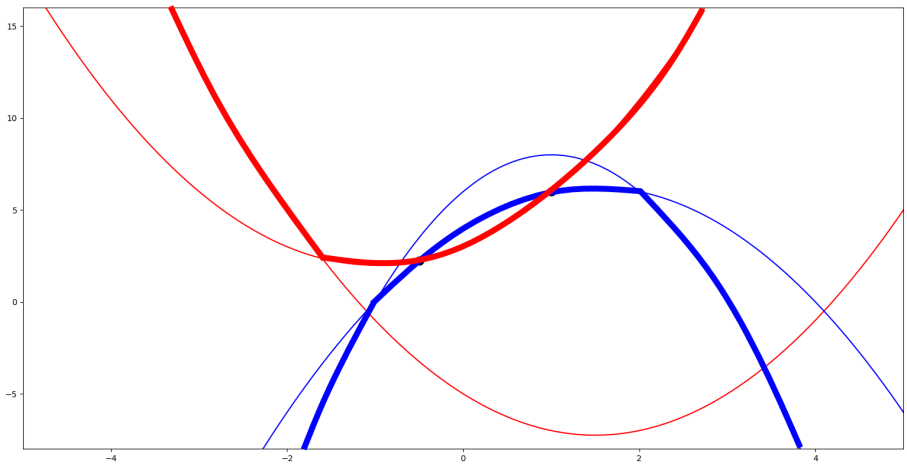
Задача H. Море парабол

Автор задачи: Михаил Пядеркин, разработчик: Сергей Мельников, разбор: Андрей Станкевич

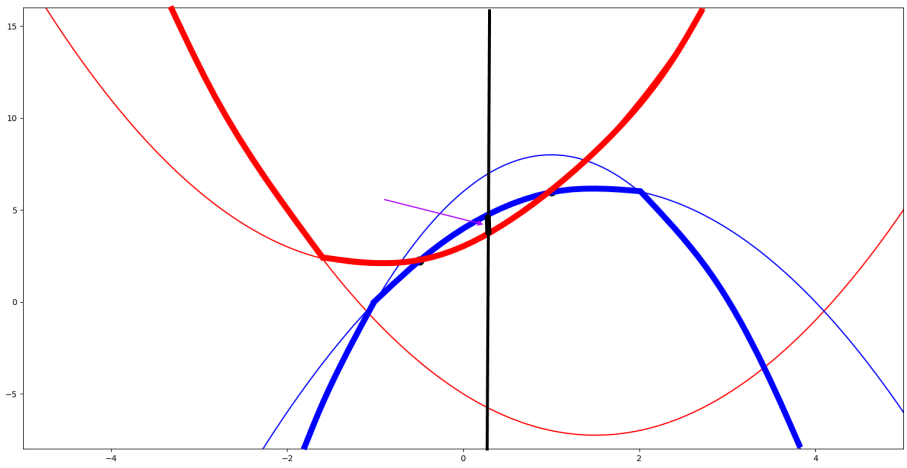
Для решения задачи можно применить алгоритм, исходно предложенный для пересечения полуплоскостей. Оказывается, он работает для любых выпуклых фукнций.

Рассмотрим все параболы с $a > 0$, то есть такие, что точка должна находиться сверху от них. Для каждой координаты x выберем самую верхнюю точку на таких параболах. Мы получим функцию $low(x)$, которая является выпуклой вниз.

Аналогично, рассмотрим все параболы с $a < 0$, то есть такие, что точка должна находиться снизу от них. Для каждой координаты x выберем самую нижнюю точку на таких параболах. Мы получим функцию $up(x)$, которая является выпуклой вверх.



Рассмотрим теперь разность $f(x) = up(x) - low(x)$. Эта функция является выпуклой вверх. Найдем её максимум троичным поиском. Чтобы вычислить значение в точке x просто вычислим значение всех квадратичных трехчленов для x и выберем максимум из значений где $a > 0$ и минимум из значений где $a < 0$.



Максимум f положительный, используем найденную координату в качестве x. Найдем на этой координате x максимум из значений где $a > 0$ и минимум из значений где $a < 0$. Среднее из этих значений отлично подойдет в качестве y.

Задача I. Поле чудес

Автор задачи: Геннадий Короткевич, разработчик: Николай Ведерников

Катя будет действовать следующий образом:

Если у неё ровно одно слово, то победа гарантирована.

Иначе если она не может найти такой буквы, которая встречается во всех словах, то не может гарантировано победить. Потому что, какую бы Катя букву ни назвала, всегда существует такое слово, в котором нет этой буквы, а оно могло быть загадано, и ход перейдёт к следующему.

Иначе Катя называет букву, которая встречается в каждом слове. Если таких букв несколько, Катя может назвать любую из них. Когда Катя назовёт букву, она сможет разбить список слов на несколько групп. В одной группе будут слова, в которых названная буква находится на одних и тех же позициях.

Для каждой группы повторим наш алгоритм, убрав из рассмотрения названную Катей букву.

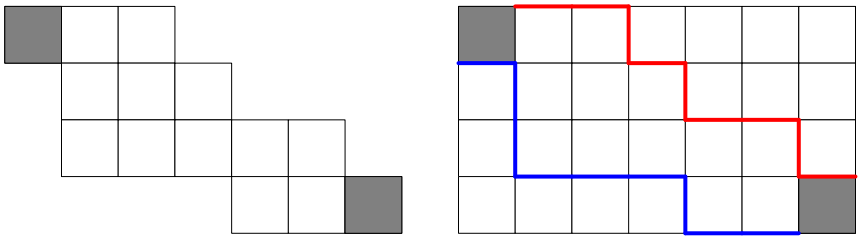
Другими словами, назвав букву, разбиваем множество слов на подмножества, и от каждого запускаем рекурсивно алгоритм. Так как всего букв 26, то максимальная глубина рекурсии будет 26. Кроме того, каждое слово на каждой глубине встречается максимум один раз. Поэтому итоговое время работы будет $O(s \cdot N \cdot L)$, где s — количество букв в алфавите.

Задача J. Юрик и урок технологии

Автор задачи: Андрей Станкевич, разработка: Михаил Первеев

Для начала поймем, что последние два условия, которым должна удовлетворять красивая доска, означают, что в каждой строке должно быть удалено некоторое, возможно нулевое, количество клеток слева и справа. При этом количество удаленных слева клеток в строке i должно быть не меньше, чем количество удаленных клеток слева в строке $i-1$. Аналогично, количество удаленных клеток справа в строке i должно быть не больше, чем количество удаленных клеток справа в строке $i-1$. Если условия на монотонность количества удаленных слева и справа клеток не выполнены, то найдется столбец, для которого не будет выполнено пятое свойство.

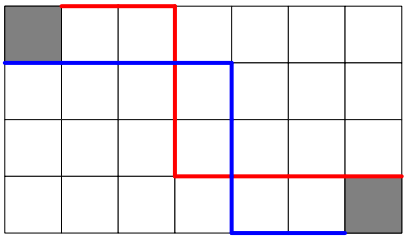
Подобные рассуждения приводят к тому, что любой корректный способ удаления клеток слева и справа в каждой строке фиксируется двумя путями следующего вида:

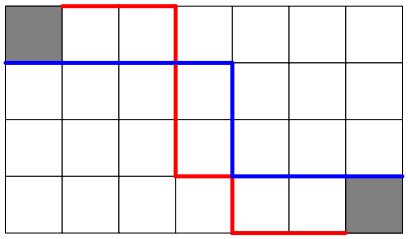


Для того, чтобы третье свойство было выполнено, пути не должны пересекаться. Таким образом, задача свелась к тому, чтобы вычислить количество пар непересекающихся путей, которые начинаются и заканчиваются в точках, показанных на рисунке.

Для начала вычислим, сколько всего существует пар таких путей. Каждый путь состоит из $N+M-2$ отрезков длины 1, каждый из которых идет либо вниз, либо вправо. Значит всего таких путей существует $\binom{N+M-2}{M-1}$, так как среди $N+M-2$ отрезков нужно выбрать $M-1$ горизонтальный отрезок. Таким образом, количество пар путей равно $\binom{N+M-2}{M-1}^2$.

Теперь нужно вычесть из этого числа количество пар пересекающихся путей. Рассмотрим изображение пары пересекающихся путей и заметим, что каждой такой паре однозначно соответствует пара других путей, и наоборот. Рассмотрим последнюю точку пересечения путей, она существует, так как мы рассматриваем только пересекающиеся пары путей. После этой точки пути расходятся и больше никогда не пересекаются. Возьмем эти концы путей и поменяем их местами: конец первого пути сделаем концом второго пути и наоборот. Пример подобного соответствия изображен на рисунке:





Количество пар таких путей равно $\binom{N+M-2}{M-2} \cdot \binom{N+M-2}{N-2}$. Таким образом, ответ равен: $\binom{N+M-2}{M-1}^2 - \binom{N+M-2}{M-2} \cdot \binom{N+M-2}{N-2}$. При помощи несложных преобразований можно получить более красивую формулу: $\dfrac{\binom{N+M-1}{M} \cdot \binom{N+M-1}{M-1}}{N+M-1}$.

Задача K. Железнодорожная сортировка

Автор и разработчик задачи: Андрей Станкевич, разбор: Арсений Кириллов, Рита Саблина

Будем отправлять вагоны на выходной путь по одному в порядке возрастания номера.

Если i-й вагон всё ещё находится на входном пути к моменту, когда первый $i-1$ вагон уже на выходном пути, то отправим все вагоны перед вагоном i на входном пути и его самого в первый тупик, после чего отправим его из первого тупика на выходной путь. Для этого потребуется не более, чем $n-1$ команда, чтобы переместить предыдущие вагоны, и ещё две команды, чтобы переместить i-й вагон в первый тупик и из него.

Если же i-й вагон находится в одном из тупиков, то отправим все вагоны перед ним в том же тупике в другой тупик, а сам этот вагон отправим на выходной путь. Для этого потребуется не более, чем $n-1$ команда, чтобы переместить предыдущие вагоны, и ещё одна команда, чтобы переместить i-й вагон из тупика.

В результате для каждого из n вагонов используется не более $n+1$ команды, значит, всего потребуется не более n^2+n команд, что меньше, чем $2 \cdot 10^6$.

Задача L. День рождения

Автор задачи и разработчик: Евгений Карпович

Для удобства будем считать, что $a_i \leqslant b_i$ (иначе просто поменяем их местами).

Научимся решать задачу на всем массиве. Заметим, что если сумма по всем b_i не делится на k, то она является ответом, иначе нужно ровно на одной карточке выбрать значение a_i вместо значения b_i, при этом $a_i \bmod k \neq b_i \bmod k$ и нужно минимизировать значение $b_i - a_i$.

Сделаем два массива c и $pref$. Если $a_i \bmod k \neq b_i \bmod k$, то $c_i = b_i - a_i$, иначе $c_i = +\infty$. А $pref$ — это массив префиксных сумм: $pref_i = pref_{i-1} + b_i$. Определим $min(l, r)$ как минимум на подотрезке $l \dots r$ массива c. Теперь ответом для подотрезка будет либо $pref_r - pref_{l-1}$, если это значение не кратно k, либо $pref_r - pref_{l-1} - min(l, r)$.

В обоих случаях у нас есть слагаемое $pref_r - pref_{l-1}$, поэтому давайте посчитаем сумму этих слагаемых по всем подотрезкам. Значение $pref_i$ будет взято со знаком минус (то есть в качестве левой границы) в $(n-i)$ подотрезках, и в i подотрезках со знаком плюс (то есть в качестве правой границы).

Теперь нужно отдельно учесть подотрезки, в которых $pref_r \bmod k = pref_{l-1} \bmod k$. Давайте для каждого остатка x выпишем все позиции i, в которых $pref_i \bmod k = x$. Будем решать для каждого остатка задачу независимо.

Пусть у нас есть список позиций для фиксированного остатка $pos_1, pos_2, \dots, pos_m$. Заметим, что нам нужно из нашего ответа вычесть такую величину: $\sum\limits_{l=1}^{m} \sum\limits_{r=l+1}^{m} min(pos_l + 1, pos_r)$. Для этого сделаем массив d, где $d_i = min(pos_i + 1, pos_{i+1})$ (минимум на отрезке массива c можно находить с помощью дерева отрезков). В массиве d нам теперь нужно посчитать сумму минимумов по всем подотрезкам, а это уже можно сделать с помощью линейных алгоритмов или дерева отрезков. Для этого для каждого элемента i посчитаем, в скольких подотрезках он является левым минимумом.

Пусть $left$ — ближайший меньше либо равный элемент слева от i, $right$ — ближайший меньший элемент справа от i, тогда количество подотрезков, в котором элемент с индексом i является левым минимумом, равно $(i - left) \cdot (right - i)$.

Также нужно будет аккуратно учесть те отрезки, на которых нельзя выбрать числа на карточках, чтобы получить сумму не кратную k. На этих отрезках минимальное значение равно $+\infty$, и нужно будет для каждого такого подотрезка вычесть величину $pref_r - pref_{l-1}$.

Решение работает за $O(n \log n)$.