Алгоритмы
Домашнее задание №4
Стрелец Владимир

Задача 1.

Задача 7: https://contest.yandex.ru/contest/1080/run-report/650495/

1) Определяем минимальные границы для каждого из цветов. (Храним массив структур с полями "левая (правая/верхняя/нижняя) граница" и расширяем по мере надобности. Заодно узнаём, какие цвета полностью скрыты под другими. Пробежка по холсту - $O(n^2)$.

2) Определяем зависимости вида "цвет a РАНЬШЕ цвета b". Это делаем 4 пробежками всего холста (справа налево, слева направо, сверху вниз и снизу вверх). Зависимость имеет место, если мы находимся в границах цвета a, причём на холсте виден цвет b. Будем представлять зависимости "a раньше b" в виде дуг $a \rightarrow b$ в ориентированном графе.

   Для их определения при пробежке каждой строки/столбца будем использовать стек. В вершине стека будет храниться самый верхний (в очерёдности закраски) цвет, в границах которого мы находимся. В стеке так же могут цвета, из границ которых мы вышли. Но они нас не интересуют и они будут удалены из стека в первом подходящем случае. Изначально стек инициализируем цветом первой в столбце/строке клетки (если это не 0).

   Итак, когда мы переходим из одной клетки в другую, мы удаляем из стека все цвета, из границ которых мы вышли. А затем, если стек ещё непуст, добавляем в граф ребро, ведущее из цвета вершины стека в текущий цвет на холсте.

   Заметим, что нам не нужно находить все зависимости (то есть говорить, что текущий цвет идёт позже всех цветов, в границах которых мы находимся и которые сейчас лежат в стеке). Ведь если мы знаем, что $a \rightarrow b$, а также узнаем, что $b \rightarrow c$, то из этого автоматически будет следовать, что $a \rightarrow c$. А при добавлении цветов в стек они образуют последовательную цепочку $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \ldots \rightarrow current\_color$, которая позже будет восстановлена.

   Время - $O(n^2)$, так как в каждой строке каждый цвет будет добавлен и извлечён из стека не более одного раза, а так же будет добавлено не более n дуг в граф (не более одной при каждом переходе).

3) Нужно удалить из графа все вершины и дуги, инцидентные им, для цветов, которые не были замечены на холсте. Либо изначально не добавлять в граф. Также найдём любую закрашенную клетку, под которую и поместим изначально "скрытые" цвета.

   Далее, для построенного орграфа строим топологическую сортировку его вершин (за $O(n^2)$, так как было всего $O(n^2)$ дуг в орграфе. Таким образом, построится порядок покраски холста в цвета. А построенные ранее минимальные границы для цветов и будут границами прямоугольников для покраски.

Итак, итоговая сложность - $O(n^2)$.

Задача 2.

В данном в условии коде сравниваются не соседние элементы массива, а элементы массива с индексами $params[i].a$ и $params[i].b$, и они меняются, если $A[params[i].a]$ был больше, чем $A[params[i].b]$. Заведём орграф на n вершинах. Неравенство вида $A[params[i].a] > A[params[i].b]$ представим в виде дуги $a -> b$. То есть если сортировка остановилась, то оказалось, что выполняются равенства $params[i].a > params[i].b$ для всех i. Представим это в виде орграфа. В этом графе сразу уберём все петли, так как они никак не влияют на решение.

a) Теперь предположим, что в построенном орграфе нету какой-либо дуги $m -> n, m < n$. Тогда любой массив, в котором изначально выполняются все равенства (дуги в графе), но $A[m] > A[n]$, не будет отсортирован никогда. Значит, для корректной сортировки любого массива в орграфе обязательно должны оказаться все дуги вида $i -> i+1$.

b) Пусть теперь в орграфе помимо уже обязательно присутствующих дуг $i -> i+1$ имеется ещё какая-либо дуга вида $k -> l. k > l$. Тогда при пробежке $params$ при попадании на неравенство, соответствующее этой дуге, будет производиться нежелательный обмен, который заставит повториться сортировку ещё раз (так как флаг $finished$ будет равен 0). И сортировка снова не упорядочит массив.

c) Также вариант, когда n меньше, чем максимальный из $params[i].a$ и $params[i].b$, можно считать также не сортирующим массив правильно. Ведь во многих языках программирования это вызовет исключение. И вообще. Это неестественно!

Заметим, что если k меньше, чем $n-1$, то пункт a не может быть выполнен. А значит $n = O(k)$, то есть сложность может быть равна $O(n + k) = O(k)$.
Итак, для входных данных построим орграф. Любой массив длины n будет отсортирован правильно, если:

1) выполнен пункт с (можно проверишь за одну пробежку массива $params$ - $O(k)$);
2) в нём есть все дуги вида $i \to i+1$ (что можно проверить, просто заведя дополнительный булевый массив размера n). Время - $O(n + k) = O(k)$;
3) нет ни одной дуги вида $k \to l, k > l$ (что можно проверять просто при пробежке массива $params$, и при обнаружении сразу говорить, что не любой массив длины n будет отсортирован). Время - $O(k)$.

Задача 3.

Задача 4

Задача 9: https://contest.yandex.ru/contest/1080/run-report/649584/

Если провести дугу из любой вершины, достижимой из помеченной, в вершину, недостижимую из помеченной, то та автоматически станет достижимой. Поэтому будем считать, что все дополнительные дуги проводятся из помеченной вершины (так как она достижимая и это не испортит ответа).

1) В вершины, которые достижимы из помеченной, вести дуги не надо. Поэтому вообще удалим их из графа. Запускаем dfs из помеченной вершины и запоминаем все вершины, достижимые из помеченной вершины.
   А затем строим граф H, не содержащий запомненных на прошлом этапе вершин и инцидентных им дуг.

   Заметим, что если провести ребро в любую вершину компоненты сильной связности, то все вершины из этой компоненты станут достижимыми. Поэтому делаем пункт 2.
2) Строим конденсацию графа C для построенного графа H, так как в каждую компоненту можно проводить не более одной дополнительной дуги.
   Наблюдение. Если есть w достижима из v, но v недостижима из w, то проводить дугу в w не стоит, так как тогда вершина v останется недостижимой, но, проведя дугу в вершину v, мы сделаем обе вершины v и w достижимыми. Конденсация графа тем и примечательна, что в ней если w достижима из v, то v недостижима из w.
3) Итак, если в какую-то вершину графа C идёт ребро из другой вершины, то в эту вершину не стоит вести дугу. Таким образом, пробежимся по всем рёбрам графа C и запишем все вершины, в которые идут дуги. А затем пробежимся по всем вершинам и подсчитаем количество вершин, в которые не шло ни одной дуги.
   Это и будет ответом для задачи.

Задача 5.

Задача 6.

Рассмотрим любой корректный выбор малыша Голода.

Утверждение 1. Пусть есть строка (столбец), в которой ответы "морить/не_морить голодом" чередуются. Тогда во всех строках (столбцах) ответы чередуются.

Докажем от противного. Пусть в соседней с "чередующейся" строкой (столбцом) строке (столбце) строка - нечередующаяся. Это значит, что в ней есть 2 подряд идущих одинаковых ответа. А значит в квадрате 2x2 на пересечении этих двух строк будет 3 одинаковых ответа, что недопустимо. Значит соседняя строка (столбец) - также "чередующийся". Утверждение доказывается по индукции.

Утверждение 2. Пусть в какой-либо строке (столбце) есть 2 одинаковых, стоящих на позициях разной чётности, либо 2 разных ответа, стоящих на позициях одинаковой чётности. Тогда все столбцы (строки) "чередующиеся", то есть всё перпендикулярное направление "чередующееся".

Также докажем от противного. Пусть перпендикулярное направление (не теряя общности, столбцы) не "чередующееся", то есть в нём существует 2 подряд идущих одинаковых ответа. Тогда в соседнем столбце рядом будут стоять 2 других ответа, и так далее. То есть 2 рядом стоящие строки - "чередующиеся". А это противоречит условию утверждения. Значит, перпендикулярное направление - "чередующееся".

Итак, мы построили аппарат для решения задачи C:
Сначала пробежимся про каждой строке и поищем ограничение в "чередовании" горизонтального (строкового) направления (в виде двух минских/общажных студентов, стоящих на позициях одинаковой чётности или в виде минского и общажного студента, стоящих на позициях разной чётности; ведь на минских и общажных студентах ответ однозначен).
Если оно было найдено, то это значит, что должны чередоваться все столбцы.
Если же не было найдено, то это значит, что строки могут чередоваться.
Теперь пробежимся по всем столбцам и поищем такое же ограничение.

а) Если было найдено ограничение и в горизонтальном и в вертикальном направлении, то получено противоречие и корректного выбора для малыша Голода нету. Ответ - 0.

б) Если ограничение было найдено лишь в одном направлении, то это значит, что ответы в другом направлении чередуются. Каждая непустая строка (столбец) фиксируют это чередование, а каждая пустая - даёт 2 различных варианта. Также

варианты для различных строк независимы:
| ……| | ……|
| … M O M O … | и | … M O M O … | -- оба корректных варианта.
| … O M O M … | | … M O M O … |
| ……| | ……|
Поэтому ещё одной пробежкой (или сразу за первую) подсчитаем количество пустых строк (или столбцов, в зависимости от того, что "чередуется") $count\_empty\_rows$. И тогда ответом на задачу будет число $2^{count\_empty\_rows}$.

в) Если ограничение не было найдено, то "чередоваться" могут как столбцы, так и строки. Но ещё нужно проверить, не получаются ли одинаковые варианты при горизонтальном и вертикальном "чередованиях". Такие варианты -- лишь выбор "морить/не_морить голодом" в шахматном порядке. А таких порядков 2 - в зависимости от того, что стоит в клетке [1, 1]. Количество пересекающихся вариантов проверим отдельно. Пусть их $count\_intersects$. Тогда ответом будет

$$2^{coutt\_empty\_rows} + 2^{count\_empty\_columns} - count\_intersects.$$

Задача 7.

Прошёл 86 тестов. Но на 87 падает :C
А можно тест, хотя бы доразобраться, что за ошибка у меня?
Неудачная посылка: https://contest.yandex.ru/contest/1080/run-report/650495/

Задача 8.

Задача 9.

Удачная посылка: https://contest.yandex.ru/contest/1080/run-report/649584/