Региональный этап XXVI Всероссийской командной олимпиады школьников по программированию

Саратовский ГУ

26 октября 2025 г.

A. Выкладывание карточек

Числа от 1 до n записаны в следующем порядке: сначала четные по возрастанию, потом нечетные по убыванию. Определить, какое число будет k-м

A. Выкладывание карточек

Числа от 1 до n записаны в следующем порядке: сначала четные по возрастанию, потом нечетные по убыванию. Определить, какое число будет k-м

- Решение 1: явно построить список из всех четных чисел по возрастанию, потом из нечетных по убыванию, потом соединить и вывести k-й элемент

A. Выкладывание карточек

Числа от 1 до n записаны в следующем порядке: сначала четные по возрастанию, потом нечетные по убыванию. Определить, какое число будет k-м

- Решение 1: явно построить список из всех четных чисел по возрастанию, потом из нечетных по убыванию, потом соединить и вывести k-й элемент
- Для быстрого формирования можно использовать либо циклы, либо range и конвертировать их в list

A. Выкладывание карточек

Числа от 1 до n записаны в следующем порядке: сначала четные по возрастанию, потом нечетные по убыванию. Определить, какое число будет k-м

- Решение 1: явно построить список из всех четных чисел по возрастанию, потом из нечетных по убыванию, потом соединить и вывести k-й элемент
- Для быстрого формирования можно использовать либо циклы, либо range и конвертировать их в list
- Решение 2: понять, четное нужно число или нечетное, и вычислить его по формуле

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

- Нам интересны только позиции 1 и n, остальные можно расставить как угодно

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

- Нам интересны только позиции 1 и n, остальные можно расставить как угодно
- Можно перебрать позицию 1, позицию n и посчитать массивы a и b, но это слишком медленно

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

- Нам интересны только позиции 1 и n, остальные можно расставить как угодно
- Можно перебрать позицию 1, позицию n и посчитать массивы a и b, но это слишком медленно
- Ускорение: массив a зависит только от позиции 1, а массив b — только от позиции n

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

- Нам интересны только позиции 1 и n, остальные можно расставить как угодно
- Можно перебрать позицию 1, позицию n и посчитать массивы a и b, но это слишком медленно
- Ускорение: массив a зависит только от позиции 1, а массив b — только от позиции n
- Для каждого i посчитаем, какая будет сумма элементов в a (или в b), если поставить на эту позицию 1 (или n)

B. Создание перестановки

По перестановке p строятся два массива a и b — расстояние от каждого элемента до минимума и максимума. Построить такую перестановку p длины n, что сумма всех элементов a и b в точности равна m

- Нам интересны только позиции 1 и n, остальные можно расставить как угодно
- Можно перебрать позицию 1, позицию n и посчитать массивы a и b, но это слишком медленно
- Ускорение: массив a зависит только от позиции 1, а массив b — только от позиции n
- Для каждого i посчитаем, какая будет сумма элементов в a (или в b), если поставить на эту позицию 1 (или n)
- После этого уже можно перебрать пару позиций и посчитать сумму для нее за $O(1)$

C. Порталы

Дан ориентированный граф. В некоторых вершинах можно создать порталы. Между любыми двумя вершинами с порталами можно перемещаться (в любую сторону). Постройте минимальное количество порталов, чтобы все вершины были достижимы друг из друга

C. Порталы

Дан ориентированный граф. В некоторых вершинах можно создать порталы. Между любыми двумя вершинами с порталами можно перемещаться (в любую сторону). Постройте минимальное количество порталов, чтобы все вершины были достижимы друг из друга

- В вершинах, в которые ничего не входит, точно нужны порталы, иначе в них нельзя попасть

C. Порталы

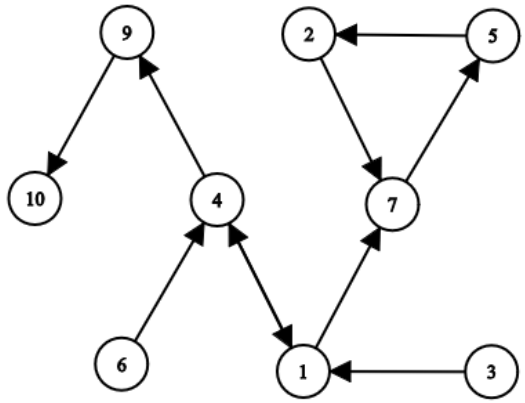
Дан ориентированный граф. В некоторых вершинах можно создать порталы. Между любыми двумя вершинами с порталами можно перемещаться (в любую сторону). Постройте минимальное количество порталов, чтобы все вершины были достижимы друг из друга

- В вершинах, в которые ничего не входит, точно нужны порталы, иначе в них нельзя попасть
- Аналогично с вершинами, из которых ничего не выходит

C. Порталы

Дан ориентированный граф. В некоторых вершинах можно создать порталы. Между любыми двумя вершинами с порталами можно перемещаться (в любую сторону). Постройте минимальное количество порталов, чтобы все вершины были достижимы друг из друга

- В вершинах, в которые ничего не входит, точно нужны порталы, иначе в них нельзя попасть
- Аналогично с вершинами, из которых ничего не выходит
- Это необходимые порталы, но не обязательно достаточные



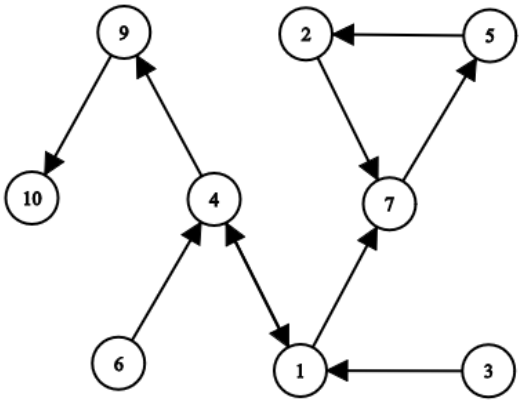
C. Порталы

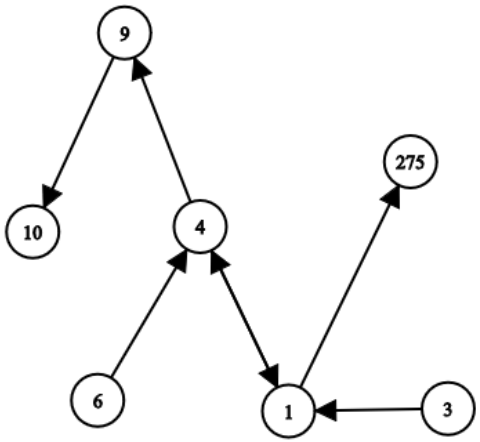
Как детектировать такие ситуации?

C. Порталы

Как детектировать такие ситуации?

- Сжать каждую группу вершин, где они достижимы друг из друга, в одну вершину

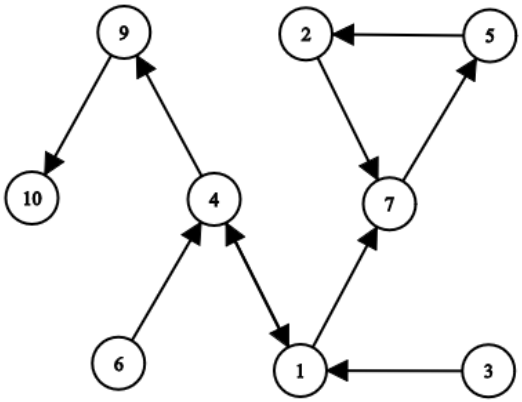


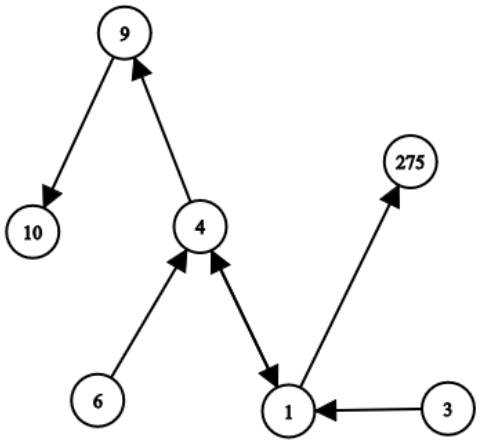


C. Порталы

Как детектировать такие ситуации?

- Сжать каждую группу вершин, где они достижимы друг из друга, в одну вершину



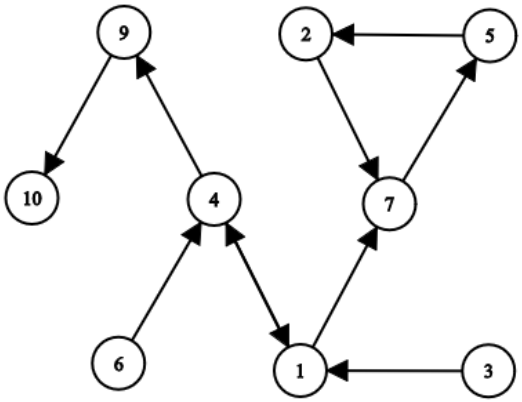


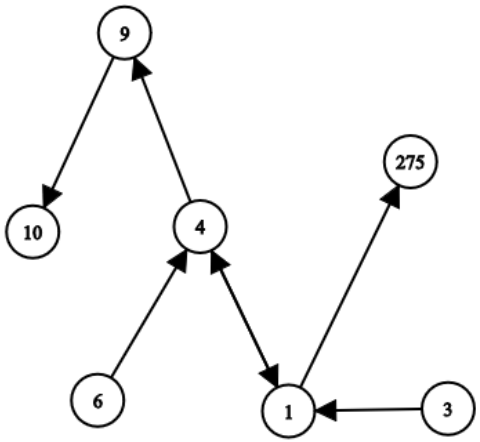
- Тогда такая группа вершин превратится в вершину, у которой нет входящих или исходящих ребер

C. Порталы

Как детектировать такие ситуации?

- Сжать каждую группу вершин, где они достижимы друг из друга, в одну вершину



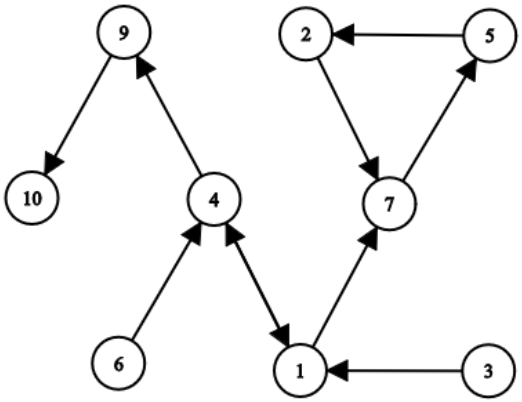


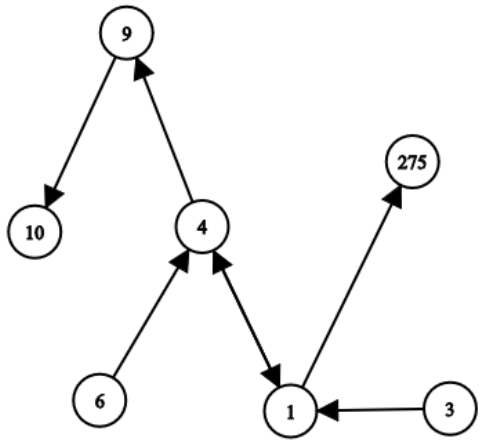
- Тогда такая группа вершин превратится в вершину, у которой нет входящих или исходящих ребер
- Такие группы вершин — компоненты сильной связности

C. Порталы

Как детектировать такие ситуации?

- Сжать каждую группу вершин, где они достижимы друг из друга, в одну вершину





- Тогда такая группа вершин превратится в вершину, у которой нет входящих или исходящих ребер
- Такие группы вершин — компоненты сильной связности
- Воспользуемся алгоритмом Косараю или Тарьяна для их выделения

C. Порталы

Что после выделения КСС?

C. Порталы

Что после выделения КСС?

- Построим конденсацию: каждой компоненте назначим номер и каждое ребро исходного графа превратим в ребро для соответствующих компонент

C. Порталы

Что после выделения КСС?

- Построим конденсацию: каждой компоненте назначим номер и каждое ребро исходного графа превратим в ребро для соответствующих компонент
- Если ребро внутри компоненты, его надо проигнорировать

C. Порталы

Что после выделения КСС?

- Построим конденсацию: каждой компоненте назначим номер и каждое ребро исходного графа превратим в ребро для соответствующих компонент
- Если ребро внутри компоненты, его надо проигнорировать
- Получим ацикличный граф, а в нем для каждой вершины есть исток, из которого она достижима, и сток, который из нее достижим

C. Порталы

Что после выделения КСС?

- Построим конденсацию: каждой компоненте назначим номер и каждое ребро исходного графа превратим в ребро для соответствующих компонент
- Если ребро внутри компоненты, его надо проигнорировать
- Получим ацикличный граф, а в нем для каждой вершины есть исток, из которого она достижима, и сток, который из нее достижим
- Поэтому достаточно расставить порталы во всех истоках и стоках

C. Порталы

Что после выделения КСС?

- Построим конденсацию: каждой компоненте назначим номер и каждое ребро исходного графа превратим в ребро для соответствующих компонент
- Если ребро внутри компоненты, его надо проигнорировать
- Получим ацикличный граф, а в нем для каждой вершины есть исток, из которого она достижима, и сток, который из нее достижим
- Поэтому достаточно расставить порталы во всех истоках и стоках
- Частный случай: только одна КСС, ответ 0 (был в условии)

D. Нужное количество единиц

Посчитать кол-во чисел от a до b с k единицами в двоичной записи

D. Нужное количество единиц

Посчитать кол-во чисел от a до b с k единицами в двоичной записи

- Посчитать кол-во чисел меньших $b + 1$, затем вычесть кол-во чисел меньших a

D. Нужное количество единиц

Посчитать кол-во чисел от a до b с k единицами в двоичной записи

- Посчитать кол-во чисел меньших $b + 1$, затем вычесть кол-во чисел меньших a
- Реализуем одну функцию $calc(r)$, вызовем дважды

D. Нужное количество единиц

Посчитать кол-во чисел от a до b с k единицами в двоичной записи

- Посчитать кол-во чисел меньших $b + 1$, затем вычесть кол-во чисел меньших a
- Реализуем одну функцию $calc(r)$, вызовем дважды
- r можно перевести в двоичную систему счисления

D. Нужное количество единиц

Решение 1. Динамическое программирование

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$
- Сколько старших цифр поставили

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$
- Сколько старших цифр поставили
- Сколько среди них единиц

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$
- Сколько старших цифр поставили
- Сколько среди них единиц
- Флаг «правда ли, что набираемое число уже строго меньше r»

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$
- Сколько старших цифр поставили
- Сколько среди них единиц
- Флаг «правда ли, что набираемое число уже строго меньше r»
- Переберем очередную цифру — 0 или 1

D. Нужное количество единиц

Решение 1. Динамическое программирование

- $dp[digits][ones][less]$
- Сколько старших цифр поставили
- Сколько среди них единиц
- Флаг «правда ли, что набираемое число уже строго меньше r»
- Переберем очередную цифру — 0 или 1
- Ответ лежит в $dp[len(r)][k][true]$

D. Нужное количество единиц

Решение 2. Комбинаторика

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше
- Переберем длину общего префикса

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше
- Переберем длину общего префикса
- После префикса в r обязана идти цифра 1

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше
- Переберем длину общего префикса
- После префикса в r обязана идти цифра 1
- Если на префиксе m единиц, то осталось поставить $k - m$ единиц

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше
- Переберем длину общего префикса
- После префикса в r обязана идти цифра 1
- Если на префиксе m единиц, то осталось поставить $k - m$ единиц
- $C(len(r) - pref - 1, k - m)$ способов

D. Нужное количество единиц

Решение 2. Комбинаторика

- Число x меньше числа r значит, что у x и r есть общий равный префикс, а следующая цифра после него в x меньше
- Переберем длину общего префикса
- После префикса в r обязана идти цифра 1
- Если на префиксе m единиц, то осталось поставить $k - m$ единиц
- $C(len(r) - pref - 1, k - m)$ способов
- Чтобы избежать переполнения, посчитаем биномиал с помощью треугольника Паскаля

E. Массив Монокарпа

Удалить самый короткий отрезок чисел из массива так, чтобы разность между максимумом и минимумом стала не больше d

E. Массив Монокарпа

Удалить самый короткий отрезок чисел из массива так, чтобы разность между максимумом и минимумом стала не больше d

- Если можно удалить k чисел, то можно удалить и $k + 1$ число

E. Массив Монокарпа

Удалить самый короткий отрезок чисел из массива так, чтобы разность между максимумом и минимумом стала не больше d

- Если можно удалить k чисел, то можно удалить и $k + 1$ число
- Применим бинарный поиск по ответу

E. Массив Монокарпа

- Зафиксируем длину отрезка в бинпоиске

E. Массив Монокарпа

- Зафиксируем длину отрезка в бинпоиске
- Переберем удаляемый отрезок чисел

E. Массив Монокарпа

- Зафиксируем длину отрезка в бинпоиске
- Переберем удаляемый отрезок чисел
- Найдем максимум и минимум среди оставшихся чисел независимо слева и справа от отрезка

E. Массив Монокарпа

- Зафиксируем длину отрезка в бинпоиске
- Переберем удаляемый отрезок чисел
- Найдем максимум и минимум среди оставшихся чисел независимо слева и справа от отрезка
- Числа слева образуют префикс, числа справа образуют суффикс

E. Массив Монокарпа

- Зафиксируем длину отрезка в бинпоиске
- Переберем удаляемый отрезок чисел
- Найдем максимум и минимум среди оставшихся чисел независимо слева и справа от отрезка
- Числа слева образуют префикс, числа справа образуют суффикс
- Заранее посчитаем максимум и минимум на префиксе и на суффиксе

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

- Для каждого элемента нужно найти количество больших

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

- Для каждого элемента нужно найти количество больших
- Перебирать за $O(n^2)$ слишком медленно

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

- Для каждого элемента нужно найти количество больших
- Перебирать за $O(n^2)$ слишком медленно
- Отсортируем элементы по убыванию

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

- Для каждого элемента нужно найти количество больших
- Перебирать за $O(n^2)$ слишком медленно
- Отсортируем элементы по убыванию
- В отсортированном порядке каждый элемент либо меньше предыдущего (тогда нужно удалить все предыдущие), либо равен предыдущему (тогда ответ для него такой же, как для предыдущего)
  $[3, 1, 4, 1, 5, 3] \to [5, 4, 3, 3, 1, 1]$

F. Сделай максимальным

Задан массив. Для каждого элемента определить, сколько элементов надо удалить, чтобы он стал максимальным

- Для каждого элемента нужно найти количество больших
- Перебирать за $O(n^2)$ слишком медленно
- Отсортируем элементы по убыванию
- В отсортированном порядке каждый элемент либо меньше предыдущего (тогда нужно удалить все предыдущие), либо равен предыдущему (тогда ответ для него такой же, как для предыдущего)
  $[3, 1, 4, 1, 5, 3] \to [5, 4, 3, 3, 1, 1]$
- Сортировка меняет порядок элементов — будем сортировать пары (a_i, i)

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

- Будет ровно один цвет, в который покрашены две клетки

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

- Будет ровно один цвет, в который покрашены две клетки
- Между этими клетками все цвета должны быть больше по номеру (чтобы можно было перекрасить все внутри отрезка)

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

- Будет ровно один цвет, в который покрашены две клетки
- Между этими клетками все цвета должны быть больше по номеру (чтобы можно было перекрасить все внутри отрезка)
- Снаружи цвета могут идти как угодно

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

- Будет ровно один цвет, в который покрашены две клетки
- Между этими клетками все цвета должны быть больше по номеру (чтобы можно было перекрасить все внутри отрезка)
- Снаружи цвета могут идти как угодно
- Пусть «двойным» будет цвет i, а внутри него будут j цветов. Тогда есть A^j_{n-i-1} способов выбрать цвета и порядок внутри отрезка

G. Количество раскрасок

Есть бесцветная полоска из n клеток. $n-1$ раз красится ее подотрезок (для i-й операции в цвет i). Посчитать количество раскрасок, где все клетки покрашены и все цвета используются

- Будет ровно один цвет, в который покрашены две клетки
- Между этими клетками все цвета должны быть больше по номеру (чтобы можно было перекрасить все внутри отрезка)
- Снаружи цвета могут идти как угодно
- Пусть «двойным» будет цвет i, а внутри него будут j цветов. Тогда есть A^j_{n-i-1} способов выбрать цвета и порядок внутри отрезка
- Оставшиеся цвета можно расставить как угодно снаружи. «Сожмем» отрезок в один цвет и получим, что надо расставить $(n-j)$ цветов — кол-во способов равно $(n-j)!$

G. Количество раскрасок

Это слишком медленно. Как ускорить?

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним
- Так как модуль простой, обратный элемент к x — это $x^{\mathrm{MOD}-2}$

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним
- Так как модуль простой, обратный элемент к x — это $x^{\mathrm{MOD}-2}$
- Бинарным возведением в степень обратный элемент считается за $O(\log \mathrm{MOD})$

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним
- Так как модуль простой, обратный элемент к x — это $x^{\mathrm{MOD}-2}$
- Бинарным возведением в степень обратный элемент считается за $O(\log \mathrm{MOD})$
- Главная проблема: перебирать и «двойной» цвет, и кол-во цветов внутри него долго

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним
- Так как модуль простой, обратный элемент к x — это $x^{\mathrm{MOD}-2}$
- Бинарным возведением в степень обратный элемент считается за $O(\log \mathrm{MOD})$
- Главная проблема: перебирать и «двойной» цвет, и кол-во цветов внутри него долго
- Переберем множество цветов внутри «двойного», **включая** «двойной», и скажем, что «двойной» будет минимальным из этого множества

G. Количество раскрасок

Это слишком медленно. Как ускорить?

- Предподсчитаем все факториалы и обратные элементы к ним
- Так как модуль простой, обратный элемент к x — это $x^{\mathrm{MOD}-2}$
- Бинарным возведением в степень обратный элемент считается за $O(\log \mathrm{MOD})$
- Главная проблема: перебирать и «двойной» цвет, и кол-во цветов внутри него долго
- Переберем множество цветов внутри «двойного», **включая** «двойной», и скажем, что «двойной» будет минимальным из этого множества
- Если у такого множества размер k, то будет C_n^k способов выбрать его, $(k-1)!$ способов расставить цвета внутри отрезка, и $(n-k+1)!$ способов расставить все остальное

H. Разделение на части

Количество способов разрезать строку на кусочки, чтобы в каждом кусочке было два отрезка из одинаковых букв

H. Разделение на части

Количество способов разрезать строку на кусочки, чтобы в каждом кусочке было два отрезка из одинаковых букв

- Выделим блоки одинаковых букв в строке

H. Разделение на части

Количество способов разрезать строку на кусочки, чтобы в каждом кусочке было два отрезка из одинаковых букв

- Выделим блоки одинаковых букв в строке
- Рассмотрим некоторый кусочек $[l, r]$

H. Разделение на части

Количество способов разрезать строку на кусочки, чтобы в каждом кусочке было два отрезка из одинаковых букв

- Выделим блоки одинаковых букв в строке
- Рассмотрим некоторый кусочек $[l, r]$
- Позиция l принадлежит предыдущему блоку от блока с позицией r

H. Разделение на части

- Динамическое программирование

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r
- Для перехода переберем последний кусочек

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r
- Для перехода переберем последний кусочек
- $dp[r] = \sum dp[l]$ по таким l, что $l + 1$ принадлежит предыдущему блоку от r

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r
- Для перехода переберем последний кусочек
- $dp[r] = \sum dp[l]$ по таким l, что $l + 1$ принадлежит предыдущему блоку от r
- Подходящие l образуют отрезок

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r
- Для перехода переберем последний кусочек
- $dp[r] = \sum dp[l]$ по таким l, что $l + 1$ принадлежит предыдущему блоку от r
- Подходящие l образуют отрезок
- Будем поддерживать префиксную сумму по динамике

H. Разделение на части

- Динамическое программирование
- $dp[r]$ — сколько способов разрезать префикс строки длины r
- Для перехода переберем последний кусочек
- $dp[r] = \sum dp[l]$ по таким l, что $l + 1$ принадлежит предыдущему блоку от r
- Подходящие l образуют отрезок
- Будем поддерживать префиксную сумму по динамике
- Ответ лежит в $dp[n]$

H. Разделение на части

- Границы блоков — позиция 1, а также позиции, в которых $s_i \neq s_{i-1}$

H. Разделение на части

- Границы блоков — позиция 1, а также позиции, в которых $s_i \neq s_{i-1}$
- Будем поддерживать две ближайшие слева границы $bd_1 < bd_2$

H. Разделение на части

- Границы блоков — позиция 1, а также позиции, в которых $s_i \neq s_{i-1}$
- Будем поддерживать две ближайшие слева границы $bd_1 < bd_2$
- $dp[r] = \sum_{l=bd_1-1}^{bd_2-1} dp[l]$

H. Разделение на части

- Границы блоков — позиция 1, а также позиции, в которых $s_i \neq s_{i-1}$
- Будем поддерживать две ближайшие слева границы $bd_1 < bd_2$
- $dp[r] = \sum_{l=bd_1-1}^{bd_2-1} dp[l]$
- $dp[r] = pref[bd_2] - pref[bd_1]$

I. Светофор

На светофоре только что загорелся красный цвет. Определить, какой цвет будет через n секунд, если для каждого цвета известно, сколько секунд он горит

I. Светофор

На светофоре только что загорелся красный цвет. Определить, какой цвет будет через n секунд, если для каждого цвета известно, сколько секунд он горит

- Можно «честно» моделировать: поддерживать, сколько секунд до смены цвета, какой цвет сейчас и какой цвет следующий

I. Светофор

На светофоре только что загорелся красный цвет. Определить, какой цвет будет через n секунд, если для каждого цвета известно, сколько секунд он горит

- Можно «честно» моделировать: поддерживать, сколько секунд до смены цвета, какой цвет сейчас и какой цвет следующий
- Если до смены цвета больше n секунд, текущий цвет — и есть ответ

I. Светофор

На светофоре только что загорелся красный цвет. Определить, какой цвет будет через n секунд, если для каждого цвета известно, сколько секунд он горит

- Можно «честно» моделировать: поддерживать, сколько секунд до смены цвета, какой цвет сейчас и какой цвет следующий
- Если до смены цвета больше n секунд, текущий цвет — и есть ответ
- Если ровно n секунд, текущий цвет меняется на следующий в нужную секунду, и они оба — ответ

I. Светофор

На светофоре только что загорелся красный цвет. Определить, какой цвет будет через n секунд, если для каждого цвета известно, сколько секунд он горит

- Можно «честно» моделировать: поддерживать, сколько секунд до смены цвета, какой цвет сейчас и какой цвет следующий
- Если до смены цвета больше n секунд, текущий цвет — и есть ответ
- Если ровно n секунд, текущий цвет меняется на следующий в нужную секунду, и они оба — ответ
- Иначе вычтем из n кол-во времени до смены цвета и поменяем цвет, а также определим, какой будет новый следующий цвет

I. Светофор

Это слишком долго. Как ускорить?

I. Светофор

Это слишком долго. Как ускорить?

- Каждые 4 смены цвета все повторяется

I. Светофор

Это слишком долго. Как ускорить?

- Каждые 4 смены цвета все повторяется
- Можно сразу промоделировать много блоков по 4 смены цвета

I. Светофор

Это слишком долго. Как ускорить?

- Каждые 4 смены цвета все повторяется
- Можно сразу промоделировать много блоков по 4 смены цвета
- Для этого посчитаем длительность одного блока $t = r + g + 2 \cdot y$, а затем возьмем остаток от n по модулю t

I. Светофор

Это слишком долго. Как ускорить?

- Каждые 4 смены цвета все повторяется
- Можно сразу промоделировать много блоков по 4 смены цвета
- Для этого посчитаем длительность одного блока $t = r + g + 2 \cdot y$, а затем возьмем остаток от n по модулю t
- После этого останется промоделировать не более 4 смен цвета

I. Светофор

Это слишком долго. Как ускорить?

- Каждые 4 смены цвета все повторяется
- Можно сразу промоделировать много блоков по 4 смены цвета
- Для этого посчитаем длительность одного блока $t = r + g + 2 \cdot y$, а затем возьмем остаток от n по модулю t
- После этого останется промоделировать не более 4 смен цвета
- Аккуратно со случаем, когда n делится на t!

J. Размещение постройки

Количество способов разместить 2×2 квадратик на поле с некоторыми занятыми клетками

J. Размещение постройки

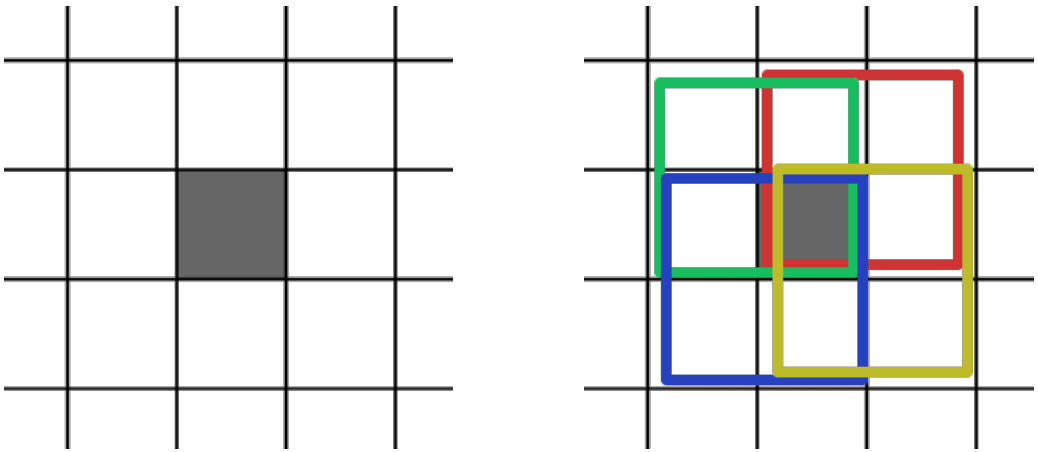
Количество способов разместить 2×2 квадратик на поле с некоторыми занятыми клетками

- Посчитаем количество плохих позиций для квадратика

J. Размещение постройки

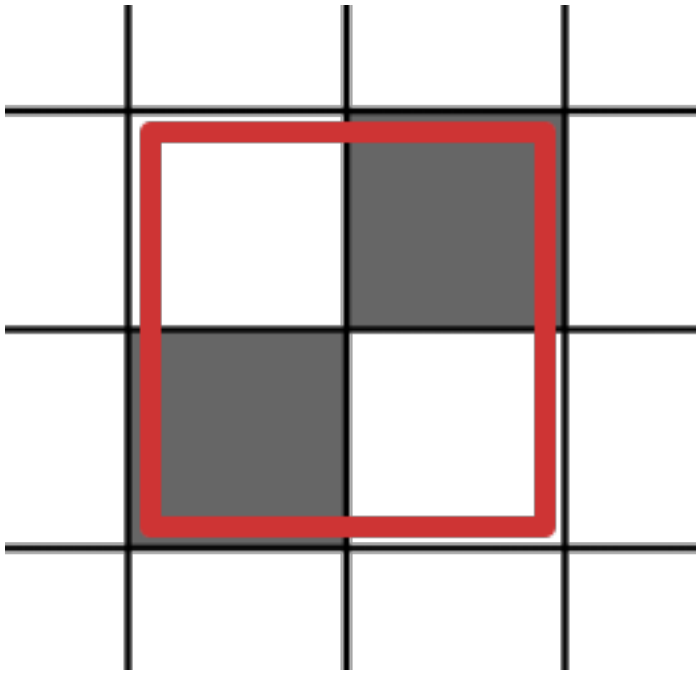
Количество способов разместить 2×2 квадратик на поле с некоторыми занятыми клетками

- Посчитаем количество плохих позиций для квадратика
- Вокруг каждой занятой клетки 4 плохих позиции



J. Размещение постройки

Случайно посчитаем некоторые плохие позиции больше одного раза



J. Размещение постройки

- Скинем все плохие позиции в сет, найдем его размер

J. Размещение постройки

- Скинем все плохие позиции в сет, найдем его размер
- Ответ равен $(n-1) \cdot (m-1) - |bad|$

K. Цикл робота

Количество треугольных маршрутов на поле с некоторыми занятыми клетками

K. Цикл робота

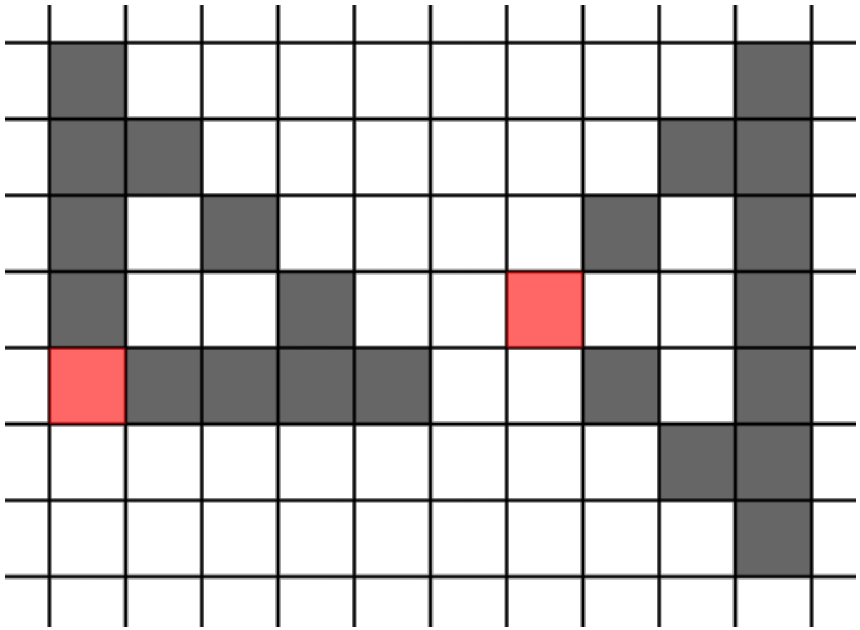
Количество треугольных маршрутов на поле с некоторыми занятыми клетками

- Треугольники равнобедренные с прямым углом

K. Цикл робота

Количество треугольных маршрутов на поле с некоторыми занятыми клетками

- Треугольники равнобедренные с прямым углом
- Два типа треугольников:



K. Цикл робота

- Зафиксируем клетку с прямым углом треугольника

K. Цикл робота

- Зафиксируем клетку с прямым углом треугольника
- Переберем длину стороны

K. Цикл робота

- Зафиксируем клетку с прямым углом треугольника
- Переберем длину стороны
- Длина стороны не превышает n и не превышает m

K. Цикл робота

- Зафиксируем клетку с прямым углом треугольника
- Переберем длину стороны
- Длина стороны не превышает n и не превышает m
- Длина стороны не превышает $\sqrt{nm}$

K. Цикл робота

- Зафиксируем клетку с прямым углом треугольника
- Переберем длину стороны
- Длина стороны не превышает n и не превышает m
- Длина стороны не превышает $\sqrt{nm}$
- Надо проверить $O(nm\sqrt{nm})$ треугольников

K. Цикл робота

- Построим префиксные суммы по каждому столбцу и каждой побочной диагонали

K. Цикл робота

- Построим префиксные суммы по каждому столбцу и каждой побочной диагонали
- Будем перебирать циклом `while`, пока на сторонах при прямом угле нет занятых клеток

K. Цикл робота

- Построим префиксные суммы по каждому столбцу и каждой побочной диагонали
- Будем перебирать циклом `while`, пока на сторонах при прямом угле нет занятых клеток
- Проверим, что на третьей стороне тоже нет занятых клеток, с помощью префиксных сумм

K. Цикл робота

- 8 различных положений треугольника с фиксированным углом

K. Цикл робота

- 8 различных положений треугольника с фиксированным углом
- Можно повернуть поле 4 раза на 90 градусов и считать только 2 положения на каждом повороте

K. Цикл робота

- 8 различных положений треугольника с фиксированным углом
- Можно повернуть поле 4 раза на 90 градусов и считать только 2 положения на каждом повороте
- Не забудем, что каждый треугольник можно обойти 2 способами

L. Третья сторона

Даны длины двух сторон треугольника — a и b. Определить, какой (минимум и максимум) может быть длина третьей, если треугольник ненулевой площади

- Неравенство треугольника: $c < a + b$ (так же с другими сторонами)

L. Третья сторона

Даны длины двух сторон треугольника — a и b. Определить, какой (минимум и максимум) может быть длина третьей, если треугольник ненулевой площади

- Неравенство треугольника: $c < a + b$ (так же с другими сторонами)
- Тогда максимальное значение c равно $a + b - 1$

L. Третья сторона

Даны длины двух сторон треугольника — a и b. Определить, какой (минимум и максимум) может быть длина третьей, если треугольник ненулевой площади

- Неравенство треугольника: $c < a + b$ (так же с другими сторонами)
- Тогда максимальное значение c равно $a + b - 1$
- Для минимального значения c должно выполняться $\max(a, b) < c - \min(a, b)$

L. Третья сторона

Даны длины двух сторон треугольника — a и b. Определить, какой (минимум и максимум) может быть длина третьей, если треугольник ненулевой площади

- Неравенство треугольника: $c < a + b$ (так же с другими сторонами)
- Тогда максимальное значение c равно $a + b - 1$
- Для минимального значения c должно выполняться $\max(a, b) < c - \min(a, b)$
- То есть $c > \max(a, b) - \min(a, b)$

L. Третья сторона

Даны длины двух сторон треугольника — a и b. Определить, какой (минимум и максимум) может быть длина третьей, если треугольник ненулевой площади

- Неравенство треугольника: $c < a + b$ (так же с другими сторонами)
- Тогда максимальное значение c равно $a + b - 1$
- Для минимального значения c должно выполняться $\max(a, b) < c - \min(a, b)$
- То есть $c > \max(a, b) - \min(a, b)$
- Минимальное c равно $\max(a, b) - \min(a, b) + 1$