Разбор задач ИОИП-2021

Подсчет хештегов

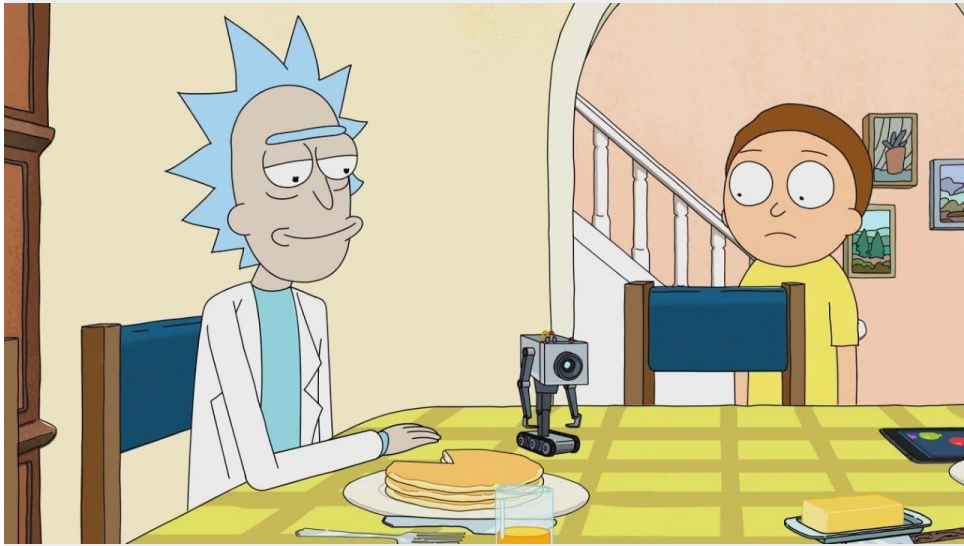
Автор: Николай Будин
Разработчик: Николай Будин

Постановка задачи и решение

- Дан текст
- Надо выделить хештеги, посчитать их количество
- Так как хештег всегда ограничен пробелами, можно считывать слова стандартным методом для вашего языка программирования
  - `cin >> s` в C++
  - `input().split()` в Python
- Для слова проверяем, что оно хештег
  - Начинается с ‘#’
  - Остальные символы не равны ‘#’
- Осталось посчитать количество раз, которое встречается каждый хештег
- Можно отсортировать список хештегов, тогда одинаковые будут идти подряд
- Можно воспользоваться `std::map` в C++ или `dict` в Python

Раскладывание приборов

Автор: Даниил Орешников
Разработчик: Григорий Хлытин



Постановка задачи

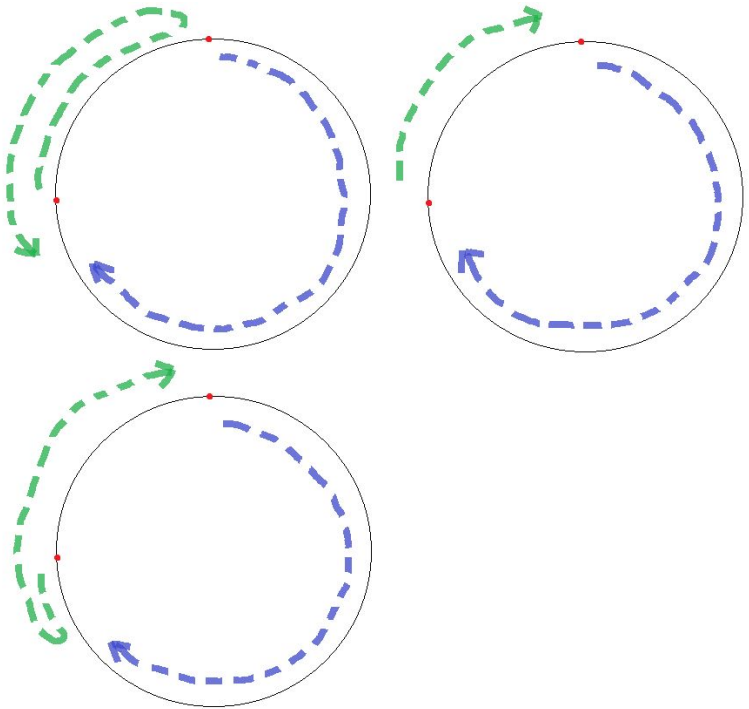
- Есть круглый стол на n мест
- Места пронумерованы натуральными числами
- Возле мест с номерами a и b находятся два официанта
- За одну секунду официант может перейти на соседнее место
- За какое наименьшее количество секунд два официанта суммарно смогут посетить каждое место хотя бы по одному разу?

Решение

- Заметим, что нас не интересуют точные положения официантов, а только количество мест между ними с одной и с другой стороны
- Можно доказать, что есть три уникальных варианта поведения официантов:
  1. Официанты все время идут в одном направлении
  2. Один из официантов идет по малой дуге до конца, разворачивается и идет навстречу другому
  3. Один из официантов идет по большой дуге сколько-то, затем разворачивается и идет до конца малой дуги

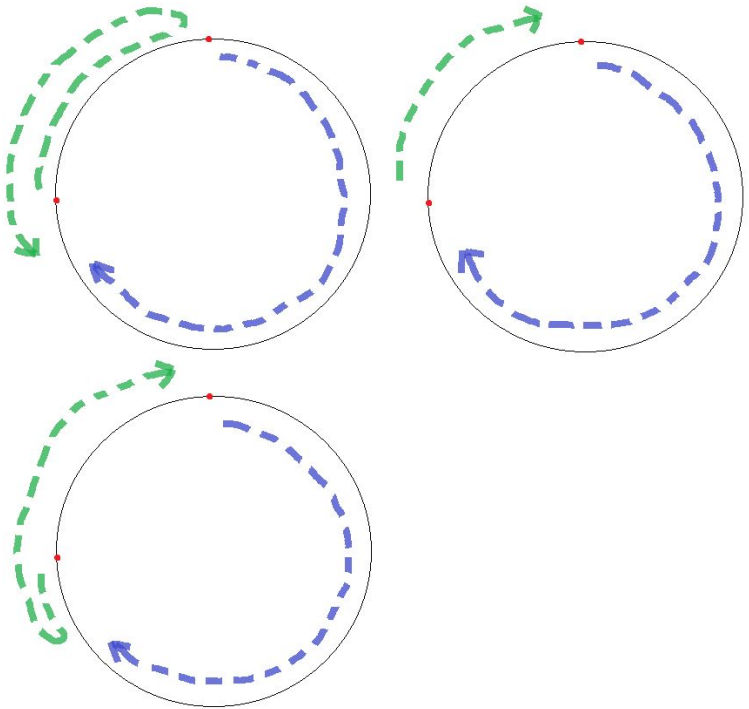
Решение

- Если обозначить за a меньшую дугу (количество мест на ней), то большая равна `n - a - 2`
- При движении в одном направлении расстояние равно большей дуге
- Иначе один из официантов дважды проходит либо меньшую дугу, либо некоторое расстояние на большей



Решение

- Если дважды проходить меньшую дугу, то за время `t` они сумеют посетить `t + 1 + a + 1 + (t - 2a)` мест
  - Отсюда `t ≥ ⌈(n + a - 2) / 2⌉`
- Иначе, формула чуть сложнее, и получится `t ≥ ⌈(2n - 4 - a) / 3⌉`
- В ответ следует взять минимум из трех полученных величин



Шестизначные документы

Автор: Даниил Орешников
Разработчик: Даниил Орешников

Постановка задачи

- Дан массив из шестизначных идентификаторов
- Инверсией в k-м разряде называется пара индексов `i < j`, что k-я цифра в `i`-м идентификаторе строго больше k-й цифры в `j`-м
- Сложностью массива называется общее количество инверсий
- Нужно сделать циклический сдвиг массива, чтобы его сложность была минимальна

Решение

- Для начала, вычислим количество инверсий в исходном массиве
- Независимо по разрядам
- Идем по массиву, поддерживаем `cnt[i][j]` – количество цифр, равных `j`, в `i`-м разряде на текущем префиксе
- К количеству инверсий нужно прибавить количество цифр, больших текущей, которые были раньше в этом разряде

Решение

- Научимся за `O(1)` пересчитывать количество инверсий при циклическом сдвиге на `1`
- Цифра, стоявшая на первом месте, перемещается на последнее место
- Из количества инверсий вычитается количество цифр, строго меньших неё, и прибавляется количество цифр, строго больших неё
- Эти значения посчитаны в `cnt[i][j]`

Государственный переполох

Автор: Даниил Орешников
Разработчик: Даниил Орешников

Постановка задачи

- Есть n городов (столбиков) с разным числом министров (разных высот)
- Можно узнавать сколько столбиков высоты не менее конкретного выбранного
- Можно уменьшать высоту любого столбика на какую-то величину, если его текущая высота не меньше этой величины
- Требуется определить сумму всех высот, уложившись в ограничение на число запросов

План решения

- Базовое решение за nh
- Особые случаи ($n + 50h$, $11n + 2h / 5$)
- Решение за $n + 2h$
- Решение за $n \cdot \log_2 h$
  - Модификация за $2\sum \log_2 a_i$
- Решение за $3n + 4\sqrt{2h}$
  - Модификация за $3n + 4\sqrt{h} + \log_2 h / 2$
- Объединение идей, $3n + (n + 1) \cdot \log_2(2h / n)$

Базовое решение

- Будем пользоваться только запросами уменьшения
- Для каждого столбца:
  - Делаем запросы «`- i 1`» для данного `i`
  - Как только получаем «`FAIL`», знаем, что дошли до нуля

- Стоит обратить внимание, что если уже сделано `h` запросов в один столбец, делать еще один не надо, иначе решение в худшем случае занимает $n \cdot (h + 1)$ запросов

Идея сортировки

- Заметим, что если сделать `n` запросов «`? i`», мы сможем получить полный порядок на высотах столбцов
- Для этого заметим, что
  - одинаковые ответы на такие запросы в разные столбцы равносильны тому, что и столбцы одинаковой высоты
  - более высокий столбец даст меньший ответ на такой запрос
- Сделав `n` запросов, можно упорядочить столбцы по ответам на них, после чего мы получим сортировку столбцов по их высотам

Идея спуска с запросами информации

- Если после сортировки спускаться из максимального столбца с некоторым шагом a, и после каждого запроса уменьшения на a делать запрос на количество, можно узнать высоты всех столбцов с точностью до блоков высоты a
- Если ответ на запрос увеличился на x по отношению с прошлым, мы проскочили ровно x столбцов последним уменьшением высоты
- Дальше эту информацию можно использовать сразу, например, уменьшая пропущенные столбцы, либо сильно позже, собрав больше данных

Тесты с особыми ограничениями

- Тест с ограничением на количество различных столбцов позволял после применения описанной сортировки сделать спуск до нуля за высоту столбца из каждого столбца разной высоты
- Поскольку спуск требует не более `h` операций, решение занимает в сумме `n + 50h` запросов

Тесты с особыми ограничениями

- Если высоты столбцов отличаются хотя бы на 5, можно сделать сортировку и спускаться с шагом 5 из максимального столбца
- Если мы проскакиваем мимо другого столбца, не более чем за 5 пар операций уменьшения на 1 и запроса количества можно будет сравнять пропущенный столбец с текущим
- Когда изначально максимальный столбец дойдет до нуля, мы будем знать высоты всех столбцов
- Требуемое число операций равно $n + 2h/5 + 10n = 11n + 2h/5$

Решение за n + 2h

- Сделаем сортировку
- После сортировки будем спускаться из максимального столбца парами запросов «-1» и «?»
- За счет второго типа запросов, когда высота текущего столбца становится равной какому-то другому, мы это сразу узнаем
- Когда дойдем до нуля, будем знать все высоты, потому что знаем, на какую величину каждый столбец изначально был меньше максимального

Решение за $n \cdot \log_2 h$

- Сделаем независимый двоичный спуск в каждом столбце
- Будем делать запросы спуска на степени двойки, уменьшая с каждым запросом степень на 1
- За счет того, что два уменьшения на одну и ту же степень двойки делать не надо (иначе можно было бы уменьшить на большую степень), всего такой спуск занимает ровно $\log_2 h$ операций в каждом столбце
- Суммарно получаем $n \cdot \log_2 h$ операций

Тесты с ограничением на среднее геометрическое

- Ограничение на среднее геометрическое дает нам ограничение на сумму логарифмов высот столбцов
- Вместо предыдущего решения, которое спускается, начиная с максимальной степени двойки, сделаем в каждом столбце подъем + спуск
- Начнем уменьшение с `1, 2, 4,` и так далее
- Когда уменьшить не получится, сделаем спуск, начиная с той степени, на которой остановились
- Такое решение работает за удвоенную сумму логарифмов высот

Главное решение

- Используем корневую декомпозицию
- Будем после сортировки спускаться из максимального столбца шагами $\sqrt{(h / 2)}$, после каждого делать запрос на количество
- Когда спускаться дальше будет нельзя, сделаем в i-й столбец запрос уменьшения на $x_i \cdot \sqrt{(h / 2)}$, где x_i – сколько шагов было сделано после того, как был пройден столбец i
- После этого размеры всех столбцов не превосходят $2\sqrt{(h / 2)}$
- Отсортируем столбцы заново и применим решение за $n + 2h$
- Суммарное число запросов – $3n + 4\sqrt{(2h)}$

Первая оптимизация

- Аналогично прошлому решению, но выбираем шаг $\sqrt{h}$
- После того, как из изначально максимального столбца нельзя спуститься еще на шаг, за логарифм от размера шага уменьшаем его до нуля двоичным спуском
- Остальные столбцы уменьшаем на пройденное число шагов + ту величину, на которую уменьшили максимальный столбец двоичным спуском
- Теперь высоты всех столбцов не выше $\sqrt{h}$
- Применим решение за n + удвоенная высота, получим $3n + 4\sqrt{h} + \log_2 h / 2$

Вторая оптимизация

- Работает на последних тестах, где n сильно меньше корня высоты
- Воспользуемся предыдущей описанной оптимизацией
- Шаг возьмем $2h \ / \ n$
- Вместо решения за n + удвоенная высота на последнем этапе применим решение с двоичными спусками
- Суммарное число операций будет $3n + (n + 1) \cdot \log_2(2h \ / \ n)$

Общее решение

- Общее решение объединяет описанные выше идеи
- Достаточно было рассмотреть оптимизации решения с корневой декомпозицией, решение за n + 2h, и два решения с двоичными подъемами и спусками
- Неполные баллы можно было набрать и другими решениями, например методом “разделяй и властвуй”

Загадочное устройство

Автор: Даниил Орешников
Разработчик: Николай Будин

Постановка задачи

- Есть n строк w_i, m чисел a_i и изначально пустая строка s
- Можно дописать в конец s любую w_i
- Можно удалить a_i символов из конца s
- Для каждой из q строк t_i сказать, можно ли ее получить

Решение

- Пусть строку t_i получить можно, тогда рассмотрим части строк w_i, из которых она будет состоять
- Несложно заметить, что все эти части являются префиксами w_i
- Обозначим за d НОД всех $|w_i|$ и a_i
- Можно доказать, что от w_i можно оставить те и только те префиксы, длина которых равна $d \cdot k$, где k – целое
- Значит, если $|t_i|$ не делится на d, то её получить нельзя
- Иначе, разобьем все w_i и строку t_i на блоки длины d, будем считать один блок за один символ
- Нужно проверить, можно ли получить строку t_i как конкатенацию префиксов строк w_i

Решение

- Для решения воспользуемся методом динамического программирования
- Пойдем по строке t_i с конца
- `dp[len]` – можно ли разбить суффикс длины `len` строки t_i
- Для перехода заметим, что нас интересует префикс максимальной длины суффикса длины `len`, который равен префиксу какой-то w_i, потому что все меньшие префиксы тоже будут являться префиксами той же w_i
- Можно воспользоваться хешами
- Добавим в `set` хеши всех префиксов всех w_i
- С помощью бинпоиска для длины `len` найдем искомую максимальную длину `x` префикса суффикса длины `len`
- `dp[len]` равно `true`, если среди `dp[len - 1] .. dp[len - x]` есть значение `true`

Дополнительные замечания

- Можно сделать решение без двоичного поиска, однако это не требовалось для решения на полный балл
- Если использовать хеш по модулю порядка 10^9, на последних подгруппах может произойти коллизия