Задача A. Карртавая стррока

Определим строку пустую строку t и булевый флаг f, установленный в False. Пройдемся по строке s слева направо:

- Если $s_i =' r'$, то припишем к строке t букву 'r' и установим флаг $f = True$;
- Иначе если флаг $f = True$ и $i \neq n$, то допишем к строке букву 'r', установим флаг в False и допишем к t букву s_i.

Сложность решения — $O(n)$.

Задача B. АрПрогрессия

Вначале рассмотрим случай, когда $a = b = c$ — в таком случае существует бесконечно много арифметических прогрессий с целочисленной разностью большей 1, в которых встречается, по сути, одно число.

Иначе рассмотрим наибольший общий делитель g попарных разностей чисел, т.е. $a - b$, $a - c$, $b - c$. Если $g = 1$, то не существует ни одной арифметической прогрессии с целочисленной разностью больше 1, что числа a, b, c являются ее членами. Во всех остальных случаях можно найти хотя бы одну такую прогрессию.

Итого сложность — $O(C)$.

Задача C. Наибольшая группа

Можно решить задачу, используя систему непересекающихся множеств (disjoint set union): обозначим каждого студента за отдельную вершину (компоненту). Для каждого языка l найдем список студентов, которые знают язык l — это можно сделать за линейное время, запоминать можно, например, в структуру данных словарь.

Далее для каждого языка l пройдемся по списку студентов, которым он знаком, и выполним операцию объединения множеств студента i и $i + 1$, таким образом мы объединим множества всех студентов, знающих язык l.

Так как объединение выполняется за $O(logn)$, а всего объединений не более $2n$, то итоговая сложность — $O(nlogn)$.

Задача D. Дешевые билеты

Линейным поиском найдем минимальное значение для каждого типа билетов.

Итоговая сложность — $O(n)$.

Задача E. Нельзя проиграть

Задачу можно решить как минимум двумя способами:

Способ 1 - Бинарный поиск + симуляция процесса

Если игрок может пройти игру со значением здоровья hp, то он может пройти её и со значением здоровья $hp + 1$, верно и обратное: если игрок не может пройти игру со значением здоровья hp, то он не сможет ее пройти со здоровьем $hp - 1$.

Описанное позволяет использовать бинарный поиск по ответу для решения этой задачи: для каждого промежуточного значения начального здоровья hp мы наивно симулируем процесс игры, действуем при этом из соображений жадности: если мы можем купить зелье, то покупаем его, причем то, которое наиболее эффективно, и сразу же применяем его. В зависимости от успеха сдвигаем нижнюю или верхнюю границу в бинарном поиске.

Итого: сортировка зелий по их эффективности занимает $O(mlogm)$ — ее мы можем сделать один раз в самом начале. Симуляция процесса игры с начальным здоровьем hp требует $O(n + m)$, бинарный поиск выполняется в границах от 1 до $sum(a) + 1$, что примерно 10^{14} при максимальных значениях a_i — $log(10^{14}) \approx 46$.

Итоговая сложность — $O(46(n + m) + mlogm)$.

Способ 2 - симуляция процесса

Пусть начальное здоровье игрока равно hp, давайте симулировать процесс игры, точно так же жадно. Но в этом решении мы позволяем здоровью героя опускаться до нуля и ниже. В процессе игры найдем минимальное значение здоровья hp_{min}, которое имелось у героя в каком-то моменте. Можно утверждать, что ответ на задачу равен 1 - hp_{min}.

Итоговая сложность — $O(n + m + mlogm)$.

Задача F. Пропускная система

Задача является чисто реализационной. Хранение пропусков можно организовать по ключу id в словаре, если пропуск имеет разрешение на вход, то значение по ключу id равно 1, иначе 0. Заблокированные пропуски можно хранить или в сете, или другом словаре, чтобы сохранять разрешения на вход/выход пропусков.

Итоговая сложность — $O(n)$.

Задача G. Сравнять строки

Заметим, что нет никакого смысла использовать в операциях букву с индексом i, если $s_i = t_i$.

Оптимальным решением является максимальное использование первой операции между буквами i и j при котором $s_i \neq t_i$ и $s_j \neq t_j$, но при этом $s_i = t_j$ и $s_j = t_i$, т.е. за одну операцию мы исправили две ошибки в строке s.

Создадим пустой словарь, в котором будем по ключу хранить пары букв (a, b) и их количества. Пройдемся по строке s и для каждой такой буквы i, что $s_i \neq t_i$, выполним следующее:

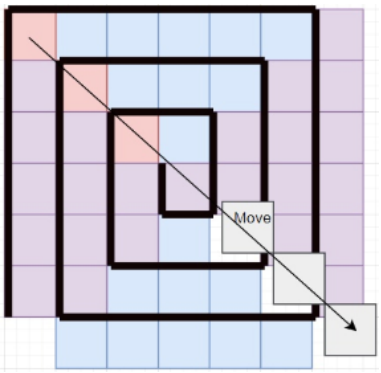
- Если в словаре есть пара (t_i, s_i) и она больше нуля, то уменьшим ее значение на 1;
- Иначе прибавим к ответу 1 и добавим в словарь пару (s_i, t_i) (или увеличим ее значение на 1, если она уже есть в словаре).

Сложность решения — $O(n)$.

Задача H. Корпус 31

Задача может быть решена множеством способов, один из них заключается в том, что для каждого единичного отрезка спирали мы можем сопоставить по одной различной клетке: например, для вертикального отрезка сопоставим клетку справа от отрезка, а для горизонтального отрезка - клетку снизу.

Тогда получим: фиолетовые клетки — принадлежащие отрезкам слева, синии - принадлежащие отрезкам сверху, а красные — это клетки, которые принадлежат одновременно двум отрезкам. Также есть вообще «незанятые» клетки, причем их ровно столько же сколько красных: давайте для каждого любого отрезка из каждой пары отрезков, которым сопоставлена одна и та же клетка, сделаем переназначение клетки на свободную.



Тогда мы получили, что клеток ровно $(n + 1)^2 - 1$ — одна клетка снизу слева не принадлежит никакому отрезку. Так как у нас для каждой клетки сопоставлен теперь только один отрезок, то ответ — это количество клеток.

Сложность — O(1).

Задача I. Люблю плавать

Из соображений жадности для получения максимального числа дней посещения бассейна будем действовать так: если в текущий день можно пойти в бассейн — мы в него идем.

Реализовать идею можно, например, так: заведем счетчик, в котором будем отмечать сколько дней подряд Влад ходил в бассейн до текущего дня. Пойдем по битовой строке слева направо, если символ равен '0' и счетчик меньше 2, то добавляем по единице к ответу и к счетчику, во всех остальных случаях обнуляем счетчик.

Итоговая сложность — $O(n)$.

Задача J. Минимальная последовательность

Будем действовать жадно: пойдем по перестановке слева направо, для каждого индекса i определим какое минимально возможное число может быть под этим индексом среди чисел под индексами i, $i+1$, $i+2$, т.к. мы можем сделать максимум две замены.

Задача участника состоит в грамотной реализации проверки описанной «минимальности» и учета количества использований каждого элемента.

Сложность — $O(n)$.

Задача K. Архивные записи

Ограничения задачи позволяют сделать полный перебор — для каждого i-го рядового сотрудника из c рассмотрим всех d директоров и посчитаем число директоров, с которыми работал сотрудник. Сотрудник i с парой $(a; b)$ работал с директором j с парой $(c; d)$, если $max(a, c) \leqslant min(b, d)$.

Итоговая сложность — $O(ds)$.

Задача L. 55 баллов

Применим для решения идею динамического программирования, а именно идею решения задачи о рюкзаке. Пусть $dp_{(i,j)}$ — количество вариантов выполнения работ среди первых i, при которых мы можем набрать ровно j баллов, в таком случае:

- База динамики: $dp_{(0,0)} = 1$;
- Переход: $dp_{(i,j)} = 1 + dp_{(i-1,j-c_i)}$.

Тогда ответ на задачу — сумма $dp_{(n,j)}$ по всем возможным $j \geqslant 55$.

Если считать динамику в том виде, как она была описана выше, то решение будет работать слишком долго. Так как нам нужно узнать сумму всех $dp_{(n,j)}$ для всех $j \geqslant 55$, то давайте обозначим все такие j за один индекс, например 55, тогда:

- База динамики: $dp_{(0,0)} = 1$;
- Переход для $j < 55$: $dp_{(i,j)} = 1 + dp_{(i-1,j-c_i)}$.
- Переход для $j = 55$: $dp_{(i,55)} = dp_{(i-1,k)} + 2 \times dp_{(i-1,55)}$ для всех таких k, что $k + c_i \geqslant 55$.

Не забываем, что все вычисления также нужно вести по модулю $10^9 + 7$. Итоговая сложность решения — $O(55n)$.

Задача M. Грузовой беспилотник

Задача состоит в реализации условного оператора с проверкой условия $m \leqslant n$ – в таком случае мы выводим «light weight», иначе следует вывести «heavy».

Итоговая сложность — $O(1)$.