ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Кащеев Максим

Академическая группа J3113

Принято

Иван Владимирович Ходненко

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Решить задачу по жадным алгоритмам на leetcode.com. Формулировка выбранной задачи на leetcode.com: У вас есть k списков отсортированных целых чисел в неубывающем порядке. Найдите минимальный диапазон, который включает хотя бы одно число из каждого из k списков.

Мы определяем, что диапазон [a,b] меньше диапазона [c,d], если b−a<d−c, или если b−a=d−c, но a<c.

1. Теоретическая подготовка

Для выполнения задачи необходимо знать отличия жадных алгоритмов, знать основные типы данных, команды и операторы языка C.

1. Реализация

Жадный алгоритм необходим для решения задачи, поскольку он минимизирует диапазон [a, b] на каждом шаге, сохраняя представительство в обрабатываемых структурах всех k списков, также на каждой итерации используются свойства отсортированных данных для обновления искомого диапазона, и при этом имеет сложность O(n \* log k) что делает его практически применимым для реальных задач. Реализация алгоритма происходит через приоритетную очередь, в которую алгоритм складывает элементы и проходится по ним, пока не дойдёт до конца одного из подмассивов и найдёт, таким образом, минимальную метрику, включающую хотя бы один элемент из каждого подмассива.

1. Экспериментальная часть

Подсчёт по памяти.

Для рассмотрения использования памяти введем обозначения – k – количество списов в nums, n – общее количество всех элементов во всех списках.

Для входных данных используется указатель на исходные данные размером O(4n), так как каждый элемент занимает 4 байта

Приоритетная очередь priority\_queue<tuple<int, int, int>> - максимально не может содержать более k элементов по алгоритму, при том что каждый элемент занимает 3х4 = 12 байт, таким образом занимаемая память для очереди O(12k).

Дополнительные переменные занимают:

max\_value, start, end по 4 байта, локальные переменные в цикле – tuple<int, int, int> 12 байт, next\_value – 4 байта.

Итоговая оценка памяти – O(4n + 12k + 28) ≈ O(4n + 12k)

Подсчёт асимптотики.

Цикл инициализации – k итераций по O(log k) добавлений в очередь будет O(k \* log k)

Основной цикл – извлечение минимального элемента O(log k), добавление следующего элемента O(log k), обновление диапазона O(1), при этом списки обрабатываются целиком, таким образом основной цикл имеет сложность O(n \* 2 \* log k) ≈ O(n \* log k)

Общая сложность O(k \* log k + n \* log k) ≈ O(n \* log k)

1. Заключение

В ходе выполнения мною данной лабораторной работы было реализовано решение задачи для поиска минимальной метрики, включающей хотя бы один элемент из каждого подмассива текущего массива. Решение было реализовано при помощи нахождения из каждого из K отсортированных списков по одному элементу с использованием приоритетной очереди. В процессе выполнения задачи алгоритм проходится по каждому полмассива, пока не исчерпает элементы одного из списков. На примере этой задачи можно сделать вывод, что жадные алгоритмы используются для уменьшения асимптотики работы алгоритма при использовании оптимального решения в алгоритме.

1. Приложения

В этом разделе вам необходимо указать всю дополнительную информацию, которая не вошла в основной текст, например полный исходный код программы. Пример:

ПРИЛОЖЕНИЕ А

Листинг кода файла с сайта leetcode.com

class Solution {

public:

    vector<int> smallestRange(vector<vector<int>>& nums) {

        priority\_queue<tuple<int, int, int>, vector<tuple<int, int, int>>, greater<>> min\_heap;

        int max\_value = INT\_MIN;

        int start = INT\_MIN;

        int end = INT\_MAX;

        for (int i = 0; i < nums.size(); i++) {

            min\_heap.push({nums[i][0], i, 0});

            max\_value = max(max\_value, nums[i][0]);

        }

        while (true) {

            auto [min\_value, list\_index, element\_index] = min\_heap.top();

            min\_heap.pop();

            if ((long)max\_value - (long)min\_value < (long)end - (long)start ||

            ((long)max\_value - (long)min\_value == (long)end - (long)start && min\_value < start)) {

                start = min\_value;

                end = max\_value;

            }

            if (element\_index + 1 == nums[list\_index].size()) {

                break;

            }

            int next\_value = nums[list\_index][element\_index + 1];

            min\_heap.push({next\_value, list\_index, element\_index + 1});

            max\_value = max(max\_value, next\_value);

        }

        return {start, end};

    }

};

ПРИЛОЖЕНИЕ B

Скриншот задачи с пройденными тестами с сайта leetcode.com

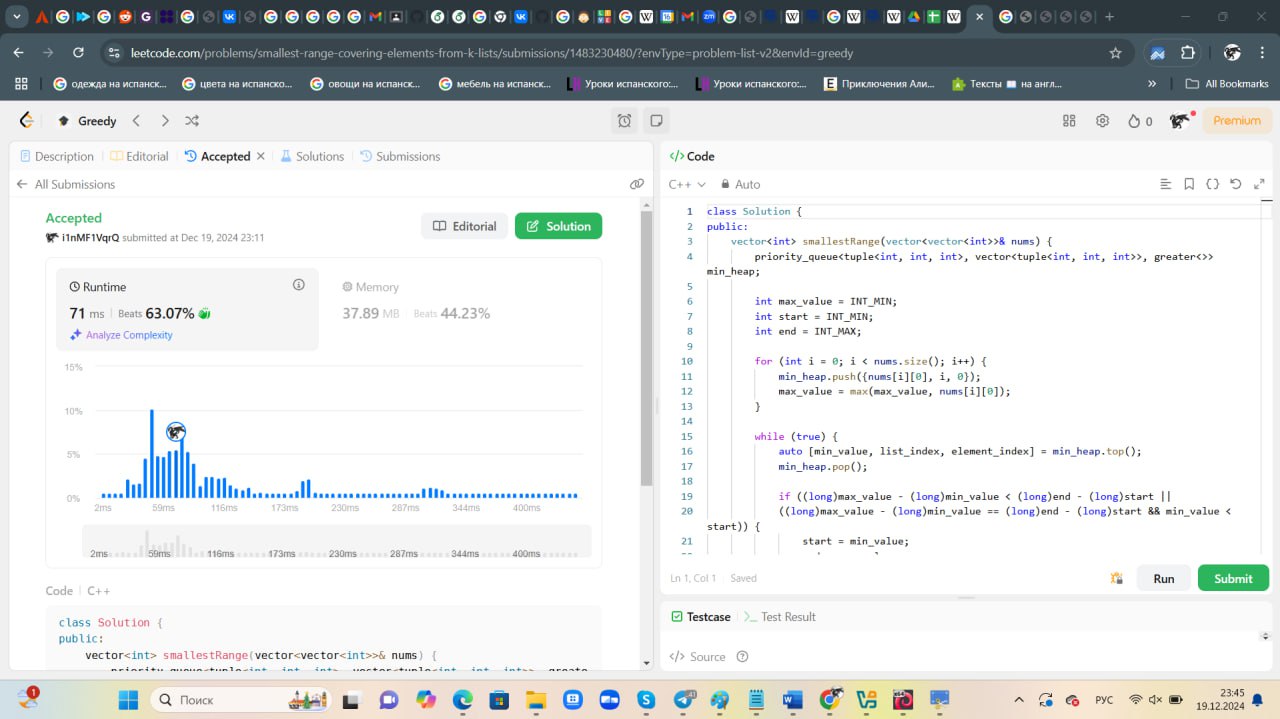


Рисунок 1 – Скриншот выполненной задачи на leetcode.com