ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Баранов Владимир

Академическая группа №J3112

Принято

Дунаев Максим Владимирович

Санкт-Петербург

2024

ВВЕДЕНИЕ

Цель работы: исследовать и реализовать жадный алгоритм для решения задачи оптимального выбора проектов с учетом ограничений по начальному капиталу и количеству допустимых проектов.

Задачи:

- Проанализировать формулировку задачи, описывающей выбор ограниченного числа проектов (не более k) при заданных начальном капитале и требованиях к минимальному капиталу для каждого проекта;

- Обосновать применение жадного подхода к задаче: на каждом шаге из всех доступных проектов с учётом текущего капитала выбирать проект с наибольшей прибылью, тем самым постепенно увеличивая капитал и расширяя доступ к более «дорогим» и прибыльным проектам;

- Реализовать описанный жадный алгоритм на языке C++ и провести тестирование на контрольных примерах, чтобы убедиться в корректности и эффективности подхода.

ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

Жадный подход основан на идее, что локально оптимальный выбор на каждом шаге ведёт к оптимальному глобальному решению или, по крайней мере, обеспечивает эффективное практическое решение. Суть применения жадного алгоритма на моей задаче такова:

1. На каждом шаге выбирать доступный проект с наибольшей прибылью.
2. "Доступность" проекта определяется тем, что его capital[i] не превышает текущий капитал w.

Чтобы эффективно выбирать проекты, необходимо:

- Сначала отсортировать список проектов по значению capital, чтобы можно было упорядоченно получать проекты, которые становятся доступными при росте капитала. Таким образом, при увеличении капитала мы можем последовательно "открывать" проекты с соответствующей планкой входного капитала.

- Хранить набор доступных проектов по прибыли в структуре, из которой легко извлечь максимальный элемент. Оптимальным выбором является приоритетная очередь, позволяющая вставлять элементы за O(log n) и извлекать максимум также за O(log n). Приоритетная очередь – структура данных, поддерживающая операции добавления элемента и выборки максимума за логарифмическое время.

Жадный подход эффективен здесь, поскольку, увеличивая капитал наиболее быстрым способом (через самый прибыльный доступный проект), мы максимизируем шансы получить доступ к ещё более прибыльным проектам позже. В отличие от полного перебора, жадный алгоритм не пытается проверить все комбинации проектов, что было бы слишком затратно по времени, а динамическое программирование также может потребовать слишком больших ресурсов.

РЕАЛИЗАЦИЯ

На входе мы имеем два массива profits и capital, а также параметры k (количество проектов, которое можно выполнить) и w (начальный капитал). Необходимо было подготовить их к дальнейшей сортировке. Вместо создания дополнительного массива пар, для экономии памяти был использован массив индексов.

Для сортировки массива индексов использовалась встроенная функция sort с использованием лямбда функции для того, чтобы проекты располагались по возрастанию значения capital[indx[i]].

Для реализации жадного алгоритма была использована приоритетная очередь priority_queue<int> для хранения прибыльных проектов. На каждом шаге мы:

1. Добавляем в приоритетную очередь все проекты, чей capital[i] не превышает текущий капитал w.
2. Извлекаем из очереди проект с максимальным profit.

Это позволяет на каждом шаге выбирать самый выгодный доступный проект, тем самым быстро наращивая капитал.

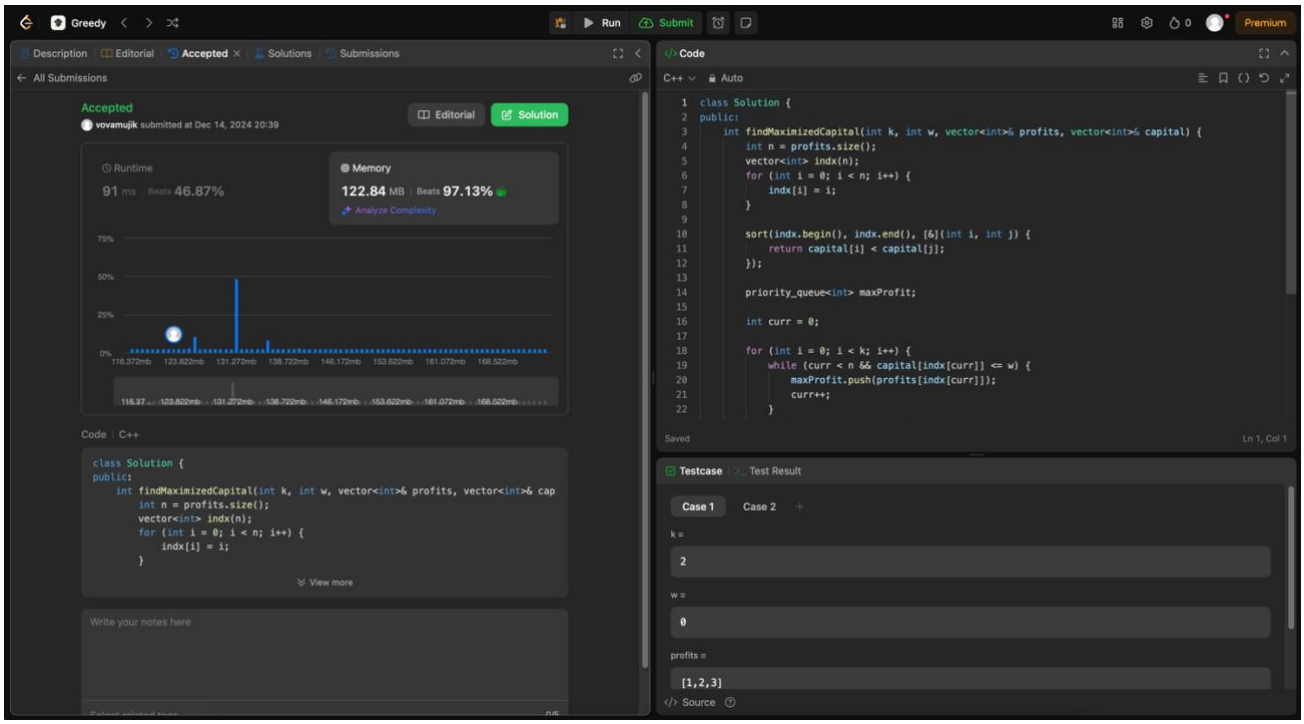
ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Подсчет по памяти:

Исходно имеется два массива profits и capitals каждый по n значений – итого O(2*n). Массив индексов занимает O(n). Приоритетная очередь в худшем случае занимает также O(n). Суммарно выходит O(4*n).

Подсчет сложности алгоритма:

Сортировка вектора индексов размера n с использованием std::sort, который имеет среднюю вычислительную сложность O(n log n). Основной цикл по выбору проектов выполняется максимум k раз. Внутри цикла: часть while (curr < n && capital[indx[curr]] <= w) происходит в сумме для всех итераций не более n раз, поскольку curr каждый раз увеличивается. Каждая вставка в priority_queue (maxProfit.push(...)) — операция со сложностью O(log n), так как приоритетная очередь устроена на базе кучи. Аналогично, извлечение максимума (maxProfit.pop()) также занимает O(log n). Итого операций над приоритетной очередью имеют сложность O(n*log(n)), а сложность сортировки также O(n*log(n)), то итоговая асимптотика алгоритма будет O(n*log(n)).



ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута поставленная цель – разработан и реализован жадный алгоритм для выбора до k проектов с максимизацией итогового капитала при заданном начальном капитале. Полученное решение соответствует теоретическим оценкам по времени (O(n log n)) и памяти (O(n)) и успешно прошло проверку на тестовых примерах, продемонстрировав корректность и эффективность.

ПРИЛОЖЕНИЕ

Листинг кода

```
class Solution {
public:
    int findMaximizedCapital(int k, int w, vector<int>& profits, vector<int>& capital) {
        int n = profits.size();
        vector<int> indx(n);
        for (int i = 0; i < n; i++) {
            indx[i] = i;
        }

        sort(indx.begin(), indx.end(), [&](int i, int j) {
            return capital[i] < capital[j];
        });

        priority_queue<int> maxProfit;

        int curr = 0;

        for (int i = 0; i < k; i++) {
            while (curr < n && capital[indx[curr]] <= w) {
                maxProfit.push(profits[indx[curr]]);
                curr++;
            }
            if (maxProfit.empty()) {
                break;
            }
            w += maxProfit.top();
            maxProfit.pop();

        }
        return w;
    }
};
```