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

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнил работу

Мирасов Константин

Академическая группа J3112

Принято

Ассистент, Дунаев Максим

Санкт-Петербург

2024

Введение

Цель и задача:

изучить применение жадных алгоритмов на примере задачи о максимизации капитала перед IPO. Задача требует выбора проектов из списка так, чтобы максимизировать общий капитал, который компания сможет накопить, учитывая ограничения на количество проектов и начальный капитал. Основная задача — применить эффективный алгоритм, который оптимально решает поставленную задачу в условиях ограниченности ресурсов.

Теоретическая подготовка

Имеется n проектов, каждый из которых характеризуется:

- profits[i] — чистая прибыль от реализации проекта i;
- capital[i] — минимальный размер капитала, необходимый для начала проекта i.

Изначально компания имеет капитал w. Нужно выбрать не более k проектов, чтобы максимизировать итоговый капитал. При выполнении проекта, чистая прибыль добавляется к общему капиталу.

Почему здесь используются жадные алгоритмы?

Жадные алгоритмы характеризуются следующим подходом: на каждом этапе выбирается наиболее оптимальное решение, которое представляется лучшим в данный момент времени, с целью получения глобально оптимального результата.

В данной задаче этот подход оправдан, так как на каждом шаге мы:

- Выбираем проект с максимальной прибылью среди доступных по текущему капиталу.
- Увеличиваем капитал за счет выполнения этого проекта.

Жадный алгоритм позволяет в кратчайшие сроки и без избыточных вычислений достичь максимального результата. Для сравнения, методы полного перебора (brute force) или динамического программирования были бы значительно менее эффективными из-за необходимости проверки всех комбинаций проектов.

Эффективность жадного подхода

Жадный алгоритм в данной задаче имеет следующие преимущества:

- **Скорость**: вместо проверки всех возможных комбинаций (что имеет экспоненциальную сложность) мы фокусируемся на сортировке и выборке.
- **Простота реализации**: идея локального выбора максимального значения легко воплощается.

Эффективность достигается за счет использования структур данных, таких как очередь с приоритетами (priority queue), которая позволяет быстро получать и удалять элемент с максимальным приоритетом.

Практическая часть

Алгоритм решения задачи

1. **Подготовка данных**:
   - Создаем список проектов, где каждый проект представлен парой (capital[i], profits[i]).
   - Сортируем этот список по минимальному капиталу (capital[i]) в порядке возрастания.
1. **Инициализация структуры данных**:
   - Используем очередь с приоритетами (max-heap) для хранения доступных проектов, чтобы быстро выбирать проект с максимальной прибылью.
1. **Выбор проектов**:
   - На каждом из k шагов:
   - Добавляем в очередь все проекты, которые можно выполнить с текущим капиталом.
   - Если очередь не пуста, выбираем проект с наибольшей прибылью, увеличиваем капитал и удаляем проект из очереди.
   - Если очередь пуста, процесс завершается, так как нет доступных проектов для выполнения.
1. **Возврат результата**: возвращаем итоговый капитал.

Код алгоритма

```
class Solution {
public:
    int findMaximizedCapital(int k, int w, vector<int>& profits, vector<int>& capital) {
        vector<pair<int, int>> projects;
        for (int i = 0; i < capital.size(); i++) {
            projects.push_back({capital[i], profits[i]});
        }
        sort(projects.begin(), projects.end());
        priority_queue<int> maxProfit;
        int i = 0;
        for (int j = 0; j < k; j++) {
            while (i < projects.size() && projects[i].first <= w) {
                maxProfit.push(projects[i].second);
                i++;
            }
            if (!maxProfit.empty()) {
                w += maxProfit.top();
                maxProfit.pop();
            }
        }
        return w;
    }
};
```

Анализ сложности

Анализ сложности

Временная сложность

1. **Сортировка списка проектов**:
   - Сложность сортировки O(n * log n), где n — количество проектов.
1. **Итерация по проектам и управление очередью**:
   - Каждый проект добавляется в очередь не более одного раза, а операция добавления/удаления в очереди имеет сложность O(log m), где m — текущий размер очереди.
   - В худшем случае, если все n проектов проходят через очередь, сложность будет O(n * log n).
1. **Цикл по k проектам**:
   - На каждом шаге мы добавляем проекты в очередь и извлекаем максимальный элемент, что также входит в O(n * log n).

Итоговая временная сложность: **O(n \log n + k \log n)**.

Пространственная сложность

1. **Хранение данных**:
   - Массив для хранения пар (capital[i], profits[i]) требует O(n) памяти.
   - Очередь с приоритетами для хранения прибылей в худшем случае содержит O(n) элементов.

Итоговая пространственная сложность: **O(n)**.

Вывод

В ходе лабораторной работы был реализован жадный алгоритм для задачи максимизации капитала перед IPO. Основной подход состоял в сортировке проектов по необходимому капиталу и последовательном выборе проектов с максимальной прибылью, доступных на каждом этапе. Жадный алгоритм показал свою эффективность, обеспечив оптимальный баланс между скоростью выполнения и использованием памяти.

Анализ сложности подтвердил, что предложенный алгоритм значительно превосходит по производительности методы полного перебора, особенно при большом числе проектов. Таким образом, жадный подход является оправданным выбором для решения подобных задач.