Лабораторная работа № 7 по курсу Дискретный Анализ. Динамическое программирование

Выполнил студент группы 08-307 МАИ *Павлов Иван.*

Условие

Кратко описывается задача:

> Вариант 1. Хитрый рюкзак.
>
> Требуется разработать программу, осуществляющую ввод двух числе n - количества предметов и m - вместимость рюкзака. Необходимо найти такое подмножество I из предметов, такое что:
>
> (a) $\sum_{i \in I} w_i \leq m$
>
> (b) $(\sum_{i \in I} c_i \cdot |I|)$
>
> является максимальным, где $|I|$ - мощность множества.

Описание алгоритма

Алгоритмы, предназначенные для решения задач оптимизации, обычно представляют собой последовательность шагов, на каждом из которых предоставляется некоторое множество выборов. В динамическом программировании исходная задача делится на связанные подзадачи и, помня результаты всех решенных подзадач на прошлом шаге, определяем наилучшее решение на текущем.

Данная задача не является классической задачей о рюкзаке, где надо максимизировать суммарную стоимость. Вместо этого надо максимизировать число, равное этой суммарной стоимости умноженной на мощность множества взятых предметов.

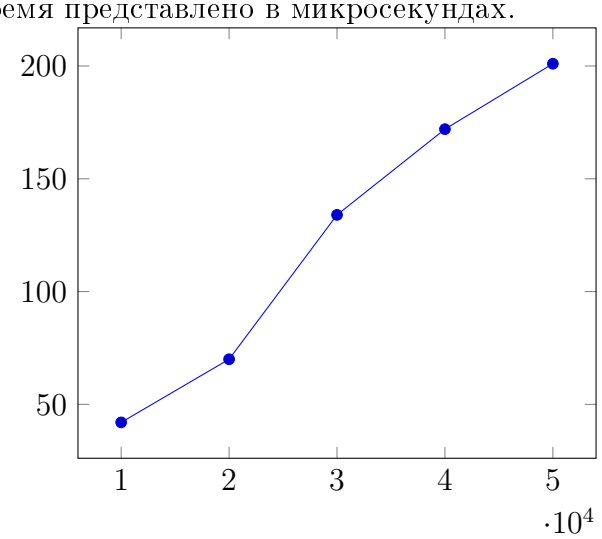
Описание программы

Будем хранить элементы в матрице размерами (вместимость рюкзака)X(индекс предмета)X(мощность подмножества). Соответственно, обойдем этот трехмерный тензор тремя циклами, и на каждом шаге пытаемся положить вещь в рюкзак. Если мощность выбранного подмножества превосходит количество оставшихся элементов, то такое количество элементов выбрать невозможно, также как и невозможно засунуть в рюкзак предмет, превосходящий вместимость рюкзака на определенном шаге. Иначе смотрим, что выгоднее: брать или не брать предмет. Это решение запоминаем, чтобы использовать на следующем шаге. Для оптимизации по памяти будем хранить в векторе только текущее и предыдущее значения, так как на каждом шаге ДП проверяет именно предыдущее.

После этого проходим по всем возможным мощностям подмножеств и выбираем максимально возможное значение произведения мощности и стоимости. После этого выводим решение, используя указатели на предыдущий объект внутри структуры.

Тест производительности

Оценим сложность данного алгоритма, проведя тест на производительность. Предоставляю график зависимости времени работы алгоритма от значения n (в тестах $m = n$). Время представлено в микросекундах.



Сложность алгоритма равна $O(n^2 * m)$, так как мы 2 раза перебираем n шагов (количества и мощности) и 1 раз перебираем вместимости m.

Выводы

Динамическое программирование - это метод решения сложных задач путем разбиения их на более простые подзадачи и последующего комбинирования их решений. С помощью этого метода можно перейти от экспоненциальной к полиномиальной сложности, за счет использования дополнительной памяти, что обычно не так критично по сравнению с временным выигрышем.