Лабораторная работа № 7 по курсу дискретного анализа: динамическое программирование

Выполнил студент группы М8о-207Б-18 МАИ *Токарев Никита*.

Условие

При помощи метода динамического программирования разработать алгоритм решения задачи, определяемой своим вариантом; оценить время выполнения алгоритма и объем затрачиваемой оперативной памяти. Перед выполнением задания необходимо обосновать применимость метода динамического программирования.Разработать программу на языке C или C++, реализующую построенный алгоритм.
Вариант:задана матрица натуральных чисел A размерности n m. Из текущей клетки можно перейти в любую из 3-х соседних, стоящих в строке с номером на единицу больше, при этом за каждый проход через клетку (i, j) взымается штраф A(i,j). Необходимо пройти из какой-нибудь клетки верхней строки до любой клетки нижней, набрав при проходе по клеткам минимальный штраф.

Метод решения

Для реализации поставленной задачи мне понадобится дополнительная структура, куда я буду записывать промежуточные данные.

1. Создам структуру для хранения промежуточных данных.

2. Проходя по каждому элементу буду определять минимальное значение штрафа для данного элемента, исходя из предыдущих вычислений.

3. Еще одним проходом выведу в консоль оптимальное решение.

Таким образом с помощью своего алгоритма я буду считать оптимальный результат для ячеек массива, исходя из результов подсчета для предыдущих ячеек. Я буду считать сумму идя от конца данного массива к его началу: данное решение значительно упростит мне вывод оптимального решения. Идя сверху вниз, к текущему элементу B(i,j) = A(i,j) я прибавляю минимальное значение штрафа, выбирая из трех уже подсчитанных значений min(B(i + 1,j - 1), B(i + 1,j), B(i + 1, j + 1)). Сложность подсчета O(n*n),а сложность вывода оптимального решения O(2n).

Описание программы

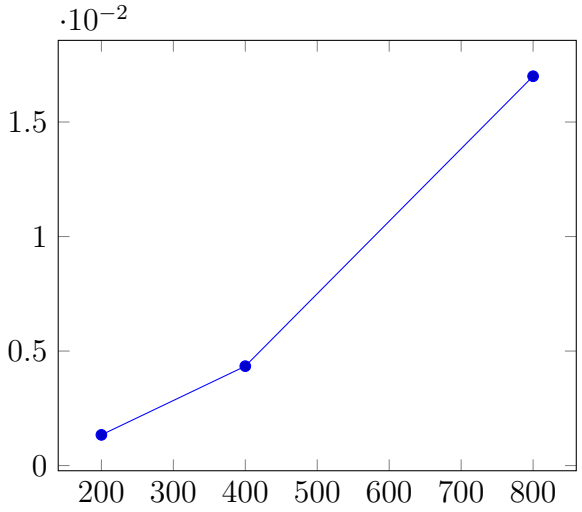
Для начала я инициализировал 5 переменных типа int, а также с помощью вызова new создал две матрицы, которые содержат элементы типа long long. Матрица A служит для хранения входной матрицы, а в матрице B записывается результат вычислений. В данной программе используется 20 байт под статические переменные, а также в ходе

выполнения работы выделяется память, равная 8 * 2 * n байт, где n - размер квадратной матрицы. Пример: для матрицы 20 на 20 будет выделяться 8 * 2 * 20 байт. Также реализована функция поиска минимального значения среди трех данных.
Подсчет значений штрафов:
B[i][j] = (Minimum(B[i + 1][j - 1], B[i + 1][j], B[i + 1][j + 1]) + A[i][j]);
Также хотелось бы отметить, что для элементов стоящих вначале или в конце строки массива, миниму считается исходя из двух значений. А начальные значения матрицы B, от которых начинается отсчет копируются из матрицы A. Оптимальное решение соответсвует спуску по полученной матрице B, проходя только минимальные значения.

Дневник отладки

1. Неправильный ответ в 11 тесте:происходило перевыполнение, так как числа из входных данных превышали тип int

Тест производительности



Пояснения к графику: Ось y - время в секундах. Ось x - размер квадратной мартицы.

Выводы

Метод разбития задач на подзадачи в рекурсивном порядке является довольно эффективным иснтрументом для решения некоторых задач. С помощью динамического программирования можно решить некоторые задачи иногда уменьшив асимптотическую сложность. Также с помощью динамического программирования можно заметно модифицировать алгоритм или программу. Свою задачу я реализовал за сложность O(n*n + 2n). Как мне кажется решить мою задачу не используя методы динамического программирования является довольно сложной задачей. Возможно, каким-либо перебором элементов можно получить решение, однако это точно будет менее эффективно.