Лабораторная работа № 7 по курсу дискретного анализа: динамическое программирование

Выполнил студент группы 08-207 МАИ *Хренов Геннадий.*

Условие

1. 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.

2. Разработать программу на языке C или C++, реализующую построенный алгоритм. Формат входных и выходных данных описан в варианте задания:

3. 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 Ai,j. Необходимо пройти из какой-нибудь клетки верхней строки до любой клетки нижней, набрав при проходе по клеткам минимальный штраф.

Метод решения

Для решения задачи создаем новую матрицу тех же размеров, в которой будем хранить длинну кратчайшего пути до каждого элемента. Это можно сделать путем прибавления к значению ячейки минимального из 3 соседей сверху. После построения матрицы минимальное значения в последний строке матрицы и есть минимально затратный путь. Далее для нахождения пути проделываем обратные операции(вычитания), и поднимаемся снизу вверх.

Описание программы

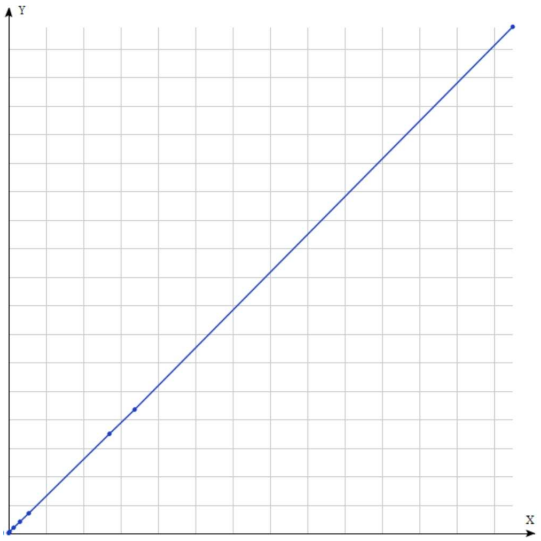
Программа состоит из файла lab7.2.cpp
Основные функции:
Triplemin - минимальный элемент из трех

Дневник отладки

1 - ошибка компиляции - запрещается создавать матрицу на переменных m и n, использовал вектор.
2 - неправильный ответ - изначально считал, что у крайних элементов по 3 соседа, исправил на двух.
3 - неправильный ответ - переполнения инта, заменил на long long.

Тест производительности

размер матрицы(m*n); время(с)
(100; 0,004)
(10000; 0,012)
(50000; 0,029)
(250000; 0,1)
(562500; 0,203)
(1000000; 0,352)
(5000000; 1,745)
(6250000; 2,171)
(25000000; 8,886)



Недочёты

Из-за представления матрицы в линейном векторе появилось некоторая громозкость с доступом к элементу, но зато нет лишних затрат по памяти.

Выводы

Идея динамического программирования в разделении задачи на более простые подзадачи, причем каждую такую подзадачу нужно решить один раз, сократив время вычислений в сравнении с наивными методами. Динамическое программирование находит применение тогда, когда подзадачи перекрываются и количество решений задачи становится крайне велико.