Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Дискретный анализ»

Студент: Т. Д. Голубев
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №7

Задача: Задана матрица натуральных чисел A размерности $n \times m$. 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.

1 Описание

Требуется решить задачу методом динамического программирования.

В [1] сказано: «Динамическое программирование, как и метод "разделяй и властвуй позволяет решать задачи, комбинируя решения вспомогательных подзадач». Соответственно, требуется разбить задачу на подзадачи, из которых мы будем формировать решение.

Я предлагаю создать дополнительную матрицу dp размера $n \times m$, в которой на позиции (i, j) будет находится минимально возможный штраф, с которым можно прийти в клетку (i, j) матрицы A. Тогда ответ (минимальный штраф) будет вычисляться как минимум из последней строки матрицы dp.

Значение элемента с индексами i, j в матрице dp вычисляется как $min(dp_{i-1,j-1}, dp_{i-1,j}, dp_{i-1,j+1}) + A_{i,j}$.

Сложность данного решения – $O(n \cdot m)$, где n – количество строк в матрице, m – количество столбцов.

2 Исходный код

main.cpp	
uint16_t Min(const std::vector<int64_t>& vec)	Поиск минимума в векторе.
uint16_t FindMinFareIndex(uint16_t n, uint16_t m, const std::vector<std::vector<int64_t»& dp, uint16_t i, uint16_t j)	Поиск минимального штрафа среди верхних соседей.
std::vector<Point> FindPath(uint16_t n, uint16_t m, const std::vector<std::vector<int64_t»& dp)	Поиск пути с минимальным штрафом.
std::pair<int64_t, std::vector<Point» Solve(uint16_t n, uint16_t m, const std::vector<std::vector<int64_t»& matrix)	Функция решения задачи.

```
1 ‖ using Point = std::pair<uint16_t, uint16_t>;
```

3 Консоль

```
cat-mood@nuclear-box:~/programming/mai-da-labs/lab07/build$ ./lab07_exe
3 3
3 1 2
7 4 5
8 6 3
8
(1,2) (2,2) (3,3)
```

4 Тест производительности

Тест производительности представляет из себя следующее: Решение с использованием динамического программирования сравнивается с наивным обходом всех решений на матрице 10×10.

```
cat-mood@nuclear-box:~/programming/mai-da-labs/lab07/build$ ./benchmark <test1.txt
naive's time: 8573ms
my solution's time: 53ms

cat-mood@nuclear-box:~/programming/mai-da-labs/lab07/build$ ./benchmark <test2.txt
naive's time: 8662ms
my solution's time: 54ms

cat-mood@nuclear-box:~/programming/mai-da-labs/lab07/build$ ./benchmark <test3.txt
naive's time: 8303ms
my solution's time: 52ms
```

Как видно, моё решение работает быстрее.

5 Выводы

Выполнив седьмую лабораторную работу по курсу «Дискретный анализ», я решил задачу с помощью динамического программирования. В ходе работы столкнулся с проблемой разбиения основной задачи на подзадачи и изначально пошёл неверным путём. Впоследствии переосмыслил задачу и решил верно.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ. - 3-е изд. - М.: ООО "И. Д. Вильямс 2013. - 1328 с.