МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра програмної інженерії та інформаційних технологій управління

Звіт з лабораторної роботи № 8
з дисципліни «Основи теорії алгоритмів»

Виконав:
ст. гр. КН-221в
Шулюпов Є.Р.
Перевірила:
Доцент каф.ПІТУ
Солонська С.В.

Харків
2022

ТЕМА: ДИНАМІЧНЕ ПРОГРАМУВАННЯ

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

Пошук оптимального способу множення матриць. Вхідні дані: натуральне число N ($1 < N < 256$) — кількість матриць, натуральні числа $x_i, i = \overline{0, N}$- розмірності матриць (матриця $B_i, i = \overline{1, N}$ має розмірність $x_{i-1} \times x_i$). Вихідні дані: таблиця динамічного програмування ($A(i, j)$ = найменша кількість множень для обчислення добутку матриць $B_i B_{i+1} \dots B_{j-1} B_j$) та оптимальна розстановка дужок у виразі $A_1 A_2 \dots A_N$.

МЕТА РОБОТИ

Ознайомлення з використанням динамічного програмування та оцінювання його складності.

1 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Термін динамічне програмування був запроваджений в 40-х роках Річардом Беллманом для характеристики процесу розв'язування проблем, при якому потрібно знаходити найкращі рішення, одне за одним. Пізніше, в 1953 році, він уточнив його в сучасному розумінні, називаючи так задачі, безпосередньо пов'язані з розв'язуванням вкладених підзадач для пошуку розв'язку всієї задачі[1] і ця сфера була пізніше визнана IEEE як підрозділ системного аналізу та інженерії. Відзначивши внесок Беллмана, його ім'ям назвали рівняння Беллмана — основну формулу динамічного програмування, яка інтерпретує задачу оптимізації в рекурсивній формі.

Слово динамічне було обране Беллманом, тому що звучало більш переконливо і краще підходило для передачі того факту, що проблема оптимального управління, яку він розв'язував цим методом, має аспект залежності від часу[2]. Слово програмування в цьому словосполученні в дійсності до «традиційного» програмування (написання тексту

програм) майже ніякого відношення не має. Це використання таке саме як і в словосполученнях лінійне програмування та математичне програмування, які фактично є синонімами для математичної оптимізації[3]. Тут воно означає оптимальну послідовність дій, оптимальну програму для отримання розв'язку задачі. Наприклад, певний розклад подій на виставці чи в театрі теж називають програмою. Програма в даному випадку розуміється як запланована послідовність подій. Хоча, динамічне програмування, як алгоритм, часто використовується при програмуванні для розв'язку відповідних задач (див. нижче).

Динамічне програмування зазвичай застосовується до завдань, в яких шукана відповідь складається з частин, кожна з яких в свою чергу дає оптимальне рішення деякої підзадачі. Динамічне програмування корисне, якщо на різних шляхах багаторазово зустрічаються одні й ті ж підзадачі; основний технічний прийом - запам'ятовувати рішення підзадач, що зустрічаються, на випадок, якщо така ж підзадача зустрінеться знову. У цьому випадку алгоритм типу «розділяй і володарюй» буде робити зайву роботу, вирішуючи одні й ті ж підпідзадачі кілька разів. Алгоритм, заснований на динамічному програмуванні, вирішує кожну з підзадач лише раз і запам'ятовує відповіді в спеціальній таблиці. Це дозволяє не обчислювати відповідь знову до вже розглянутої підзадачі.

2 ОПИСАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Ця програма реалізована в одному файлі FirstClass.java

```
package first;

import java.util.Scanner;

public class FirstClass {
        static char name;
        static void printParenthesis(int i, int j, int n, int[][] bracket) {
              if (i == j){
                    System.out.print(name++);
                    return;
              }
              System.out.print("(");
              printParenthesis(i, bracket[i][j], n, bracket);
              printParenthesis(bracket[i][j] + 1, j, n, bracket);
```

```
      System.out.print(")");
}
static void matrixChainOrder(int p[], int n){
    int[][] m = new int[n][n];
    int[][] bracket = new int[n][n];
    for (int i = 1; i < n; i++) {
        m[0][i] = i;
        m[i][0] = i;
    }
    for (int i = 1; i < n; i++)
        m[i][i] = 0;
    for (int L = 2; L < n; L++)
    {
        for (int i = 1; i < n - L + 1; i++)
        {
            int j = i + L - 1;
            m[i][j] = Integer.MAX_VALUE;
            for (int k = i; k <= j - 1; k++)
                {
                 int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                 if (q < m[i][j])
                 {
                     m[i][j] = q;
                     bracket[i][j] = k;
                 }
                }
        }
    }
    name = 'A';
    System.out.print("\nОптимальна таблиця: \n");
    for(int count1 = 0; count1 < n; count1++) {
      for(int count2 = 0; count2 < n; count2++) {
            System.out.print(m[count1][count2]);
            System.out.print("\t");
      }
      System.out.print("\n");
   }
    System.out.print("Оптимальне розставлення дужок: ");
    printParenthesis(1, n - 1, n, bracket);

}
public static void main(String[] args){
      Scanner scanner = new Scanner(System.in);
      char name2 = 'A';
    System.out.println("Уведіть n - кількість матриць");
    int s = scanner.nextInt();
    int arr[];
    arr = new int[s+1];
    System.out.println("Уведіть розмірності матриць");
    for(int i = 0; i < s+1; i++) {
      int temp = scanner.nextInt();
      arr[i] = temp;
    }
    System.out.println("Елементи матриці: ");
    for(int i = 0; i < s; i++) {
      System.out.println(name2 + "("+ arr[i] + "; " + arr[i+1] + ")");
      name2++;
    }
    System.out.println();
    int n = arr.length;
    matrixChainOrder(arr, n);
```

```
    }
}
```

Результати роботи програми наведені на рис. 2.1:

```
Уведіть n - кількість матриць
5
Уведіть розмірності матриць
20 1 10 3 5 7
Елементи матриці:
A(20; 1)
B(1; 10)
C(10; 3)
D(3; 5)
E(5; 7)


Оптимальна таблиця:
0       1       2       3       4       5
1       0       200     90      145     220
2       0       0       30      45      80
3       0       0       0       150     315
4       0       0       0       0       105
5       0       0       0       0       0
Оптимальне розставлення дужок: (A(((BC)D)E))
```

Рисунок 2.1 – Результат

Крім того була оцінена складність даного алгоритму. Її зображено в таблиці 2.2.

Таблиця 2.2 – Складність алгоритму

№ рядка	Алгоритм	Кількість операцій	Скільки разів буде виконаний рядок
1	for (int i = 1; i < n; i++)	c_1	n+1
2	m[i][i] = 0;	c_2	n
3	for (int L = 2; L < n; L++)	c_3	n
4	for (int i = 1; i < n - L + 1; i++)	c_4	(n-L)(n-1)
5	for (int k = i; k <= j - 1; k++)	c_5	(n-L)(n-1)(j-1)
6	int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];	c_6	(n-L)(n-1)(j-1)
7	if (q < m[i][j])	c_7	(n-L)(n-1)(j-1)
8	m[i][j] = q;	c_8	(n-L)(n-1)(j-1) - 1
9	bracket[i][j] = k;	c_9	(n-L)(n-1)(j-1) – 1

В таблиці 2.1 описано основні кроки алгоритму, а всі інші місця виконуватимуться менше раз(1 або n). Порахуємо складність алгоритму:

$T(n) = c(n+1)_1+c_2n+c_3n+c_4(n-L)(n-1)+c_5(n-L)(n-1)(j-1)+c_6(n-L)(n-1)(j-1)+$
$+c_7(n-L)(n-1)(j-1)+c_8(n-L)(n-1)(j-1) – 1)+c_9((n-L)(n-1)(j-1) – 1) =$

Якщо ми відкинемо елементи нижчих порядків:

$T(n) = O(n^3)$.

ВИСНОВКИ

Під час цієї лабораторної роботи була розроблена програма, яка за допомогою алгоритму динамічного програмування виконує необхідну роботу, а саме записує оптимальний спосіб множення матриць. Цей алгоритм є корисним для використання в даному випадку, тому що динамічне програмування допомагає знайти найвигідніший спосіб множення матриць з найменшою кількістю виконаних дій. Крім того була оцінена складність даного алгоритму, яка характеризує ефективність даного алгоритму.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Методичні вказівки до виконання лабораторних робіт з курсу "Алгоритми і структури даних": для студентів, які навчаються за спец. 121 "Інженерія програмного забезпечення" [Електронний ресурс] / уклад. Н. К. Стратієнко, І. О. Бородіна ; Харківський політехнічний інститут, національний технічний університет університет – Електрон. текстові дані. – Харків, 2017. – 36 с. 05.05.2021

2 Алгоритми і структури даних: практикум: навч. посіб./ Н.К. Стратієнко, М.Д. Годлевський, І.О. Бородіна.- Харьков: НТУ"ХПИ", 2017. - 224 с. 05.05.2021