ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КУРСОВАЯ РАБОТА

по дисциплине “ Методы оптимизации ”

на тему:

«Задача о перемножении матриц»

Выполнил студент ____

Ф.И.О.

Группы ____

Работу принял ________________________ Доцент к. ф.м.н. А.А. Рубан

подпись

Защищена __________________________ Оценка ____________________

Новосибирск 2021

Оглавление

Задание

Расставить скобки при перемножении матриц оптимальным образом.

Алгорит работы программы

Как известно из высшей математики, умножение матриц ассоциативно, то есть результат перемножения зависит только от порядка матриц и не зависит от расстановки скобок:

(A*B)*C = A*(B*C).

Результат перемножения от расстановки скобок не зависит, зато трудоемкость этого перемножения при разных расстановках скобок может отличаться существенно.

Оценим трудоемкость умножения двух матриц A(p×q) и B(q×r):

C(p×r) = A(p×q)* B(q×r),

каждый элемент матрицу C(p×r) есть сумма q попарных произведений.

Трудоемкость перемножения двух матриц: T = p·q·r.

Здесь *f(k,p)* – минимальное количество действий, за которое можно вычислить произведение матриц с *k*-той по *p*-тую.
Выбираем минимум среди всех этих чисел и получаем общую формулу:

$$f(k,p) = \min_{k \le j \le p-1} \left(f(k,j) + f(j+1,p) + r_{k-1} \cdot r_j \cdot r_p \right).$$

Итак, в окончательном виде эта задача решается с помощью следующего алгоритма.

1) Заполняем трудоемкости матриц: Трудоемкости по главной диагонали равны 0:
   for i:=0 to n do f(i,i):=0;
2) Внешний цикл по t – длине перемножаемого блока;
   Средний цикл по k – местоположению блока;
   Внутренний – поиски минимума по j.
   for t:=1 to n-1 do
   for k:=1 to n-1 do
   $$f(k,k+t) = \min_{k \le j \le k+t-1} \left(f(k,j) + f(j+1,k+t) + r_{k-1} \cdot r_j \cdot r_{k+t} \right).$$

Для матриц M_1, M_2, M_3, M_4 из рассмотренного выше примера расставим скобки оптимальным образом.

f(1,1)	*f(1,2)*	*f(1,3)*	*f(1,4)*
	f(2,2)	*f(2,3)*	*f(2,4)*
		f(3,3)	*f(3,4)*
			f(4,4)

Итак, заполняем такую матрицу в следующем порядке:

f(1,1) = f(2,2) = f(3,3) = f(4,4) = 0

f(1,2) = min (f(1,1) + f(2,2) + 10·20·50) = 10 000

f(2,3) = min (f(2,2) + f(3,3,) + 20·50·1) = 1000
f(3,4) = min (f(3,3) + f(4,4) + 50·1·100) = 5000
f(1,3) = min (f(1,1) + f(2,3) + 10·20·1; f(1,2) + f(3,3) + 10·50·1) = 1200
f(2,4) = min (f(2,2) + f (3,4) +20·50·100; f(2,3) + f (4,4) + 20·1·100) = 3000
f(1,4) = min (f(1,1) + f(2,4) + 10·20·100; f(1,2) + f(3,4) + 10·50·100; f(1,3) + f(4,4) + 10·1·100) = 2200

После вычисления оптимальных трудоемкостей восстанавливаем оптимальную расстановку скобок.

Смотрим, где достигнут минимум в *f(1,4)*. Он достигнут в *f(1,3) + f(4,4) + 10·1·100.* Следовательно, последними скобками будут $(M_1\ M_2\ M_3)(M_4)$.

Далее смотрим, где достигнут минимум в *f(1,3)*. Он достигнут в *f(1,1) + f(2,3) + 10·20·1*. Т.е. следующими скобками будут $(M_1\ (M_2\ M_3)\ M_4$.

Т.к. у нас 3 вложенных цикла, длина каждого порядка *n* (*n* – количество перемножаемых матриц), то трудоемкость решаемой задачи методом ДП $T = C \cdot n^3$.

Пример:

Б) Расставить скобки при перемножении матриц оптимальным образом.

M_1=[10×20], M_2=[20×5], M_3=[5×4], M_4=[4×30], M_5=[30×6].

Решение.

r_0=10, r_1=20, r_2=5, r_3=4, r_4=30, r_5=6.

Вычислим оптимальные трудоемкости перемножения матриц:

f(1,1)= f(2,2)= f(3,3)= f(4,4)= f(5,5)=0.

f(1,2)= f(1,1)+ f(2,2)+ $r_0 \cdot r_1 \cdot r_2$=0+0+10·20·5=1000;

f(2,3)= f(2,2)+ f(3,3)+ $r_0 \cdot r_1 \cdot r_2$=0+0+20·4·5=400;

f(3,4)= f(3,3)+ f(4,4)+ $r_0 \cdot r_1 \cdot r_2$=0+0+5·4·30=600;

f(4,5)= f(4,4)+ f(5,5)+ $r_0 \cdot r_1 \cdot r_2$=0+0+4·30·6=720;

f(1,3)=min(f(1,1)+ f(2,3)+ $r_0 \cdot r_1 \cdot r_2$=0+400+10·20·4=1200;f(1,2)+ f(3,3)+ $r_0 \cdot r_1 \cdot r_2$=1000+0+10·4·5=1200)=1200

f(2,4)=min(f(2,2)+ f(3,4)+ $r_1 \cdot r_2 \cdot r_4$=0+600+20·5·30=3600;f(2,3)+ f(4,4)+ $r_1 \cdot r_3 \cdot r_4$=400+0+20·4·30=2800)=2800

f(3,5)=min(f(3,3)+ f(4,5)+ $r_2 \cdot r_3 \cdot r_5$=0+720+5·4·6=840;f(3,4)+ f(5,5)+ $r_2 \cdot r_4 \cdot r_5$=600+0+5·30·6=900)=840

$f(1,4)=\min(f(1,1)+ f(2,4)+ r_0 \cdot r_1 \cdot r_4=0+2800+10\cdot 20\cdot 30=8800; f(1,2)+ f(3,4)+ r_0 \cdot r_2 \cdot r_4=1000+600+10\cdot 5\cdot 30=3100; f(1,3)+ f(4,4)+ r_0 \cdot r_3 \cdot r_4=1200+0+10\cdot 4\cdot 30=2400)=2400;$

$f(2,5)=\min(f(2,2)+ f(3,5)+ r_1 \cdot r_2 \cdot r_5=0+840+20\cdot 5\cdot 6=1440;$

$f(2,3)+ f(4,5)+ r_1 \cdot r_3 \cdot r_5=400+720+20\cdot 4\cdot 6=1600; f(2,4)+ f(5,5)+ r_1 \cdot r_4 \cdot r_5=2800+0+20\cdot 30\cdot 6=6400)=1400;$

$f(1,5)=\min(f(1,1)+ f(2,5)+ r_0 \cdot r_1 \cdot r_5=0+1440+10\cdot 20\cdot 6=2640; f(1,2)+ f(3,5)+ r_0 \cdot r_2 \cdot r_5=1000+840+10\cdot 5\cdot 6=2140; f(1,3)+ f(4,5)+ r_0 \cdot r_3 \cdot r_5=1200+720+10\cdot 4\cdot 6=2160;$

$f(1,4)+ f(5,5)+ r_0 \cdot r_4 \cdot r_5=2400+0+10\cdot 30\cdot 6=4200)=2140;$

Заполним таблицу:

[10x20]	[20x5]	[5x4]	[4x30]	[30x6]
0	1000	1200	2400	**2140**
0	0	400	2800	1440
0	0	0	600	840
0	0	0	0	720

Минимальная трудоемкость перемножения матриц Min=2140, значение всегда будет находиться в `a[0][N - 1]`

Алгоритм расстановки скобок

Восстановим оптимальную расстановку скобок:

На каждом шаге будем запоминать, при каком соотношении было достигнуто минимальное число. Результат запомним в виде Хэш таблицы, где ключ: <строка, столбец>, а значение < ***k*** – итерация цикла, при которой получили минимум>

min f(1,5) достигнут на f(1,2)+ f(3,5): = f(1,2)+840 **(k=2)**

$(M_1 \cdot M_2) \cdot (M_3 \cdot M_4 \cdot M_5)$

min f(3,5) достигнут на f(3,3)+ f(4,5): **(k=3)**

$(M_1 \cdot M_2) \cdot (M_3 (M_4 \cdot M_5))$

Ответ: $(M_1 \cdot M_2) \cdot (M_3 (M_4 \cdot M_5))$

Выведем значения записанные в хэштаблицу:

f[1][2]: min=1000, l=1 = ~~f[1][1]~~ + = ~~f[2][2]~~

f[2][3]: min=400, l=2

f[3][4]: min=600, l=3

f[4][5]: min=720, l=4 = ~~f[4][4]~~ + = ~~f[5][5]~~

f[1][3]: min=1200, l=1

f[2][4]: min=2800, l=3

f[3][5]: min=840, l=3 = ~~f[3][3]~~ + f[4][5]:

f[1][4]: min=2400, l=3

f[2][5]: min=1440, l=2

f[1][5]: min=2140, l=2 = f[1][2] + f[3][5]

начинаем тут всегда l l+1

Рекурсивно восстановим последовательность:

```
f(row, col) = f(row, k) + f(k, col);
```

Получаем скобки для (1,2), (3,5), (4,5)

[10x20]	[20x5]	[5x4]	[4x30]	[30x6]
0	1000	1200	2400	2140
0	0	400	2800	1440
0	0	0	600	840
0	0	0	0	720
0	0	0	0	0

```
(M1*M2)*(M3*(M4*M5))
```

Приложение 1. Пример выполнения программы

Input (matrmulti.in)	Output Console
5 10 10 10 10 10 10	[10x10][10x10][10x10][10x10][10x10] 0 1000 2000 3000 4000 0 0 1000 2000 3000 0 0 0 1000 2000 0 0 0 0 1000 0 0 0 0 0 Min=4000 (((M1*M2)*M3)*M4)*M5
5 10 20 5 4 30 6	[10x20][20x5][5x4] [4x30] [30x6] 0 1000 1200 2400 2140 0 0 400 2800 1440 0 0 0 600 840 0 0 0 0 720 0 0 0 0 0 Min=2140 (M1*M2)*(M3*(M4*M5))
3 10 5 30 5	[10x5] [5x30] [30x5] 0 1500 1000 0 0 750 0 0 0 Min=1000 M1*(M2*M3)

Приложение 2. Листинг программы

```java
package labs;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Scanner;
/**
 * Расставить скобки при
 * перемножении матриц оптимальным образом.
 */
public class ParenthesesMatricesMultiplying {
	private HashMap<Key, Integer> hashMap = new HashMap<Key, Integer>();
	private int[] begin;
	private int[] end;

	public static void main(String[] args) throws IOException {
		new ParenthesesMatricesMultiplying();
	}

	public ParenthesesMatricesMultiplying() throws IOException {
		Scanner scanner = new Scanner(new File("matrmulti.in"));
		int n = scanner.nextInt(); // количество матриц
		int[] r = new int[n + 1];

		begin = new int[n];  //открывающие скобки
		end = new int[n]; // закрывающие скобки

		for (int i = 0; i <= n; i++) {
			r[i] = scanner.nextInt();
		}

		// Алгоритм ______________________________________________
		int[][] f = new int[n][n];
		for (int t = 1; t < n; ++t) {
			for (int row = 0; row < n - t; ++row) {
				int col = row + t; //
// ходим по диагонали слева направо сверха сниз, потом более правая
									// диагональ
				f[row][col] = Integer.MAX_VALUE;
				for (int l = row; l < col; ++l) {// цикл по расстановкам

					int tmp = f[row][l] + f[l + 1][col] + r[row] * r[l +
1] * r[col + 1];
					if (f[row][col] >= tmp) {
						hashMap.put(new Key(row, col), l);  // при
добавлении по повторному ключу значение перезаписывается, не надо выносить за цикл
						f[row][col] = tmp;
						}
				}

			}
		}

		// вывод
		mapRecursion(0, n - 1); // расстановка скобок
		outPrint(r, f); //таблица
		begin[0]--;
		end[n - 1]--; //убираем для красоты скобки от первого и до последнего
		outPrint(r);// скобки
	}
```

```java
	void mapRecursion(int row, int col) {
		if (row != col) {
			begin[row]++;
			end[col]++;
			if (row - col != 1) {
				int l = hashMap.get(new Key(row, col));
				mapRecursion(row, l);
				mapRecursion(l + 1, col);

			}
		}
	}

	void outPrint(int r[], int a[][]) {
		for (int i = 1; i < r.length; i++) {
			System.out.print("[" + r[i - 1] + "x" + r[i] + "]\t");
		}
		System.out.println();
		for (int i = 0; i < a.length; i++) {
			for (int j = 0; j < a.length; j++) {
				System.out.print(a[i][j] + "\t");

			}
			System.out.println();
		}

		System.out.println("\nMin=" + a[0][a.length - 1]);
// в правом верхнем углу матрицы, оптимальное перемножение

		// матриц
	}

	public void outPrint(int a[]) {
		for (int i = 0; i < a.length - 1; i++) {
			if (i != 0) {
				System.out.print("*");
			}
			for (int j = 0; j < begin[i]; j++) {
				System.out.print("(");
			}
//System.out.print("M" + (i + 1) + "[" + a[i] + "x" + a[i + 1] + "]");
			 System.out.print("M" + (i + 1) );  // краткая запись
			for (int j = 0; j < end[i]; j++) {
				System.out.print(")");
			}
		}
		System.out.println();
	}
}
```