Centro Universitário da FEI

Programa de Mestrado e Doutorado em Engenharia Elétrica

PEL 201 – Algoritmos Computacionais

**Relatório 4**

**Algoritmo de Multiplicação de Matrizes**

Claudio Aparecido Borges Junior

Matrícula: 120122-7

**MULTIPLICAÇÃO DE MATRIZES**

O produto de duas matrizes é definido somente quando o número de colunas da primeira matriz é igual ao número de linhas da segunda. Sendo A uma matriz , e B uma matriz , então seu produto é uma matriz C . Cada elemento da i-ésima linha da matriz A é multiplicado por um elemento da coluna j-ésima da matriz B, e a somatória das multiplicações escalar é o elemento da matriz C. Desse modo, o produto de duas matrizes e resulta em multiplicações escalares, onde .

A multiplicação de cadeia de matrizes é associativa, ou seja, dado três matrizes A, B e C, o resulta é o mesmo se for computada A e B ou B e C em primeiro lugar. Desse modo, há uma sequencia de multiplicação onde há uma redução do número de multiplicações escalares. Por exemplo, considerando as matrizes , e temos duas possibilidades: (1) onde havendo 1920 multiplicações escalares e (2) havendo 3840 multiplicações escalares.

**ALGORITMO**

O algoritmo de multiplicação de matrizes busca a melhor forma de multiplicar um conjunto de matrizes, que reduz a quantidade de multiplicações escalares. Como visto anteriormente, utilizando a propriedade associativa, podemos encontrar uma sequência de multiplicação onde o número de multiplicações escalares é o mínimo.

Utilizando programação dinâmica, em uma abordagem *bottom-*up, primeiramente computamos a quantidade de multiplicações escalares para pares de um conjunto de Matrizes. Depois, utilizando os valores já computados no passo anterior, computamos trios de de um conjunto de de Matrizes. Como não sabemos qual o melhor conjunto de Matrizes, haverá testes, nessa etapa. A próxima etapa busca encontrar o melhor quarteto, haverá testes, produzindo um conjunto de possíveis melhores casos. Esse processo ocorre até existir apenas 1 elemento restante que será o menor valor para a multiplicação de um conjunto de matrizes. Se guardarmos qual a sequencia de multiplicação para cada etapa do processo, podemos reconstruir toda a sequencia em uma abordagem *top-down*.

**IMPLEMENTAÇÃO**

O algoritmo foi implementado utilizando Python 3.7.7.

01 def matrix\_infix(s, i, j):

02 if i == j:

03 return 'A' + str(i)

04 mid = s[i][j]

05 left = matrix\_infix(s, i, mid)

06 right = matrix\_infix(s, mid + 1, j)

07 if i != mid:

08 left = '(' + left + ')'

09 if mid + 1 != j:

10 right = '(' + right + ')'

11 return left + '.' + right

12

13

14 def best\_matrix\_mult(p):

15 # p is a list of matrices dimensions without repetition with n + 1 elements

16 # where n is the number of matrices, e.g. [2, 3, 4] represent 2 matrices

17 # of 2x3 and 3x4

18 n = len(p) - 1

19 m = [[0 for i in range(n)] for j in range(n)]

20 s = [[0 for i in range(n)] for j in range(n)]

21

22 for d in range(n):

23 for j in range(d, n):

24 i = j - d

25 if i == j:

26 m[i][j] = 0

27 s[i][j] = i

28 continue

29

30 v\_min = None

31 for k in range(i, j):

32 v = m[i][k] + m[k + 1][j] + p[i] \* p[k + 1] \* p[j + 1]

33 if v\_min is None or v < v\_min:

34 v\_min = v

35 m[i][j] = v\_min

36 s[i][j] = k

37

38 print('Number of matrices: ' + str(n))

39 print('Matrices: ' + str(p))

40 print('Scalar multiplication: ' + str(m[0][n - 1]))

41 print('Result: ' + matrix\_infix(s, 0, n - 1))

A função best\_matrix\_mult recebe uma lista de dimensões de matrizes sem repetição com n + 1 elementos, onde n é a quantidade de matrizes. Por exemplo, o vetor [2, 3, 4] representa 2 matrizes sendo elas de dimensões 2x3 e 3x4. Uma vez que foi identificado qual melhor sequência de multiplicação para cada etapa, a função matriz\_infix percorre recursivamente o melhor caminho.

Por exemplo, dada a operação , representada por [10, 2, 8, 9, 14], o algoritmo produz a saída:

01 Number of matrices: 4

02 Matrices: [10, 2, 8, 9, 14]

03 Scalar multiplication: 676

04 Result: A0.((A1.A2).A3)

Em que a linha 01 representa a quantidade de matrizes, a linha 02 as dimensões das matrizes como explicado anteriormente, a linha 03 a quantidade de multiplicações escalares, e a linha 04 a posições dos parêntesis para obter a menor quantidade de multiplicações escalares em notação infixa.

**RESULTADOS DO ALGORÍTMO**

Abaixo estão 3 exemplos de multiplicação de matrizes para entradas de 10, 12, e 15, 20 matrizes respectivamente:

Number of matrices: 10

Matrices: [90, 87, 18, 24, 24, 53, 71, 42, 71, 57, 71]

Scalar multiplication: 610002

Result: (A0.A1).(((((((A2.A3).A4).A5).A6).A7).A8).A9)

Number of matrices: 12

Matrices: [47, 18, 78, 66, 28, 42, 51, 47, 48, 18, 27, 35, 34]

Scalar multiplication: 354420

Result: A0.((((((A1.A2).A3).(A4.(A5.(A6.(A7.A8))))).A9).A10).A11)

Number of matrices: 15

Matrices: [69, 25, 65, 68, 89, 94, 52, 77, 58, 35, 16, 80, 79, 87, 98, 29]

Scalar multiplication: 1026208

Result: (A0.(A1.(A2.(A3.(A4.(A5.(A6.(A7.(A8.A9))))))))).((((A10.A11).A12).A13).A14)

Number of matrices: 17

Matrices: [68, 56, 40, 75, 26, 21, 71, 20, 29, 40, 90, 90, 43, 51, 86, 28, 51, 97]

Scalar multiplication: 1034260

Result: (A0.(A1.((A2.A3).(A4.(A5.A6))))).(((((((((A7.A8).A9).A10).A11).A12).A13).A14).A15).A16)

**DESEMPENHO DO ALGORITMO**

O algoritmo possui desempenho assintótico de Θ (n3). O algoritmo foi testado utilizando 25 pontos, iniciando em 10 e terminando em 490 matrizes. O gráfico abaixo mostra o relacionamento entre a quantidade de matrizes e o tempo de execução.

Dada a curva do gráfico acima , é possível encontrar , onde , a partir de um número de matrizes, onde i é o elemento da i-ésima posição. O gráfico abaixo mostra esse relacionamento onde foram utilizados , representado pela curva laranja e representado pela curva cinza.