ICMC - Instituto de Ciências Matemáticas e da Computação
BCC - Bacharelado em Ciências de Computação
Aluno: Bernardo Marques Costa
Número USP: 11795551
Docente: Moacir Ponti
Disciplina: Introdução a Ciência da Computação II

PROVA DE INTRODUÇÃO A CIÊNCIA DA COMPUTAÇÃO II - ANÁLISE E OTIMIZAÇÃO DE ALGORITMOS

Especificação:

Considerando uma matriz quadrada A, de tamanho n = m x m, e considerando os índices i = 0, …, m - 1 para as linhas e j = 0, …, m - 1 para as colunas. Computamos uma matriz da seguinte maneira

- se $i > j$, valor $A(i,j) = (i + j)/2$
- se $i < j$, valor $A(i, j) = (i + j)/4$
- se $i = j$, $i = 0$, valor A(i,j) = 0
- se $i = j$, $i > 0$, valor $A(i,j) = \sum_{j=0}^{m-1} A(i-1, j)$ para i > j

Análise do algoritmo iterativo

```
for (int i = 0; i < m; i++) {       // ocorre m vezes

    for (int j = 0; j < m; j++) {  // ocorre m vezes
        if (i > j) {               // comparação - c
            mat[i][j]= (i+j)/2.0;  // aritmética - 2a
        } else if (i < j) {            // comparação - c
            mat[i][j]= (i+j)/4.0;  // aritmética - 2a
        } else {
            double soma = 0.0;

            for (int a = 0; a < m; a++) {          // ocorre m vezes
                if (i-1 >= 0)
                    soma += mat[i-1][a]; // comparação e aritmética
```

```
                                        // (a + c)m
                                        // interno: 2a(m - 1)
            mat[i][j]= soma;
        }
    }
}
```

Devemos realizar a análise para o pior caso, mas sabemos que algumas linhas do algoritmo não são executadas durante todo o processo, como por exemplo, as linhas do else:

```
else{ // Entra neste else 1 vez para cada linha da matriz
    double soma;
    for(int a = 0; a < m; a++) // Roda m vezes
        if (i-1 >= 0) soma += mat[i-1][a]; // condição - a + c
    mat[i][j]= soma;                       // interno  - 2a
}
// A condição ocorre m vezes, entrando somente nos casos em que i > 0
```

Imaginando uma matriz m x m com o seguinte formato:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Situação (1) $i > j$:

A matriz possui um total de $m(m-1)/2$ índices tais que $i > j$.

Equação: $(m(m-1)/2) * (2a + c)$

Situação (2) $i < j$:

A matriz possui um total de $m(m-1)/2$ índices tais que $i < j$.

Equação: $(m(m-1)/2) * (2a + 2c)$

Situação (3) $i = j, i = 0$:

A matriz possui um total de 1 vez em que $i = j, i = 0$.

Temos essa situação para apenas a primeira linha da matriz.

Equação: $2c + 1 * (m(a + c))$

Situação (4) $i = j, i > 0$:

A matriz possui um total de $m - 1$ vezes em que $i = j, i > 0$.

Equação: $(m - 1) * (2c + m * (3a + c)) = 3am^2 - 3am + cm^2 + cm - 2c$

Obtendo agora a equação final do número de operações a partir da soma de todas situação encontrada acima, obtemos:

$$f(m) = 5am^2 + 2,5cm^2 - 4am + 0,5cm$$

Análise do algoritmo recursivo

Analisando agora o algoritmo com implementação recursiva, com a primeira chamada sendo feita por:

```
recursive(mat, m, 0)
```

```
    if(i >= m) return;                 // comparação
    for (int j = 0; j < m; j++) {  // roda m vezes
        if (i > j) mat[i][j]= (i+j)/2.0;        // c -> 2a
        else if (i < j) mat[i][j]= (i+j)/4.0;   // c -> 2a
        else {
            double soma = 0.0;
            for (int a = 0; a < m; a++)       // roda m vezes
                if (i-1 >= 0) soma += mat[i-1][a];  // c + a -> 2a
            mat[i][j]= soma;
        }
    }
    recursive(mat, m, i + 1);
```

Mais uma vez, temos a mesma análise de quando entra no primeiro if, quando entra no else if e quando entra de fato no else (em que a quantidade de operações é maximizada).
Devemos partir de i = 0 até i = m, sendo o momento que atinge o caso base, retornando a estrutura recursiva.

Para i = 0: $c + (m - 1)(2a + 2c) + 0(2a + c) + 2c + m(a + c)$
Para i = 1: $c + (m - 2)(2a + 2c) + 1(2a + c) + 2c + m(a + c) + 2am$
Para i = 2: $c + (m - 3)(2a + 2c) + 2(2a + c) + 2c + m(a + c) + 2am$
Para i = 3: $c + (m - 4)(2a + 2c) + 3(2a + c) + 2c + m(a + c) + 2am$
…
Para i = k: $c + (m - k - 1)(2a + 2c) + k(2a + c) + 2c + m(a + c) + 2am$
...

Para i = m: c

Podemos observar um padrão de crescimento, sendo que os passos anterior correspondem a cada chamada independente, bastando realizar a somatória de todas estas chamadas para encontrar o número de operações do algoritmo recursivo.

Como devemos encontrar o caso base, em que i = m, devemos ter m iterações.

$$f(m) = (m+1)c + \sum_{i=0}^{m-1} (m-i-1)(2a+2c) + \sum_{i=0}^{m-1} i(2a+c) + 2cm + m^2(a+c) + (m-1)2am$$

$$f(m) = cm + (m^2 - m)(a+c) + (m^2 - m)(2a+c)/2 + 2cm + am^2 + cm^2 + 2am^2 - 2am + c$$

Finalmente:

$$f(m) = 5am^2 + 2,5cm^2 - 4am + 1,5cm + c$$

Como é possível observar, o número de operações cresce em (m + 1) operações de comparações, já que confere m + 1 vezes se o caso base foi encontrado.

Otimização de algoritmo

```
void optimus_prime_algoritm(double **mat, int m){
    double *sum_vec = calloc(m, sizeof(double));
    double sum = 0;
    for (int i = 0; i < m; ++i) { // roda m vezes
        double sum = 0.0;
        for (int j = 0; j < m; ++j) {  // roda m*m vezes

            if(i == j) { // m*m*(c) -> Entra uma vez apenas
                // c - caso em que i = 0
                // (m-1)*(c + a) -  [se i > 0 : + a]
                if(i > 0) mat[i][j] = sum_vec[i-1];
                else mat[i][j] = 0.0;
            }
            else // entra (m - 1) vezes
                 // m*(m - 1) * (c + 2a)
                mat[i][j] = (i+j) / ((i > j)? 2.0 : 4.0);
            sum += mat[i][j]; // m*m*a
        }
        sum_vec[i] = sum;
    }
    free(sum_vec);
```

```
}
```

Podemos verificar que tanto o for externo quanto interno realizam m iterações.

A linha `if (j == i)` é executada m vezes, mas só entra uma vez a cada linha da matriz. Aninhado a este if, temos mais uma condição e a operação de aritmética.

O código entra na linha do else m - 1 vezes, executando as operações de comparação e aritmética. Finalizando o laço for mais interno com uma aritmética na atribuição `sum += mat[i][j]`

Contabilizando agora a equação do número de operações do algoritmo otimizado, teremos:

$$f(n) = cm^2 + (m^2 - m)(c + 2a) + am^2 + c + (m-1)(c + a)$$

$$f(n) = cm^2 + cm^2 + 2am^2 - cm - 2am + am^2 + c + cm + am - c - a$$

Assim, obtemos:

$$f(n) = 3am^2 + 2cm^2 - am - a$$

como a equação final do algoritmo otimizado implementado,