N584 – Projeto e Análise de Algoritmos

Prof. Napoleão Nepomuceno

AV2 - Lab04

Data do Laboratório: 23/10/2019

Márcio Heleno Matrícula: 1814038

> Entrega do trabalho: Data da Entrega: 27/10/2019 (enviar arquivo .odt)

Exercício 4 - Av2

- **Passo 1:** Implementar o seguinte código em Java ou equivalente em outra linguagem de programação.

```java
public class Exercicio1 {

    static int[][] qtde;
    static int[][] queb;

    public static void main(String[] args) {
        int [] p = {3, 1, 2, 3, 4, 2, 1, 5, 9, 8, 2, 4, 3, 6, 3, 8};
        //int [] p = {4, 5, 3, 2, 6, 1, 5, 6, 2, 7, 2, 1, 4, 3, 2, 8, 3,
        //      6, 5, 2, 6, 2, 5, 2, 8, 3, 6, 4, 5, 3, 7, 4, 6, 3};
        int n = p.length-1;
        double inicio, fim, tempo;
        int m;

        qtde = new int[n+1][n+1];
        queb = new int[n+1][n+1];
        for (int i = 1; i <= n; i++) {
            for (int j = i; j <= n; j++) {
                qtde[i][j] = Integer.MAX_VALUE;
            }
        }

        inicio = System.currentTimeMillis();
        m = recursivo(p, 1, n);
        fim = System.currentTimeMillis();
        tempo = fim - inicio;
        imprime(queb, 1, n);
        System.out.println();
        System.out.printf("%-15s%10s%10s\n", "Método", "Qtde", "Tempo");
```

```java
        System.out.printf("%-15s%10d%10.2f\n\n", "Recursivo", m, tempo);

        inicio = System.currentTimeMillis();
        m = memoization(p, 1, n);
        fim = System.currentTimeMillis();
        tempo = fim - inicio;
        imprime(queb, 1, n);
        System.out.println();
        System.out.printf("%-15s%10s%10s\n", "Método", "Qtde", "Tempo");
        System.out.printf("%-15s%10d%10.2f\n\n", "Memoization", m, tempo);

        inicio = System.currentTimeMillis();
        m = dinamico(p);
        fim = System.currentTimeMillis();
        tempo = fim - inicio;
        System.out.println();
        System.out.printf("%-15s%10s%10s\n", "Método", "Qtde", "Tempo");
        System.out.printf("%-15s%10d%10.2f\n", "Dinamico", m, tempo);
    }

    static int recursivo(int[] p, int i, int j) {
        if (i == j) {
            queb[i][j] = i;
            return 0;
        } else {
            int s = 0;
            int qMin = Integer.MAX_VALUE;
            for (int k = i; k < j; k++) {
                int q = recursivo(p, i, k) + recursivo(p, k+1, j) + p[i-1] * p[k] * p[j];
                if (q < qMin) {
                    qMin = q;
                    s = k;
                }
            }
            queb[i][j] = s;
            return qMin;
        }
    }

    static int memoization(int[] p, int i, int j) {
        if (i == j) {
            queb[i][j] = i;
            return 0;
        } else {
            int s = 0;
            int qMin = Integer.MAX_VALUE;
```

```java
            for (int k = i; k < j; k++) {
                int q = memoization(p, i, k) + memoization(p, k+1, j) + p[i-1] * p[k] * p[j
                if (q < qMin) {
                    qMin = q;
                    s = k;
                }
            }
            queb[i][j] = s;
            return qMin;
        }
    }

    static int dinamico(int[] p) {
        int n = p.length-1;
        int m[][] = new int[n+1][n+1];
        int s[][] = new int[n+1][n+1];
        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; k++) {
                    //to do
                }
            }
        }
        imprime(s, 1, n);
        return m[1][n];
    }

    static void imprime(int[][]s, int i, int j) {
        if (i == j) {
            System.out.print("A" + i);
        } else {
            System.out.print("(");
            imprime(s, i, s[i][j]);
            imprime(s, s[i][j]+1, j);
            System.out.print(")");
        }
    }

}
```

Passo 2: Considere o problema da parentização de uma cadeia de multiplicação de matrizes, onde o vetor p expressa as dimensões das matrizes. Seja m[i,j] a quantidade de multiplicações escalares da parentização ótima da cadeia de matrizes Ai..j. Sua definição recursiva é dada abaixo:

$$m[i,j] = \begin{cases} 0 & \text{se i=j} \\ min = \{m[i,j] + m[k+1,j] + p_{i-1}\ P_k\ P_j\} & sei \text{ != j} \\ 0 < i < m \end{cases}$$

a) Por que m[i, j] = 0 se i = j? (5%) Pois o problema e trivial, a cadeia de consiste em apenas uma matriz, de modo que nenhuma multiplicação e necessária.

b) O que representa cada valor de k ? (5%) O valor da solução otima para a parentização, ou ponto de quebra.

c) O que representa cada um dos 3 termos da expressão $m[i,k] + m[k+1,j] + p_{i-1}\ p_k\ p_j$? (10%) É o cálculo da divisão ótima. A matrix P_1 representada por $m[i,k]$, de i ate o ponto de quebra k, P_2 do proximo valor de dimesão k+1 ate j, e a parte de calculo que é p, onde p representa o valor de indice da matriz P_1 multiplicado por P_k o indice onde localiza o ponto de quebra, e p_j onde a matriz no caso P_2 e o indice de colunas dessa matriz.

d) Qual o cálculo para o valor de m[i, i+1] ? (10%)

e) Por que a abordagem de divisão e conquista não se mostra adequada para este problema? (5%) Repetição de Subproblemas, ou seja ela calcula as matrizes mais de uma vez.

f) Para o vetor p = [3, 1, 2, 3, 4], quantas são as matrizes e quais são suas dimensões? (5%) $P^1_{3x1}, P^2_{1x2}, P^3_{2x3}, P^4_{3x4}$

g) Quais são todas as possíveis parentizações para este exemplo? Dica: são cinco. (5%) $(P_1(P_2(P_3P_4)))$ $(P_1((P_2P_3)P_4))$ $((P_1P_2)(P_3P_4))$ $((P_1(P_2P_3))P_4)$ $(((P_1P_2)P_3)P_4)$

Passo 3: Considerando a implementação recursiva dada na função memoization, adicione as instruções para efetivamente realizar memoization. (15%)

```java
static int memoization(int[] p, int i, int j) {
  if (qtde[i][j] != Integer.MAX_VALUE) {
      return qtde[i][j];
  }
  if (i == j) {
      qtde[i][j] = 0;
      queb[i][j] = i;
      return 0;
  } else {
      int s = 0;
      int qMin = Integer.MAX_VALUE;
      for (int k = i; k < j; k++) {
          int q = memoization(p, i, k) + memoization(p, k+1, j) + p[i-1] * p[k] * p[j];
          if (q < qMin) {
              qMin = q;
              s = k;
```

```
            }
        }
        queb[i][j] = s;
        qtde[i][j] = qMin;
        return qMin;
    }
}
```

Passo 4: Considerando a implementação da função dinamico, adicione as instruções para o preenchimento das matrizes m e s. (15%)

```
static int dinamico(int[] p) {
  int n = p.length-1;
  int m[][] = new int[n+1][n+1];
  int s[][] = new int[n+1][n+1];
  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; k++) {
              //to do
              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;
                  s[i][j] = k;
              }
          }
      }
  }
  imprime(s, 1, n);
  return m[1][n];
}
```

Passo 5: Qual a equação de recorrência para o melhor caso do método imprime? Qual a sua complexidade? (10%)

Passo 6: Qual a equação de recorrência para o pior caso do método imprime? Qual a sua complexidade? (10%)

$$T(n) = \begin{cases} \Theta(1) & \text{se i=j} \\ 2T(\frac{n}{2}) + \Theta(n) & \text{se != j} \end{cases}$$

ou seja a equação: $T(n) = 2T(n/2) + \Theta(n)$

Complexidade: $T(n) = 2t(\frac{n}{2}) + \Theta(n)$

\$ {a = 2, b = 3, f(n) = n }\$

$n \log_b a = n \log_2 2 \Rightarrow n^1 = n$

Solução caso 2:

$f(n) = \Theta(n \lg n)$

Passo 7: O que acontece com a execução do programa caso o vetor p seja mudado conforme comentário no código? Explique porque o programa se comporta desta maneira. (5%)