Trabalho Final de Análise e Projeto de Algoritmos

Ícaro Machado Crespo[1]

Universidade Federal do Pampa (UNIPAMPA)
96.546.550 – Alegrete – RS – Brasil

{icarocrespo.aluno@unipampa.edu.br}

Resumo. *Este relatório-artigo tem como objetivo dissertar sobre o desenvolvimento de um dos problemas elencados como tema ao Trabalho Final da disciplina de Análise e Projeto de Algoritmos, ministrada pelo Prof. Dr. Arthur Francisco Lorenzon. Além da visão geral do software, é também visto explicitado os cálculos referentes ao custo dos algoritmos. Este trabalho é parte à aquisição da aprovação do componente curricular citado.*

1. Introdução

O presente artigo é fruto da disciplina de Análise e Projeto de Algoritmos da Universidade Federal do Pampa – *Campus* Alegrete, ministrada pelo Prof. Dr. Arthur Lorenzon, a qual visa a capacitação do acadêmico às práticas de análise e desenvolvimento de algoritmos, observando pontos como valoração do que é programado. O trabalho com entrega ao dia 28 de novembro de 2019 e sua apresentação entre os dias 02 e 05 de dezembro de 2019, é parte fundamental ao Engenheiro de Software que queira desempenhar seu papel no mercado, visto que o custo à máquina a ser executada dever ser observado e melhor aplicado.

Tendo em vista o disposto no parágrafo anterior, fora escolhido um dos temas elencados pelo professor, sendo este: “Maximizando a Soma”. A solução deve ser em cima de dois *arrays* de tamanho N, onde aos mesmos devem ser empregadas quatro operações. A estes, soluções utilizando os métodos vistos em sala de aula e análise de suas respectivas complexidades devem ser entregues, juntamente com o presente artigo.

2. Desenvolvimento

O software começou a ser desenvolvido a partir da escolha de um dos enunciados disponibilizado pelo professor (Maximizando a Soma). Após, fora feita a escolha da linguagem de programação, Java Desktop. O software teve seu versionamento controlado pela plataforma Git

O desenvolvimento consistiu na programação de uma classe (ForcaBruta), a qual é solução do problema utilizando o método aprendido em sala de aula, Força Bruta, e uma classe (ProgramacaoDinamica) aplicando um dos conceitos possíveis à escolha ao desenvolvimento, Programação Dinâmica. Além disso, uma classe centralizando a chamada (Main) fora desenvolvida, a fim de ser a interface com o usuário.

2.1 Força Bruta

A implementação do código utilizando o método força bruta totalizou em 122 linhas de código, contendo métodos assistivos para a população e exibição dos *arrays*. A Figura 1

mostra um trecho do código desta classe para conhecimento das nomenclaturas utilizadas e sua inicialização.

```
6   public class ForcaBruta {
7
8       Integer[] arrayA;
9       Integer[] arrayP;
10
11      public ForcaBruta(Integer tamanho) {
12          this.arrayA = new Integer[tamanho];
13          this.arrayP = new Integer[tamanho];
14      }
15
16      public void populaArray() {
17          Scanner x = new Scanner(System.in);
18          System.out.println("Digite os " + arrayA.length + " valores");
19          for (int i = 0; i < arrayA.length; i++) {
20              arrayA[i] = x.nextInt();
21          }
22      }
23
24      public void operacao1() {
25          if (arrayA.length > 0) {
26              for (int i = 0; i < arrayP.length; i++) {
```

Figura 1. Trecho de código, classe ForcaBruta

2.2 Programação Dinâmica

O paradigma de Programação Dinâmica teve seu emprego na classe Programação Dinâmica e não fora implementado em sua totalidade, apenas métodos de exibição e inserção.

3. Resultados Obtidos – Análise de Complexidade

3.1 Força Bruta

À Força Bruta, cada operação é mostrada, seguida pelo seu cálculo relacionando com a linha do código.

```
24      public void operacao1() {
25          if (arrayA.length > 0) {
26              for (int i = 0; i < arrayP.length; i++) {
27                  if (arrayP[i] == null) {
28                      for (int j = 0; j < arrayA.length; j++) {
29                          if (arrayA[j] == null) {
30                              arrayP[i] = arrayA[j - 1];
31                              arrayA[j - 1] = null;
32                          }
33                      }
34                  }
35              }
36          }
37      }
```

Figura 2. Operação 1 de Força Bruta

Sendo N o tamanho do vetor, segue a disposição do cálculo: Linha 25 = 1; Linha 26 = N + 1; Linha 27 = 1; Linha 28 = N + 1; Linha 29 = 1; Linha 30 = 1; Linha 31 = 1.

```
39      public void operacao2() {
40          if (arrayA.length > 1) {
41              for (int i = 0; i < arrayA.length; i++) {
42                  if (i - arrayA.length == 0 && (i + 1) - arrayA.length == -1) {
43                      for (int j = 0; j < arrayP.length; j++) {
44                          if (arrayP[j] == null) {
45                              arrayP[j] = arrayA[i] * arrayA[i + 1];
46                              arrayA[i] = null;
47                              arrayA[i + 1] = null;
48                          }
49                      }
50                  }
51              }
52          } else {
53              System.out.println("Tamanho do Array A menor que 1.");
54          }
55      }
```

Figura 3. Operação 2 de Força Bruta

Sendo N o tamanho do vetor, segue a disposição do cálculo: Linha 40 = 1; Linha 41 = N + 1; Linha 42 = 1; Linha 43 = N + 1; Linha 44 = 1; Linhas 45 a 47 e 53 = 1.

```
57      public void operacao3() {
58          Integer new_arrayA[] = new Integer[arrayA.length];
59
60          int i = arrayA.length - 1;
61          int j = 0;
62
63          while (i >= 0) {
64              new_arrayA[j] = arrayA[i];
65              arrayA[i] = null;
66              i--;
67              j++;
68          }
69
70          for (int k = 0; k < arrayA.length; k++) {
71              arrayA[k] = new_arrayA[k];
72              if (k + 1 == arrayA.length) {
73                  for (int l = 0; l < arrayP.length; l++) {
74                      if (arrayP[l] == null) {
75                          arrayP[l] = arrayA[k];
76                      }
77                  }
78                  arrayA[k] = null;
79              }
80          }
81
82      }
```

Figura 4. Operação 3 de Força Bruta

Sendo N o tamanho do vetor, segue a disposição do cálculo: Linha 58 = 1; Linhas 60 e 61= 1; Linha 63 = N.logN; Linhas 64 a 67 = 1; Linha 70 = N + 1; Linha 71 e 72 = 1; Linha 73 = N + 1; Linhas 74 a 78 = 1.

```
84      public void operacao4() {
85          Integer new_arrayA[] = new Integer[arrayA.length];
86
87          int i = arrayA.length - 1;
88          int j = 0;
89
90          while (i >= 0) {
91              new_arrayA[j] = arrayA[i];
92              arrayA[i] = null;
93              i--;
94              j++;
95          }
96          if (new_arrayA.length > 1) {
97              for (int k = 0; k < new_arrayA.length; k++) {
98                  if (k + 2 == new_arrayA.length) {
99                      for (int l = 0; l < arrayP.length; l++) {
100                         if (arrayP[l] == null) {
101                             arrayP[l] = new_arrayA[k] * new_arrayA[k + 1];
102                         }
103                     }
```

Figura 5. Operação 4 de Força Bruta

3.2 Programação Dinâmica

Não houve análise da Programação Dinâmica deste trabalho, o qual não fora implementado em sua completude.

5. Considerações Finais

O emprego do paradigma de Programação Dinâmica (PD) teve um desempenho melhor que o ForcaBruta (FB), como esperado. Com o PD, o não é necessário explorar todas as situações possíveis no algoritmo, fazendo com que ele seja mais “inteligente” e tenha um custo inferior ao FB.

Em relação aos valores, a complexidade é possível afirmar o custo maior da implementação a partir do paradigma Força Bruta, visto que ele testa todas as possibilidades. Como não houve implementação da Programação Dinâmica, podemos afirma apenas os estudos prévios feitos neste paradigma

Referências

CORMEN, T. H.; LEISERSON, C.; RIVEST, R.; STEIN, C. Algoritmos: teoria e prática. Rio de Janeiro: Elsevier, 2002.

DASGUPTA, S.; PAPADIMITRIOU, C.; VAZIRANI, U. Algoritmos. São Paulo: McGrawHill, 2009. NETTO, P. O. B. Grafos: teoria, modelos, algoritmos. 4.ed. São Paulo: Edgard Blucher, 2006