FCT/Unesp – Presidente Prudente

Projeto e Análise de Algoritmos

Prof. Danilo Medeiros Eler

Exercícios Aula 03

1) Verifique se cada questão abaixo é verdadeira ou falsa.

(a) $10^{56} \cdot n^2 \in O(n^2)$?

(b) $10^{56} \cdot n^2 \in O(n^3)$?

(c) $10^{56} \cdot n^2 \in O(n)$?

(d) $2^{n+1} \in O(2^n)$?

(e) $2^{2n} \in O(2^n)$?

(f) $n \in O(n^3)$?

2) Coloque em ordem crescente de complexidade as principais classes de problemas listadas a seguir.

$O(n!),\ O(n\ log\ n),\ O(n),\ O(log\ n),\ O(1),\ O(2^n),\ O(n^3), O(n^2)$

3) Expresse a função abaixo em termos da notação assintótica "O".

$$n^3/1000 - 100n^2 - 100n + 3$$

4) Analise o algoritmo abaixo e identifique o seu pior caso usando a notação assintótica "O".

```
exibe_matriz_3D(M)
    for i ← 1 to comprimento_x[M]
        for j ← 1 to comprimento_y[M]
            for k ← 1 to comprimento_z[M]
                do escreva(M[i][j][k]))
```

5) Apresente a análise detalhada de complexidade do método **main1**. Neste caso, é necessário analisar a complexidade dos métodos **subAlgoritmo01** e **subAlgoritmo02**. Utilize a notação assintótica para indicar a complexidade do algoritmo. Lembre que **size()** é um método que retorna o número de elementos de uma lista.

```
public void main1(ArrayList args){
        subAlgoritmo01();
        double x, y, z;
        for(int i = (1000+args.size()); i >= 0; i--){
                x=(double)i/2; y=(double)x/2; z=(double)x+y;
                System.out.print(z);
        }
        subAlgoritmo02(args);
}
public void subAlgoritmo01(){
        double x, y, z;
        for(int i = 1000; i >= 0; i--){
                x=i/2; y=x/2; z=x+y;
                System.out.print(z);
        }
}
public void subAlgoritmo02(ArrayList args){
        for(int i = 0; i < args.size(); i++){
                System.out.print(args.get(i));
        }
        for(int i = 0; i < Math.pow(args.size(),10); i++){
                System.out.println("aloAlo");
        }
        double x, y, z;
        for(int i = 1000+Math.pow(args.size(),5); i >= 0; i--){
                x=(double)i/2; y=(double)x/2; z=(double)x+y;
                System.out.print(z);
        }
}
```

6) Conforme o exercício anterior, apresente a análise detalhada de complexidade do algoritmo **main2**. Utilize a notação assintótica para indicar a complexidade. Lembre que **size()** é um método que retorna o número de elementos de uma lista.

```
public void main2(ArrayList args){
        subAlgoritmo01();
        subAlgoritmo02(args);
        subAlgoritmo03(args);
        subAlgoritmo04(args);
}
```

```
public void subAlgoritmo01(){
        double x, y, z;
        for(int i = 1000; i >= 0; i--){
                x=i/2; y=x/2; z=x+y;
                System.out.print(z);
        }
}
public void subAlgoritmo02(ArrayList args){
        for(int i = 0; i < args.size(); i++){
                System.out.print(args.get(i));
        }
        for(int i = 0; i < args.size(); i++){
                System.out.print(args.get(i));
        }
        double x, y, z;
        for(int i = 1000; i >= 0; i--){
                x=(double)i/2; y=(double)x/2; z=(double)x+y;
                System.out.print(z);
        }
}
public void subAlgoritmo03(ArrayList args){
        for(int i = 0; i < args.size(); i++){
                for(int j = 0; j < args.size(); j++){
                        for(int ki = 0; ki < args.size(); k++){
                                System.out.print("Alo mundo " + i*j*k);
                        }
                }
        }
}

public void subAlgoritmo04(ArrayList args){
        for(int i = 0; i < Math.pow(args.size(),2); i++){
                System.out.print("Alo mundo " + i);
        }
}
```

7) Apresente a análise detalhada de complexidade dos subprogramas abaixo. Lembre que **size()** é um método que retorna o número de elementos de uma lista. Utilize a notação assintótica.

```
Pessoa busca(String nome){
  for (int i = 0; i< pessoas.size(); i++){
       if (pessoas.get(i).getNome().equals(nome))
          return pessoas.get(i);
  }
  return null;
}
```

a)
```
void exibir(String nome){
   Pessoa p = busca(nome);
   if (p != null){
      p.exibirDados();
   }
  else{  System.out.println(“Pessoa não encontrada”);
  }
}
```

b)
```
void exibir(String nome){
   if (busca(nome) != null){
      busca(nome).exibirDados();
   }
  else{
       System.out.println(“Pessoa não encontrada”);
  }
}
```
c)
```
void atualizar(String nome, int idade, float salario){
    Pessoa p = busca(nome);
   if (p != null){
      p.setIdade(idade);
      p.setSalario(salario);
   }
  else{
       System.out.println(“Pessoa não encontrada”);
  }
}
```

d)
```
void atualizar(String nome, int idade, float salario){
   if (busca(nome) != null){
      busca(nome).setIdade(idade);
      busca(nome).setSalario(salario);
   }
  else{
       System.out.println(“Pessoa não encontrada”);
  }
}
```

8) Apresente a análise detalhada de complexidade dos subprogramas abaixo. Utilize a notação assintótica para apresenta a análise.

Algoritmo UnicoElemento (A, n)
//Determina se todos os elementos de um dado vetor são distintos
//**Entrada:** um vetor **A** e o número de elementos do vetor **n**
//Saída: retorna "verdadeiro" se todos os elementos em **A** são distintos e "falso" caso contrário

```
Para i = 1 até n – 1 faça
  Para j = i+1 até n faça
    Se A[i] = A[j] retorne falso
retorne verdadeiro
```