FCT/Unesp – Presidente Prudente

Projeto e Análise de Algoritmos

Prof. Danilo Medeiros Eler

Exercícios Aula 02

https://daniloeler.github.io/teaching/PAA2020/index.html

1) Para cada um dos trechos de código abaixo, analise o tempo estimado de execução no **melhor** e no **pior** caso, considerando o modelo RAM. Considere que as variáveis **n**, **m** e **vetor** sejam dados de entrada.

a)

```
int soma = 0;------------------------------------ 1
for (int i=0; i<n; i++)
  soma = soma + i; ------------------------------ 1 * n
```

MELHOR: 1 + n

PIOR: 1 + n

~~Θ~~(n)

b)

```
int soma1 = 0; ----------------------------------- 1
int soma2 = 0; ----------------------------------- 1
for (int i=0; i<n; i++){
  soma1 = soma1 + 1;--------------------------- 1*n
  soma2 = soma2 + i; --------------------------- 1*n
}
```

MELHOR: 2 + 2n

PIOR: 2 + 2n

~~Θ~~(n)

c)

```
int soma = 0;------------------------------------ 1
for (int i=0; i<n; i++){
  if ( vetor[i] % 2 == 0) //se for par ------------ 1*n
     soma = soma + vetor[i];-------------------- 1*n
}
```

MELHOR: 1 + n

PIOR: 1 + 2n

~~Θ~~(n)

d)

```
int soma1 = 0; ---------------------------------- 1
for (int i=0; i<n; i++){
  soma1 = soma1 + 1;---------------------------- 1*n
}
for (int j=0; j<n;j++){
  soma1 = soma1 + j; ---------------------------- 1*n
}
```

MELHOR:1 +2n
PIOR:1+2n
~~Θ~~(n)

e)

```
int soma = 0;  ---------------------------------- 1
for (int i=0; i<n; i++){
   for (int j=0; j<n; j++){
      soma = soma + 1;--------------------------- 1*n*n
   }
}
```

MELHOR: $1+n^2$
PIOR: $1+n^2$

~~Θ~~(n^2)

f)

```
int soma = 0;----------------------------------- 1
for (int i=0; i<n; i++){
   for (int j=0; j<m; j++){
      soma = soma + 1;--------------------------- 1*m*n
   }
}
```

MELHOR:1+n*m
PIOR: 1+n*m

~~Θ~~(nm)

g)

```
int menor = MAIOR-INTEIRO; ---------- 1
for (int i=0; i<n; i++){
  if (vetor[i] < menor) ---------- 1*n
     menor = vetor[i]; ---------- 1*n
}
```

MELHOR:2+n
PIOR:1+2n
~~Θ~~(n)

h)

```
int v[][] = new int[n][n]; ---------- 1

for (int i=0; i<n; i++){
  for (int j=0; j<n; j++){
     v[i][j] = i * j; ---------- 1*n*n
  }
}
```

MELHOR:$1+n^2$
PIOR: $1+n^2$

~~Θ~~(n^2)

i)

```
int menor = MAIOR-INTEIRO; ---------- 1
for (int i=0; i<n; i++){
  if (vetor[i] < menor) ---------- 1*n
     menor = vetor[i]; ---------- 1*n
}
if (menor < 0){ ---------- 1
   for (int i=0; i<n; i++){
        menor = menor * (i+1); ---------- 1*n
   }
}
```

MELHOR:3+n
PIOR:2+3n

~~Θ~~(n)

j)

```
int menor = MAIOR-INTEIRO; ---------- 1
for (int i=0; i<n; i++){
  if (vetor[i] < menor) ---------- 1*n
     menor = vetor[i]; ---------- 1*n
}
if (menor < 0){ ---------- 1
   for (int i=0; i<n; i++){
        menor = menor * (i+1);
   }
}else if (menor > 0){ ---------- 1
    for (int i=0; i<n*n; i++)
       printf("%d\n", menor);---------- 1*n*n
  } else {
        printf("%d\n", menor);
     }
```

MELHOR: 5 + n
PIOR:$3+2n+n^2$

Ω(n) e $O(n^2)$

2) Dado o método de busca a seguir, analise o tempo estimado de execução no **melhor** e no **pior** caso para cada um dos trechos de código, considerando o modelo RAM. Lembre que **size()** é um método que retorna a quantidade de elementos de uma lista.

```
Pessoa busca(String nome){
  for (int i = 0; i< pessoas.size(); i++){
       if (pessoas.get(i).getNome().equals(nome)) ---------- 1*n
          return pessoas.get(i);------------------- 1
  }
  return null;
}
```

MELHOR: 2
PIOR: 1 + n

Ω(1) e O(n)

a)

```
void exibir(String nome){
   Pessoa p = busca(nome);---------------------- 1 + (1+n)
   if (p != null){ ---------------------------------- 1
      p.exibirDados();--------------------------- 1
   }
  else{
       System.out.println("Pessoa não encontrada");
  }
}
```

MELHOR: 5
PIOR: 4 + n

Ω(1) e O(n)

b)

```
void exibir(String nome){
   if (busca(nome) != null){ ---------- (n+1)+1
      busca(nome).exibirDados();---------- (n+1)+1
   }
  else{
       System.out.println("Pessoa não encontrada");
  }
}
```

MELHOR:6
PIOR:2n+4

Ω(1) e O(n)

c)

```
void atualizar(String nome, int idade, float salario){
    Pessoa p = busca(nome);---------- 1+(n+1)
   if (p != null){ ---------- 1
      p.setIdade(idade);---------- 1
      p.setSalario(salario);---------- 1
   }
  else{
       System.out.println("Pessoa não encontrada");
  }
}
```

MELHOR:6
PIOR: 5 + n
Ω(1) e O(n)

d)

```
void atualizar(String nome, int idade, float salario){
    if (busca(nome) != null){ ---------- (n+1)+1
        busca(nome).setIdade(idade); ---------- (n+1)+1
        busca(nome).setSalario(salario);--------- (n+1)+1
    }
   else{
         System.out.println(“Pessoa não encontrada”);
   }
}
```

MELHOR: 9
PIOR: 3n + 6
Ω(1) e O(n)