

Exercícios

Capítulo 2. Análise de Complexidade de Algoritmos

2.1. Análise de Complexidade

2.1.1 Obtenha aproximações tilde para as seguintes expressões:

- a) $n + 1$
- b) $1 + 1/n$
- c) $(1 + 1/n)(1 + 2/n)$
- d) $2n^3 - 15n^2 + n$
- e) $\log(2n)/\log(n)$
- f) $\log(n^2 + 1)/\log(n)$
- g) $n^{100}/2^n$

2.1.2 Indique uma aproximação tilde para a ordem de crescimento (em função de n) para a instrução mais frequente de cada um dos seguintes excertos de código. Indique também a ordem de crescimento assintótica para cada excerto:

a)

```
int sum = 0;
for(int i = 0; i < n; i++ )
{
    for(int j = 1; j < 10 ; j++)
    {
        sum++;
    }
}
```

b)

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

c)

```
int sum = 0;
for(int k = n; k > 0; k/= 2 )
{
    for(int j = 0; j < k ; j++)
    {
        sum++;
    }
}
```

d)

```
int sum = 0;
for(int i = 1; i <= n; i*= 2 )
{
    for(int j = 0; j < i ; j++)
    {
        sum++;
    }
}
```

e)

```
int sum = 0;
for(int i = 1; i <= n; i*= 2 )
{
    for(int j = 0; j < n ; j++)
    {
        sum++;
    }
}
```

2.2. Análise Empírica

2.2.1. Utilize o ensaio de razão dobrada para determinar a ordem de crescimento do método 2-sum, e do método 3-sum. Indique as respectivas ordens de crescimento.

2.2.2. Estime o tempo de execução do método 2-sum, e do método 3-sum no seu computador para quando $n = 16\,000$.

2.2.3. Escreva um programa que dado um inteiro n , utiliza uma distribuição uniforme para gerar uma sequência de inteiros entre 0 e $n-1$. Execute experiências para validar a hipótese de que o número de inteiros gerados antes do primeiro valor repetido ser encontrado é $\sim \sqrt{\pi \frac{n}{2}}$