Lista de Exercícios 1

complexidade de algoritmos

Algoritmos e Estrutura de Dados 2 (AE23CP)

DAINF DEPARTAMENTO ACADÊMICO DE INFORMÁTICA CÂMPUS PATO BRANCO

UTFPR UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Prof. Jefferson T. Oliva

1. Seja a seguinte definição: "Dadas duas funções, $f(n)$ e $g(n)$, diz-se que $f(n)$ é da ordem de $g(n)$ ou que $f(n)$ é $O(g(n))$, se existirem inteiros positivos a e b tais que $f(n) \leq a * g(n)$ para todo $n \geq b$." Verifique se as seguintes proposições estão corretas:

    1. $7 \in O(n)$ F
    2. $n \in O(1)$ V
    3. $n + 7 \in \Theta(1)$ F
    4. $n + 7 \in \Omega(1)$ V
    5. $n^2 + 2 \in O(n)$ F
    6. $n^2 + 2 \in \Omega(n)$ V
    7. $n + 2 \in O(n^2)$ V
    8. $2n^4 \in O(n^4)$ V
    9. $2n^4 \in \Omega(n^4)$ V
    10. $2n^4 \in \Theta(n^4)$ V
    11. $\log n \in O(1)$
    12. $\log n \in \Theta(1)$
    13. $\log n + 1 \in O(logn)$
    14. $\log n + 1 \in \Omega(logn)$
    15. $\log n + 1 \in O(n)$
    16. $\log n + 1 \in \Omega(n^2)$
    17. $\log n + 1 \in O(n^3)$
    18. $n \log n \in O(1)$ F
    19. $n \log n \in \Omega(1)$ V
    20. $n \log n + 1 \in \Theta(\log n)$ F
    21. $n \log n + 1 \in \Omega(n^2)$ F
    22. $n \log n + 1 \in O(n^3)$
    23. $2 \log n \in \Theta(n \log n)$
    24. $2n + n \in O(2^3)$ F
    25. $n^n \in O(2^n)$ F
    26. $n^{100} \in O(n^n)$ V
    27. $2n^n \in \Omega(n^n)$

 Continua na próxima página...

2. Compare as duas funções n^2 e $\frac{2^n}{4}$ para vários valores de n. Determine quando a segunda se torna maior que a primeira.

3. Compare as duas funções $4n + 1$ e n^2 para vários valores de n. Determine quando a segunda se torna maior que a primeira.

4. Determine a quantidade máxima de instruções para os seguintes segmentos de código:

(a)
```
for (i = n - 1; i >= 0; i--)
  x -= (v[i] / 2);
```

(b)
```
for (i = 0; (i < n) && !(achou); i++)
  if (x == v[i])
    achou = 1;
```
5n + 4

(c)
```
if (a < b)
  for (i = 0; i <= n / 2; i++);
    resp += v[i];
else
  for (i = (n / 2) + 1; i < n; i++);
    resp += v[i];
```

(d)
```
for (i = 0; i < n; i++)
  for (j = 0; j <= i; j++)
    x = x + 1;
```
$2n^2 + 12$
Complexidade: $O(n^2)$

5. Determine a quantidade máxima de instruções e a complexidade no pior caso para os seguintes algoritmos. Também, determine a quantidade de unidades de espaço extras necessárias (desconsiderando o conjunto de entrada) para execução dos algoritmos.

(a)
```
float func(float x, float v[], int n){
  int k;
  float y = 0.0;
  for (k = n - 1; k >= 0; k--)
    y += v[k] / x;
  return y;
}
```

(b)
```
void sort(int v[], int n){
  int k, j, aux;
  for (k = 1; k < n; k++)
    for (j = 0; j < n - 1; j++)
      if (v[j] > v[j + 1]){
        aux = v[j];
        v[j] = v[j + 1];
        v[j + 1] = aux;
      }
}
```

(c)
```
void selection_sort(int v[], int n){
  int i, j, min, aux;
  for (i = 0; i < (n - 1); i++){
    min = i;
    for (j = (i+1); j < n; j++){
      if(v[j] < v[min])
        min = j;
    }
    if (v[i] != v[min]){
      aux = v[i];
      v[i] = v[min];
      v[min] = aux;
    }
  }
}
```

 Continua na próxima página. . .

(d)
```
int faz_algo(int *v, int i, int j){
   if (i < j)
       return v[i] + v[j] + fal_algo(v, i + 1, j + 1);
   else if (i == j)
       return v[i];
   else
       return 0;
}
```

(e)
```
#define min(a, b) b < a ?  b :  a
int menor(int *v, int ini, int fim){
      if ((fim - ini) <= 1)
        return min(v[ini], v[fim]);
     else{
        int t = (fim + ini) / 2;
        return min(menor(v, ini, t), menor(v, t + 1, fim));
      }
}
```

6. Dada a função $T(n) = 6n^3 + 2n^2 + 3n + \log n$. Verifique se as seguintes informações são verdadeiras ou falsas:

   a) - $T(n) = O(n \log n)$
   b) - $T(n) = \Omega(logn)$
   c) - $T(n) = \Theta(n^3)$
   d) - $T(n) = O(n^3)$
   e) - $T(n) = O(1)$

7. Escreva um algoritmo que verifica se o vetor está em ordem crescente. Faça a análise de complexidade de tempo e de espaço do algoritmo.

Continua na próxima página…

8. Resolva as seguintes recorrências:

(a)

$$T(n) = \begin{cases} 1, & se\, n \leq 1 \\ T(n-2)+1, & se\, n > 1 \end{cases}$$

(b)

$$T(n) = \begin{cases} 1, & se\, n < 1 \\ T(n-1)+n^2, & se\, n \geq 1 \end{cases}$$

(c)

$$T(n) = \begin{cases} 1, & se\, n = 1 \\ T(n-1)+2n+1, & se\, n > 1 \end{cases}$$

(d)

$$T(n) = \begin{cases} 1, & se\, n = 1 \\ T(n-1)+(n-1), & se\, n > 1 \end{cases}$$

(e)

$$T(n) = 9T(\frac{n}{3}) + n$$

(f)

$$T(n) = 2T(\frac{n}{3}) + n + 1$$

(g)

$$T(n) = 2T(\frac{n}{4}) + n^2$$

(h)

$$T(n) = 16T(\frac{n}{4}) + 2n$$

(i)

$$T(n) = \begin{cases} 1, & se\, n = 1 \\ 3T(n-1), & se\, n > 1 \end{cases}$$

(j)

$$T(n) = \begin{cases} 1, & se\, n = 0 \\ nT(n-1), & se\, n > 0 \end{cases}$$

 Continua na próxima página…

References

[1] Horowitz, E., Sahni, S. Rajasekaran, S. *Computer Algorithms*, Comp Sc Press, 1998.

[2] Rosa, J. L. G. Análise de Algoritmos. SCC-0201 – Introdução à Ciência da Computação II. *Lista de Exercícios*. Ciência de Computação. ICMC/USP, 2008.

[3] Silva, R. C. Notações Assintóticas e Recorrências. Projeto e Análise de Algoritmos. *Lista de exercícios*. Unioeste/Foz do Iguaçu, 2010.

[4] Szwarcfiter, J.; Markenzon, L. *Estruturas de Dados e Seus Algoritmos*. LTC, 2010.

[5] Tenenbaum, A.; Langsam, Y. *Estruturas de Dados usando C*. Pearson, 1995.

[6] Ziviani, N. *Projeto de Algoritmos - com implementações em Java e C++*. Thomson, 2007.