Universidade Federal do ABC

CMCC Centro de Matemática, Computação e Cognição

Algoritmos e Estruturas de Dados I

Lista de Exercícios 3: Complexidade de algoritmos

Profa. Mirtha Lina Fernández Venero
Prof. Paulo Henrique Pisani

1 de março de 2019

1 Notações assintóticas

1. Ordene as seguintes funções por ordem de crescimento assintótico de forma não decrescente. Preencha a tabela com os números correspondentes à ordem de crescimento. Use o mesmo número caso duas funções tenham a mesma ordem de crescimento, i.e. $f(n) = \Theta(g(n))$. Justifique de forma apropriada sua ordenação.

Função	4^n	$n \log n$	3^n	$4^{\log_2 n}$	$n^{4/7}$	5000	$5n^2 + 8$	$(n-1)!$	$\log^4(n)$
Ordem									

2. Responda se as seguintes afirmações são verdadeiras ou falsas. Justifique de forma apropriada sua resposta.

   (a) $100n^3 = \Omega(n^4 - 300n)$
   (b) $2^{5n} = O(2^n)$
   (c) $3^{\sqrt{n}} = O(3^n)$
   (d) $\log(n^2) = \Theta(\log n)$
   (e) $n^{1/3} = \Theta(\sqrt{n})$
   (f) $f(n) - g(n) = O(min(f(n), g(n))$
   (g) $f(n) - g(n) = \Omega(g(n))$

2 Complexidade de algoritmos

1. Qual a complexidade assintótica das seguintes funções?

```
3  int a(int n, int *p){
4    int i, j, count = 0;
5    if (p)
6      for( i = 0; i < n; i = i+2)
7        for( j = n-i ; j >=0; j--)
8          count++;
9    else
10     for( i =1; i < n-1; i++)
11       for( j =1; j < 2*n; j++)
12         count++;
13
14   return count;
15 }
16
17 int b(int n){
18   int i, j, count = 0;
19   for( i=n; i>=1; i/=2 )
20     for( j=1; j<=n*n; j++ )
21       count++;
22   return count;
23 }
24
25 int c(int n){
26   int i, j, k, count = 0;
27   for( i = n/2 ; i<=n; i++ )
28     for( j=1 ; j<=n; j *= 2 )
29       for( k=1; k<=n; k*=2 )
30         count++;
31   return count;
32 }
33
34 int d(int n, int m, int *p){
35   if (a(n,p) > m)
36     return b(n+m);
37   else
38     return c(n*m);
39 }
```

2. Qual a complexidade assintótica das seguintes recorrências e funções recursivas?

   (a) $T(n) = 9T(n/2) + n^2$

(b) $T(n) = 4T(n/2) + n^2\sqrt{n}$

(c) $T(n) = 3T(n-5) + n$

(d) $T(n) = 3T(n/3) + \sqrt{n}$

```
41  void e(int n){
42    if (n<=1)
43      return;
44    int i = 1;
45    for( ; i<=n; i++ )
46      printf("*");
47    e(0.6*n);
48  }
49
50  void f(int n){
51    if (n<=2)
52      return;
53    int i = 1, count = 0;
54    for( ; i<=8; i++ )
55      f(n/2);
56    for( i=1 ; i<=n*n*n; i++ )
57      count ++;
58  }
59
60  void fractal(float x, float y, float r, float m)
61  {
62    if( r <= m )
63     return;
64    fractal(x-r, y+r, r/2, m);
65    fractal(x+r, y+r, r/2, m);
66    fractal(x-r, y-r, r/2, m);
67    fractal(x+r, y-r, r/2, m);
68    pintarQuadrado(x, y, r);    // Assuma custo O(n^2)
69  }
70
71  void printNumBin(int x)
72  {
73    if( x == 0 || x == 1)
74      printf("%d", x);
75    else{
76     printNumBin(x / 2);
77     printf("%d", x % 2);
78    }
79  }
```

3. Se deseja implementar um TDA para **multiconjuntos** (i.e. conjuntos onde há repetições de elementos) no universo $\mathcal{U} = \{1, ..., 100\}$.

   (a) Defina as estruturas de dados necessárias para representar esse TDA usando i) vetores e ii) listas ligadas.

   (b) Implemente as operações da primeira coluna da seguinte tabela usando as duas estruturas definidas acima.

   (c) Preencha as restantes colunas da tabela o tamanho da entrada e a complexidade de tempo no caso pior das operações. Argumente de forma apropriada suas respostas.

Operação	Tamanho Entrada	Vetores	Listas Ligadas
Pertinência dum elemento			
União			
Interseção			
Diferença			
Inserir elemento (ocorrência)			
Eliminar um elemento (ocorrência)			