Algoritmos

Introdução

Lista de exercícios

1. Defina algoritmo.
2. Explique, com exemplos, as 5 características de um algoritmo.
3. Explique de que formas podemos representar algoritmos.
4. Das formas de representar algoritmos explique, justificando, qual a mais adequada para o desenvolvimento de programas.
5. Explique o que é **eficiência** de um algoritmo e porque seu estudo é importante para o desenvolvimento de programas de computador.
6. Escreva um programa que leia um número inteiro n ($1 \leq n \leq 10^{15}$) e mostre se n é primo.
   (a) Verifique o resultado e o tempo de execução do seu programa para cada um dos seguintes valores: $13, 21, 83, 87, 103, 111, 6471, 6473, 11021, 11027, 32531, 32567, 111467$, $2040010289, 2040010291, 61553013897, 62558003897, 62558013447, 91989664471$ e 91989664481.
   (b) Colque o resultado em uma tabela:

Valor	**Tempo**	**Resultado**
13	$0ms$	Primo
...		

   (c) Escreva um texto explicando o resultado obtido na tabela anterior e apontando estratégias de melhoria no desempenho do seu programa.
7. Considere as equações de tempo de execução dos algoritmos A e B a seguir
   - $T(A) = 100n + 500$
   - $T(B) = 2n^2 + 10$

   onde n é o tamanho da entrada.

   Escreva uma análise comparativa sobre o desempenho de A e B em função do tamanho da entrada. Para realizar a análise forneça as informações a seguir.
   (a) Determine para quais faixas de valores A é melhor do que B e B é melhor do que A, considerando eficiência temporal.
   (b) Desenhe o gráfico das funções dos algoritmos A e B.
8. Escreva um algoritmo que encontre o menor elemento de *array* de n elementos. Encontre a equação do tempo de execução do algoritmo.
9. Escreva um algoritmo que determine se um valor V encontra-se em um *array* de n elementos. Escreva a equação de tempo de execução do seu algoritmo.
10. Sejam 4 diferentes algoritmos com as equações de tempo de execução a seguir:
    (a) $\frac{2^n}{8} + 2n$
    (b) $10n + 50$
    (c) $10 + 2n^2 + n$

(d) $100 + 20\log_2 n$

Preencha a tabela a seguir e classifique os algoritmos do melhor (mais rápido) para o pior.

Entrada	2	4	6	8	10	12	14	16	18	20
Algo a										
Algo b										
Algo c										
Algo d										

Faça um gráfico com as 4 equações para facilitar a visualização.

11. Uma algoritmo realiza $n^2 + 2^{n+2}$ operações primitivas até terminar, no pior caso. Determine a sua complexidade (taxa de crescimento) usando a notação O. Explique como você chegou a esse resultado.

12. Ordene as funções a seguir pela ordem de complexidade: n^2, n, $\log_2 n$, 2^n, $n\log_2 n$, n^3.

13. Para cada uma das equações a seguir, determine sua complexidade usando a notação O.

    (a) $\frac{2^n}{8} + 2n$
    (b) $10n + 50$
    (c) $10 + 2n^2 + n$
    (d) $100 + 20\log_2 n$
    (e) $50n + 2^n + 200$
    (f) $1000 + 3n\log_2 n + 300n$
    (g) $\log_2 n + 5n$
    (h) $n^2 - 400n + 50n^3$

14. Considere 5 diferentes algoritmos que resolvem o mesmo problema. Todos os algoritmos levam $5s$ (cinco segundos) para terminar com uma entrada de tamanho 40. Após realizar uma análise você descobriu que cada algoritmo possui uma complexidade diferente. Determine em quanto tempo cada algoritmo termina para uma entrada de tamanho 80, considerando as seguintes complexidades:

    (a) 1
    (b) $\log_2 n$
    (c) n
    (d) n^2
    (e) 2^n

15. Considere 4 diferentes algoritmos que resolvem o mesmo problema. Todos os algoritmos levam $2s$ (dois segundos) para terminar com uma entrada de tamanho 10. Após realizar uma análise você descobriu que cada algoritmo possui uma complexidade diferente. Determine o tamanho da entrada para que o algoritmo termine em 15 segundos, considerando as seguintes complexidades:

    (a) n
    (b) $\log_2 n$
    (c) n^2
    (d) 2^n

16. Forneça uma estimativa do tempo de execução, usando a notação O, em função de n para as seguintes funções em linguagem C:

(a)
```
int fA(int n){
  int i,a=1;
  for ( i=0 ; i<n ; i++ ){
     a += i;
  }
  return a;
}
```

(b)
```
int fB(int n){
  int i, a=1;
  for ( i=1 ; i<=n ; i*=2 ){
     a += i;
  }
  return a;
}
```

(c)
```
int fC(int n){
  int i,j,x=1,y;
  for ( i=0 ; i<n ; i++ ){
    y = 2;
    for ( j=1 ; j<=n ; j++ ){
      y+=j;
    }
    x*=y;
  }
  return x;
}
```

(d)
```
int fD(int n){
  int i,x=n,y=0;
  for ( i=0 ; i<512 ; i++ ){
    y+=x;
    if ( i%7==0 && i<2*n )
      x++;
  }
  return y;
}
```

(e)
```
int fE(int n){
  int i,j,x=1,y=0;
  for ( i=n ; i>0 ; i=i/2 ){
    y+=x;
    for ( j=0 ; j<n/2 ; j++ ){
      if ( n%j==0 )
        y++;
    }
  }
  return y;
}
```