Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Projeto e Análise de Algoritmos
Profª. Raquel Mini

1ª LISTA DE EXERCÍCIOS

1. São dados $2n$ números distintos distribuídos em dois vetores com k elementos A e B ordenados de maneira tal que:

   $A[1] > A[2] > A[3] > \ldots > A[n]$ e $B[1] > B[2] > B[3] > \ldots > B[n]$

   Apresente um algoritmo linear para encontrar o k-ésimo maior número dentre estes $2n$ elementos.

2. Considere o problema de encontrar a posição de inserção de um novo elemento em um conjunto ordenado:

   $$A[1] > A[2] > A[3] > \ldots > A[n]$$

   a) Apresente a situação e/ou entrada de dados em que ocorre o melhor caso e o pior caso.
   b) Apresente um algoritmo para resolver o problema acima.

3. Dada uma lista ordenada de n elementos de valor inteiro, o problema de unificação de lista consiste em realizar seguidamente a operação de remover os dois elementos de menor valor da lista e inserir um novo elemento com valor igual a soma dos dois primeiros de forma que a lista continue ordenada. A cada operação a lista passa a ter um elemento a menos. A unificação termina quando restar somente um elemento na lista.

   a) Apresente um algoritmo que realiza a unificação da lista em tempo $O(n)$.
   b) Apresente a função de complexidade de tempo para o melhor caso e forneça um exemplo de entrada de dados deste caso.
   c) Apresente a função de complexidade de tempo para o pior caso e forneça um exemplo de entrada de dados deste caso.
   d) É possível realizar a unificação da lista em tempo sublinear? Justifique sua resposta.

4. Indique se as afirmativas a seguir são verdadeiras ou falsas e justifique a sua resposta.
   a) $2^{n+1} = O(2^n)$
   b) $2^{2n} = O(2^n)$
   c) $f(n) = O(u(n))$ e $g(n) = O(v(n)) \Rightarrow f(n) + g(n) = O(u(n) + v(n))$

5. Indique para cada par de expressões (A,B) na tabela abaixo, se A é O, o, Ω, ω ou Θ de B. Assuma que $k \geq 1$ e $0 < \varepsilon < 1 < c$ são constantes. Sua resposta deve ser da forma SIM ou NÃO.

   Nota: $\log^k n = \underbrace{\log\log\cdots n}_{k}$ e $n! \approx \left(\frac{n}{e}\right)^n$.

	A	B	O	o	Ω	ω	Θ
(i)	$\log^k n$	n^ε					
(ii)	n^k	c^n					
(iii)	c^n	$c^{n/2}$					
(iv)	$\log(n!)$	$\log(n^n)$					
(v)	$\log^{k+1} n$	$\log^k n$					
(vi)	c^ε	$(c+1)^\varepsilon$					

6. Sejam as seguintes funções:

$g_1 = n^{\frac{1}{\log n}}$ $\quad g_2 = \ln\ln n$ $\quad g_3 = (\ln n)^2$ $\quad g_4 = n$

$g_5 = 2^{\log n}$ $\quad g_6 = n\log n$ $\quad g_7 = \log(n!)$ $\quad g_8 = n^2$

$g_9 = 4^{\log n}$ $\quad g_{10} = \left(\frac{3}{2}\right)^n$ $\quad g_{11} = 2^n$ $\quad g_{12} = e^n$

e os seguintes fatos ($a > 0$, $b > 0$, $c > 0$, $n \in \Re$):

$\log n = \log_2^n$ $\quad \ln n = \log_e^n$

$a = b^{\log_b^a}$ $\quad \log_c^{(ab)} = \log_c^a + \log_c^b$

$\log_c^{a^n} = n\log_b^a$ $\quad \log_b^a = \frac{\log_c^a}{\log_c^b}$

$\log_b^a = \frac{1}{\log_a^b}$ $\quad a^{\log_b^n} = n^{\log_b^a}$

$n^{\frac{1}{\log n}} = n^{\log_n^2} = 2$ $\quad 2^{\log n} = n$

$4^{\log n} = 2^{2\log n} = 2^{\log n^2} = n^2$ $\quad n! \approx \left(\frac{n}{e}\right)^n$

$\log(n!) = \Theta(n\log n)$

Pede-se: Mostre para cada par de funções g_i e g_{i+1} para $1 \leq i \leq 11$ se g_i é o ou Θ de g_{i+1}.

7. Avalie as seguintes somas:
   a) $\sum_{i=1}^{n} i$
   b) $\sum_{i=1}^{n} a^i$
   c) $\sum_{i=1}^{n} ia^i$
   d) $\sum_{i=1}^{n} \frac{1}{i}$
   e) $\sum_{i=1}^{n} i2^{-i}$
   f) $1 + 1/7 + 1/49 + \cdots + (1/7)^n$
   g) $\sum_{i=1}^{n} \frac{1}{i(i+1)}$

8. Considerando que a operação relevante é o número de vezes que a operação soma é executada, apresente a função de complexidade de tempo para:

   a)
```
for i ← 1 to n do
   for j ← 1 to n do
      for k ← 1 to n do
         temp ← temp + i + j + k
```

   b)
```
for i ← 1 to n do
   for j ← 1 to i do
      for k ← 1 to j do
         temp ← temp + i + j + k
```

   c)
```
for i ← 1 to n do
   for j ← 1 to n do
      for k ← i to n do
         temp ← temp + i + j + k
```

   d)
```
for i ← 1 to n do
   for j ← i to n do
      for k ← i to n do
         temp ← temp + i + j + k
```

   e)
```
for i ← 1 to n do
   for j ← i to n do
      for k ← i to j do
         temp ← temp + i + j + k
```

9.

a) O que a função abaixo faz?
b) Qual é a operação relevante?
c) Qual é sua função de complexidade?

```
void p2 (int n)
{
   int i, j, x, y;

   x = y = 0;
   for (i=1; i<=n; i++) {
       for (j=i; j<=n; j++)
            x = x + 1;
       for (j=1; j<i; j++)
            y = y + 1;
   }
   cout<<x<<" "<<y;
}
```

10. Qual é a função de complexidade no pior caso para o número de atribuições ao vetor x?

```
void Exercicio3(int n){
   int i, j, a;

   for (i=0; i<n; i++){
      if (x[i] > 10)
         for (j=i+1; j<n; j++)
            x[j] = x[j] + 2;
      else {
         x[i] = 1;
         j = n-1;
         while (j >= 0) {
            x[j] = x[j] - 2;
            j = j - 1;
         }
      }
   }
}
```

11. Resolva as seguintes equações de recorrência:

a) $\begin{cases} T(n) = T(n-1) + c & \text{c constante, } n > 1 \\ T(1) = 0 \end{cases}$

b) $\begin{cases} T(n) = T(n-1) + 2^n & n \geq 1 \\ T(0) = 1 \end{cases}$

c) $\begin{cases} T(n) = cT(n-1) & \text{c, k constantes, } n > 0 \\ T(0) = k \end{cases}$

d) $\begin{cases} T(n) = 3T(n/2) + n & n > 1 \\ T(1) = 1 \end{cases}$

e) $\begin{cases} T(n) = 3T(n-1) - 2T(n-2) & n > 1 \\ T(0) = 0 \\ T(1) = 1 \end{cases}$

12. Considere o algoritmo a seguir. Suponha que a operação crucial é o fato de inspecionar um elemento. O algoritmo inspeciona os *n* elementos de um conjunto e, de alguma forma, isso permite descartar 2/5 dos elementos e então fazer uma chamada recursiva sobre os 3*n*/5 elementos restantes.

```
procedure Pesquisa (n: integer);
begin
   if n <= 1 then
      'inspecione elemento' e termine
   else
   begin
      para cada um dos n elementos 'inspecione elemento';
      Pesquise (3n/5);
   end
end
```

a) Escreva uma equação de recorrência que descreva este comportamento.
b) Converta esta equação para um somatório.
c) Dê a fórmula fechada para este somatório.

13. Considere o algoritmo abaixo.

```
procedure Sort2 (var A: array[1..n] of integer; i,j: integer);
{-- n uma potencia de 3 --}
begin
   if i < j then
   begin
      k := ((j-i)+1)/3;
      Sort2(A,i,i+k-1);
      Sort2(A,i+k,i+2k-1);
      Sort2(A,i+2k,j);
      Merge(A,i,i+k,i+2k,j);
      { Merge intercala
        A[i…(i+k-1)], A[(i+k)..(i+2k-1)] e
        A[i+2k..j] em A[i..j]
        a um custo 5n/3-2}
   end
end
```

a) Escreva uma equação de recorrência que descreva este comportamento.
b) Converta esta equação para um somatório.
c) Dê a fórmula fechada para este somatório.

14. Para o problema a seguir apresente a recorrência e a forma fechada. **Torre de Hanói**. Em 1883, o matemático francês Edouard Lucas criou um jogo chamado Torre de Hanói. O jogo começa com um conjunto de oito discos empilhados em tamanho decrescente em uma das três varetas, conforme mostrado na Figura 1. O objetivo do jogo é transferir toda a torre para uma das outras varetas, movendo um disco de cada vez, mas nunca movendo um disco maior sobre um menor.

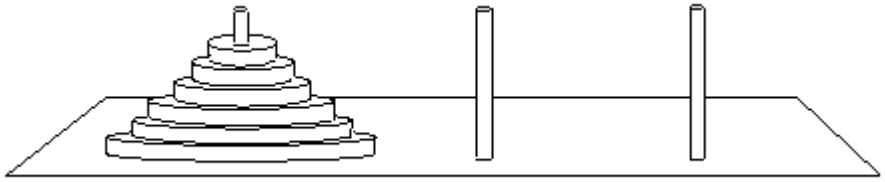

Figura 1: Configuração inicial da Torre de Hanói.

15. Para o problema a seguir apresente a recorrência e a forma fechada. **Linhas no plano ou Cortando a sua pizza favorita**. Quantas fatias de pizza uma pessoa pode obter ao fazer *n* cortes retos com uma faca? Ou, expressando de outra forma, qual é o número máximo de regiões L_n determinado por *n* retas no plano? Lembre-se que um plano sem nenhuma reta tem uma região, com uma reta tem duas regiões e com duas retas tem quatro regiões, conforme mostrado na figura 2.

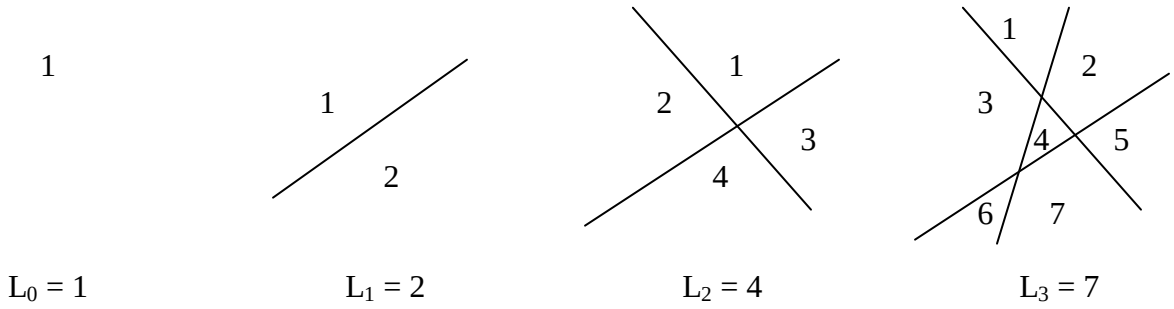

Figura 2: Regiões no plano.

16. Apresente a complexidade de tempo para os procedimentos abaixo:

a)
```
PROCEDURE Pesquisa (n: integer);
BEGIN
   IF n > 1 THEN
   BEGIN
      Inspecione n*n*n elementos;
      Pesquisa (2n/3);
   END;
END;
```

b)
```
PROCEDURE T (n: integer);
BEGIN
   IF n = 0 THEN
      RETURN 1
   ELSE
      RETURN T(n-1)+T(n-1)
END;
```

c)
```
PROCEDURE D (VAR L);
BEGIN
   IF |L| > 1 THEN
   BEGIN
      Divide L em 3 partes iguais L1, L2, L3 com custo de |L|*|L| passos;
      D(L1); D(L2); D(L3);
   END;
END;
```

17. Use o teorema mestre para derivar um limite assintótico Θ para as seguintes recorrências:
    a) $T(n) = 2T(n/2) + n - 1$
    b) $T(n) = 3T(n/2) + n$
    c) $T(n) = 4T(n/2) + n^2$
    d) $T(n) = 4T(n/2) + n^3$

18. O tempo de execução de um Algoritmo `A` é descrito pela recorrência $T(n) = 7\ T(n/2) + n^2$. Um outro algoritmo `A'` tem um tempo de execução descrito pela recorrência $T'(n) = a\ T'(n/4) + n^2$. Qual é o maior valor inteiro de *a* tal que `A'` é assintoticamente mais rápido que `A`?

19. Considerando que no algoritmo abaixo a operação relevante seja o número comparações com os elementos do vetor A, responda às seguintes perguntas.

```
void SORTSORT(int Esq, int Dir, int A[]) {
   int i, j, x, w;

   i = Esq;
   j = Dir;
   x = A[(i + j) / 2];
   do {
      while (x > A[i]) i++;
      while (x < A[j]) j--;
      if (i <= j) {
         w = A[i]; A[i] = A[j]; A[j] = w;
         i++; j--;
      }
    } while (i <= j);
    if (Esq < j)
       SORTSORT(Esq, j, A);
    if (i < Dir)
       SORTSORT(i, Dir, A);
}
```

    a) Qual é a situação que leva ao melhor caso desse algoritmo?
    b) Escreva a equação de recorrência que descreva o seu comportamento no melhor caso.

c) Converta esta equação de recorrência para um somatório.
d) Forneça a fórmula fechada para este somatório.

20. Para cada equação de recorrência responda: (i) o teorema mestre pode ser aplicado à esta recorrência? Justifique; (ii) forneça um limite assintótico para a recorrência utilizando o teorema mestre, se possível; (iii) Se não for possível resolver pelo teorema mestre, resolva a recorrência.

a) $\begin{cases} T(n) = 2T\left(\frac{n}{2}\right) + n\log n \\ T(1) = 0 \end{cases}$

b) $\begin{cases} T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n} \\ T(1) = 1 \end{cases}$

c) $\begin{cases} T(n) = 3T\left(\frac{n}{2}\right) + n \\ T(1) = 1 \end{cases}$