SSC0503 - Introdução à Ciência de Computação II
2ª Lista
Professor: Claudio Fabiano Motta Toledo (`claudio@icmc.usp.br`)
Estagiário PAE: Jesimar da Silva Arantes (`jesimar.arantes@usp.br`)

---

1. Desenvolva algoritmos recursivos para os seguintes problemas:
   - Impressão de um número natural em base binária.
   - Multiplicação de dois números naturais, através de somas sucessivas (Ex.: $6.4 = 4+4+4+4+4+4$).
   - Cálculo de $1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N}$
   - A partir de um vetor de números inteiros, calcule a soma e o produto dos elementos do vetor.
   - Gerador de máximo divisor comum (mdc):
     - $mdc(x, y) = y$, se $x \geq y$ e $xmody = 0$.
     - $mdc(x, y) = mdc(y, x)$, se $x < y$.
     - $mdc(x, y) = mdc(y, xmody)$, caso contrário.
   - Verifique se uma palavra é palíndromo.
   - Dado um número n, gere todas as possíveis combinações com as n primeiras letras do alfabeto.
2. Usando o método da substituição, prove que:
   - T(n) = T(n - 1) + c é ???, c constante, $n > 1$ e $T(1) = 0$. Dica: expanda e conjecture, antes de verificar.
   - T(n) = cT(n - 1) para $n > 0$ é ??? com $T(0) = k$, c e k constantes. Dica: expanda e conjecture, antes de verificar.
   - $T(n) = 3T(n/2) + n$ para $n > 1$ é ??? com $T(1) = 1$. Dica1: resolva a recorrência para $n = 2^k$.
   - $T(n) = 3T(n/3) + 1$ para $n > 1$ é ??? com $T(1) = 1$. Dica1: resolva a recorrência para $n = 3^k$.
   - $T(n) = T(n-1) + n$ é $O(n^2)$.
   - $T(n) = T(\lceil n/2 \rceil) + 1$ é $O(lgn)$.
   - $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $\Theta(nlgn)$. Dica: Provamos em sala que $T(n) = O(nlgn)$.
   - $T(n) = T(\lfloor n/2 \rfloor + 17) + n$ é $O(nlgn)$.
3. Use árvore de recursão para determinar um bom limitante assintótico superior e o método da substituição para verificar a resposta.
   - $T(n) = 3T(n/2) + n$.

- $T(n) = T(n/2) + n^2$.
- $T(n) = 4T(n/2 + 2) + n$.
- $T(n) = 2T(n - 1) + 1$.
- $T(n) = T(n - 1) + T(n/2) + n$.
- $T(n) = 4T(n/2) + cn$ para c constante.

4. 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 $2n/5$ dos elementos e então fazer uma chamada recursiva sobre os $3n/5$ elementos restantes.

```
void Pesquisa (int n)
{
 if (n <= 1)
    'inspecione elemento' e termine;
 else {
    para cada um dos n elementos 'inspecione o elemento';
    Pesquisa (3n/5);
    }
}
```

   - Faça a análise de complexidade da função Pesquisa.

5. Considere o seguinte algoritmo recursivo que devolve a soma dos primeiros n cubos.

```
void Cubo (int n)
{
 if (n = 1)
    return 1;
 else {
    return Cubo (n-1) + n*n*n;     }
}
```

   - Faça a análise de complexidade da função Cubo.

6. Implemente uma função que exibe todas as substrings de uma string. Dica: primeiro enumere todos os substrings que começam com o primeiro caractere. Existem n deles se o string tem tamanho n. Então, enumere as substrings da string após remover o primeiro caractere. Exemplo: substrings de rum: r, ru, rum, u, um, m.

7. Implemente as versões recursiva e iterativa da função para obter números de Fibonacci.

   - Utilize a biblioteca time.h para medir e observar o tempo necessário para calcular n = 15; 30; 45 e 60, utilizando as duas versões.
   - Faça a análise de complexidade da versão recursiva.
   - Faça a análise de complexidade da versão iterativa. Dica: obtenha uma fórmula pela contagem de operações e encontre constantes de forma a mostrar o limite assintótico superior.
   - Mostre que a versão iterativa é $\Theta(n)$.