Fundamentos da Programação

lista de exercícios 11

1. **Números de Fibonacci**

   Os dois primeiros números de Fibonacci são

   $$f_0 = 0 \qquad\qquad f_1 = 1$$

   E os próximos são obtidos pela seguinte fórmula

   $$f_n = f_{n-2} + f_{n-1}$$

   Abaixo nós temos o cálculo de mais alguns números de Fibonacci

   - $f_2 = f_0 + f_1 = 0 + 1 = 1$
   - $f_3 = f_1 + f_2 = 1 + 1 = 2$
   - $f_4 = f_2 + f_3 = 1 + 2 = 3$
   - $f_5 = f_3 + f_4 = 2 + 3 = 5$
   - $f_6 = f_4 + f_5 = 3 + 5 = 8$

   e por aí vai ...

   a) Faça um programa recursivo que calcula o n-ésimo número de Fibonacci para um n qualquer.

   b) Qual é o maior número de Fibonacci que você consegue calcular com o seu programa?

2. **Regras de divisibilidade**

   a) A regra de divisibilidade do 11 é assim

      - Some os dígitos das posições pares
        Some os dígitos das posições ímpares
        E subtraia uma coisa da outra.
        Se o resultado for divisível por 11, então é
        Se não for, então não é.

   Por exemplo,

   $$1234567 \quad \Rightarrow \quad (1 + 3 + 5 + 7) - (2 + 4 + 6) = 4$$

   logo o número `1234567` não é divisível por 11.

   Faça um programa recursivo que implementa essa regra.

b) A regra de divisibilidade do 7 é assim

- Multiplique o dígito das unidades por 2

  E subtraia o resultado do número formado pelos outros dígitos

  Repita essa operação até obter um número menor que 100.

  Se o resultado for divisível por 7, então é

  Se não for, então não é.

A aplicação dessa regra ao número

1 2 3 4 5 6 7

produz o seguinte resultado

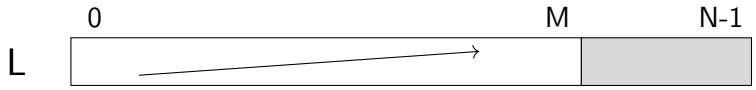
- 1 2 3 4 5 6 − 2 × 7 = 1 2 3 4 4 2
- 1 2 3 4 4 − 2 × 2 = 1 2 3 4 0
- 1 2 3 4 − 2 × 0 = 1 2 3 4
- 1 2 3 − 2 × 4 = 1 1 5
- 1 1 − 2 × 5 = 1

Logo, o número 1234567 não é divisível por 11.

Faça um programa que implementa essa regra.

3. **Manipulação recursiva de listas**

Considere uma lista ordenada com uma porção vazia no final



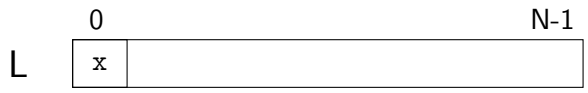
a) Faça um programa recursivo que insere um número `x` na lista, mantendo a lista ordenada.

b) Faça um programa recursivo que insere um número `x` na lista, mantendo a lista ordenada.

4. **Método de ordenação *Quicksort***

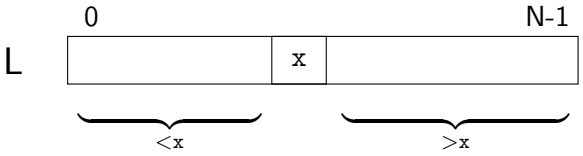
O método Quicksort funciona da seguinte maneira:

1. Dada uma lista desordenada



com um primeiro elemento `x` qualquer

2. Mova os elementos menores que `x` para o início da lista, e mova os elementos maiores que `x` para o fim da lista (deixando o `x` entre eles)



3. Finalmente, ordene os elementos menores que `x` em separado, e ordene os elementos maiores que `x` em separado.

A ideia, claro, é que a ordenação do passo 3 seja feita pelo mesmo método.

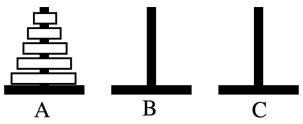
a) Escreva uma função `particiona`() que implementa a operação do passo 2.

b) Faça um programa recursivo que implementa o método de ordenação Quicksort.

c) Compare o tempo de execução da ordenação Quicksort com o tempo de execução da ordenação por intercalação.

5. **Torres de Hanoi**

A figura abaixo ilustra o jogo das torres de Hanoi



O problema consiste em mover os discos da haste A para a haste C, com o auxílio da haste B, sem nunca colocar um disco maior sobre um disco menor.

Por exemplo, a solução com três discos 1,2,3 fica assim

```
- mover disco 1 de A para C
- mover disco 2 de A para B
- mover disco 1 de C para B
- mover disco 3 de A para C
- mover disco 1 de B para A
- mover disco 2 de B para C
- mover disco 1 de A para C
```

No caso geral com n discos, a solução fica sempre assim

etapa 1: mover discos 1,. . . ,n-1 de A para B (com o auxílio de C)

etapa 2: mover disco n de A para C

etapa 3: mover discos 1,. . . ,n-1 de B para C (com o auxílio de A)

Como se pode ver, essa é uma solução recursiva.

Faça um programa que imprime a solução para o problema das torres de hanoi com n discos.