DCA0214.1 - LABORATÓRIO DE ESTRUTURAS DE DADOS
Aula 3: Busca, complexidade e ordenação em listas sequenciais (vetores)

Prof. Felipe Fernandes

09 Agosto de 2019

1. A sequência de Fibonacci pode ser definida recursivamente da seguinte forma: o primeiro termo é 0 e o segundo termo é 1. O n-ésimo termo é definido recursivamente com base na soma dos dois termos anteriores. Formalmente:

$$fibo(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ fibo(n-1) + fibo(n-2) & \text{case contrário} \end{cases} \quad (1)$$

   (a) Com base na definição acima, formule um algoritmo recursivo para encontrar o n-ésimo, $fibo(n)$, da sequência de Fibonacci. Qual a complexidade do seu algoritmo recursivo?

   (b) Escreva um algoritmo iterativo, **utilizando um vetor**, para encontrar o n-ésimo, $fibo(n)$, da sequência de Fibonacci. Qual a complexidade do seu algoritmo iterativo?

   (c) Escreva um algoritmo iterativo, **utilizando apenas três variáveis auxiliares**, para encontrar o n-ésimo, $fibo(n)$, da sequência de Fibonacci. Qual a complexidade do seu algoritmo iterativo?

   (d) Execute as três implementações (a),(b),(c) e verifique qual delas é mais eficiente.

2. Escreva uma função que recebe um número inteiro $n \geq 0$, um vetor de números inteiros distintos $v[0...n-1]$ e um número inteiro x e devolve k no intervalo $[0, n-1]$ tal que $v[k] == x$. Se tal k não existe, devolve -1. Faça isso para os itens abaixo.

   (a) Busca sequencial simples

   (b) Busca binária iterativa (assuma que o vetor já está ordenado)

(c) Busca binária recursiva (assuma que o vetor já está ordenado)

3. Refaça as funções de busca sequencial e busca binária assumindo que o vetor possui chaves que podem aparecer repetidas. Neste caso, a função deve retornar um outro vetor (*posicoes*) no qual constam todas as posições onde a chave foi encontrada. A função também deve retornar a quantidade de posições (*quant*) em que a chave procurada se repete.

```
void busca(int V[MAX1], int n, int x, int
posicoes[MAX1], int &quant)
```

4. Seja $V[0...n-1]$ um vetor com n valores numéricos. Faça o que se pede:

   (a) Escreva uma função iterativa que encontre o segundo maior elemento de V. Seu algoritmo deve ser baseado em comparações.

   (b) Qual a complexidade no melhor e no pior caso?

5. Uma forma de se obter a raiz quadrada de um número qualquer n seria simulando a busca binária. Assuma que a raiz quadrada de n está entre 0 e n (Se o número for negativo, retorne 0). Para sabermos se um palpite y é a raiz quadrada de n, basta testar se $y * y$ é próximo o suficiente de n ou seja, se o módulo da diferença entre eles está dentro de uma tolerância definida (*eps*). Caso contrário, podemos restringir a busca entre 0 e y ou entre y e n. Escreva a função abaixo que implemente este algoritmo.

```
double raiz_quadrada(double n, double eps) {
```

6. Dado um inteiro k, considere a sequência definida por:

$$f(n) = \begin{cases} n-1, 1 \leq n \leq k \\ f(n-1) + f(n-2), n > k \end{cases} \tag{2}$$

   Implemente um algoritmo $O(n)$ que calcule $f(n)$.