

AED1 - Lista 2

Vetores e busca

Seguem alguns exercícios relacionados com operações em vetores, busca sequencial e binária.

1 - [3.2.2] - O autor da função abaixo afirma que ela decide se x está no vetor $v[0..n-1]$. Critique o código.

```
int busc (int x, int n, int v[]) {
    if (v[n-1] == x) return 1;
    else return busc (x, n-1, v);
}
```

2 - [3.2.1] - Critique a seguinte função. Ela promete decidir se x está em $v[0 .. n - 1]$, devolvendo 1 em caso afirmativo e 0 em caso negativo. Qual a eficiência no melhor e no pior caso?

```
int muitofeio(int x, int v[], int n) {
    if (n == 0) return 0;
    else {
        int achei;
        achei = muitofeio(x, v, n-1);
        if (achei || x == v[n - 1]) return 1;
        else return 0; }
}
```

3 - [3.5.3] - Critique a seguinte função. Ela promete eliminar os zeros de $v[0..n-1]$, deixar o resultado em $v[0..m-1]$ e devolver m . Há alguma maneira simples de corrigir o código?

```
int tira0(int n, int v[]) {
    int i, z = 0;
    for (i = 0; i < n; i++) {
        if (v[i] == 0) z += 1;
        v[i - z] = v[i]; }
    return n - z;
}
```

4 - [7.2.2, 7.2.4] - Considere a seguinte função que faz uma busca sequencial em um vetor ordenado e devolve a posição em que x está ou deveria ser inserido.

Note que a convenção desta busca é diferente da vista em aula, pois quando não encontra x ela devolve a primeira (mais a esquerda) posição em que x poderia ser inserido de modo a manter o vetor ordenado, ao invés de devolver -1 .

```
int buscaSequencial (int x, int n, int v[]) {
    int j = 0;
    while (j < n && v[j] < x)
        ++j;
    return j;
}
```

a) Quais são os invariantes do processo iterativo na função `buscaSequencial`? Use os invariantes para mostrar que a função está correta.

b) Escreva uma versão recursiva da função `buscaSequencial`.

5 - [7.3.2, 7.4.3] - Considere a seguinte função que faz uma busca binária em um vetor ordenado e devolve a posição em que x está ou deveria ser inserido.

Note que a convenção desta busca é diferente da vista em aula, pois quando não encontra x ela devolve a primeira (mais a esquerda) posição em que x poderia ser inserido de modo a manter o vetor ordenado, ao invés de devolver -1 .

```
int buscaBinaria (int x, int n, int v[]) {
    int e, m, d;
    e = -1; d = n; // atenção!
    while (e < d-1) {
        m = (e + d)/2;
        if (v[m] < x) e = m;
        else d = m;
    }
    return d;
}
```

a) Suponha que $v[i] = i$ para todo i . Execute a função `buscaBinaria` com $n = 9$ e $x = 3$. Repita o exercício com $n = 14$ e $x = 7$. Repita o exercício com $n = 15$ e $x = 7$.

b) Na função `buscaBinaria`, mostre que temos $e < m < d$ imediatamente depois da atribuição " $m = (e + d) / 2$ ".

6 - [7.5.2] - Se preciso de t segundos para fazer uma busca binária em um vetor com n elementos, de quando tempo preciso para fazer uma busca em um vetor com n^2 elementos?

7 - [7.5.3] - Overflow aritmético. Se o número de elementos do vetor $v[0 .. n - 1]$ estiver próximo de INT_MAX , o código da busca binária pode descarrilar ao calcular a expressão $m = (e + d)/2$. Como evitar isso?

Para revisar conceitos sobre vetores e busca, além de encontrar mais exercícios, acesse:

- <https://www.ime.usp.br/~pf/algoritmos/aulas/array.html>
- <https://www.ime.usp.br/~pf/algoritmos/aulas/bubi.html>