24 Algoritmos de Busca

O objetivo desta aula é introduzir algoritmos que verificam se uma dada informação ocorre em uma sequência ou não. Estes algoritmos são conhecidos com **algoritmos de busca**.

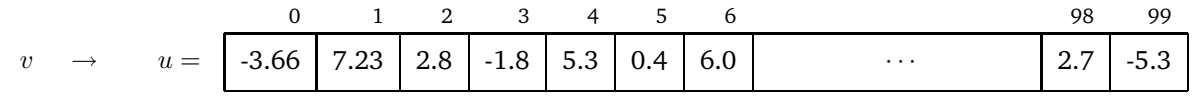
Ao final dessa aula você deverá saber:

- Definir o problema de busca de um elemento em um vetor.
- Descrever o que é busca sequencial.
- Descrever a vantagem de usar sentinela em busca sequencial.

24.1 Problema Básico

Dados um inteiro positivo n, uma sequência de n números reais guardados em vetor v e um número real x, faça uma função que responda à pergunta: x ocorre na sequência?

Exemplo: para $n = 5$, $x = 5.3$ e v apontando para o vetor u (por exemplo, da função `main`):



a função deve devolver 1, pois $v[4] = 5.3$. Agora, para $x = 2.7$ a função deve retornar 0. Por que se $v[98] = 2.7$? A resposta para esta pergunta é que o 2.7 em $v[98]$ é lixo pois somente as primeiras $n = 5$ casas de v (isto é, as casas cujos índices estão entre 0 e 4) contém a sequência.

Solução: usando o padrão de percorrimento de vetor, fazendo i variar de 0 a $n - 1$, verificar, elemento a elemento, se $v[i]$ é igual x. No caso de algum elemento $v[i]$ ser igual a x, então a função deve devolver 1 (verdadeiro). Caso contrário, a função deve devolver 0 (falso).

A solução acima é conhecida como **Busca Sequencial**.

```
int busca_sequencial (float v[], int n, float x) {
  int i;

  for (i=0; i<n; i++) {
    if (v[i] == x)
      return 1;
  }
  return 0;
}
```

24.2 Exercício

Faça um programa que leia um inteiro n, com $0 < n < 200$, e uma sequência de n números reais, e imprime a sequência eliminando os elementos repetidos.

Este exercício pode ser dividido em três partes:

(a) Escreva uma função com protótipo

```
int acha (float v[MAX], int n, float x);
```

que devolve a posição em que o real `x` ocorre no vetor `v` ou devolve -1 se `x` não aparece no vetor. O número de elementos do vetor é `n`.

Solução: a função pedida é uma variante da busca sequencial que, em vez de devolver 1 ou 0, devolve posição de `x` caso ele esteja em `v` ou o inteiro -1, caso contrário.

A solução pode ser então escrita como:

```
int acha (float v[MAX], int n, float x) {
  int i;

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

(b) Escreva uma função com protótipo

```
void insere (float v[MAX], int *n, float x);
```

que insere `x` na última posição do vetor `v` e altera o valor de `*n`.

Solução: simplesmente inserir `x` em `v[*n]` e fazer `*n=*n+1`.

```
void insere (float v[MAX], int *n, float x) {
  v[*n] = x;
  *n = *n + 1;
}
```

(c) Dada uma sequência de `n` números reais, com $0 <$ `n` < 200, imprimi-la eliminando as repetições.

Solução: considerar que você tem uma sacola vazia (um vetor `sacola` com `m=0` elementos). A cada elemento lido, verifica se ele está na sacola (usando a função `acha` do item (a)). Se ele já está na sacola, simplesmente faça o programa ler o próximo número; caso contrário, você o coloca na sacola (usando a função `insere` do item (b)). No final, o programa deve imprimir os números dentro da sacola.

```
# include <stdio.h>
# define MAX 200
int main () {
  int i, n, m=0, esta;
  float sacola[MAX], x;

  printf ("Entre com 0 < n < 200: ");
  scanf ("%d", &n);

  for (i=0; i<n; i++) {
    printf ("Entre com o elemento %d: ", i);
    scanf ("%f", &x);
    esta = acha (sacola, m, x);
    if (esta == -1) {
      insere (sacola, &m, x);
    }
  }

  for (i=0; i<m; i++) {
    printf ("%f\n", sacola[i])
  }

  return 0;
}
```