ALGORITMOS DE BUSCA

Na manipulação de dados, executa-se várias operações, uma das mais freqüentes e importantes é a de consulta a dados armazenados em estruturas como tabelas.

Um algoritmo é avaliado de acordo com a quantidade de memória que utiliza e o tempo de execução que gasta. Para otimizarmos o algoritmo deve-se escolher a melhor maneira de pesquisar os dados que o algoritmo vai manipular.

BUSCA SEQUENCIAL

Este tipo de busca consiste em verificar todos os elementos da lista, um por um, até o elemento procurado ser encontrado. Esse algoritmo é executado com complexidade O(n) portanto não se demonstra eficiente em sistemas com grandes quantidades de dados ou que necessitam efetuar buscas a todo momento (como nos bancos de dados presentes na maioria das empresas atuais).

A grande vantagem desse método de busca é que os dados que serão analisados não precisam passar por nenhum tipo de preparo (como a ordenação, por exemplo) antes da execução do algoritmo, tornando seu uso extremamente simples e confiável. Portanto, é muito comum a sua utilização nos sistemas atuais.

Exercícios:

1 Faça uma rotina recursiva que encontre um elemento em um vetor (não ordenado). A rotina deve ter como parâmetro o vetor, o número de elementos do vetor e o valor a ser procurado e, retornará o índice do elemento no vetor, caso este se encontre no vetor, -1 caso contrário.

Versão iterativa

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

Versão recursiva

```
int buscaR (int n, int *v, int num) {
    if (n == 0)
       return -1;
    if (v[n-1] == num)
       return n-1;
    return buscaR (n-1,v,num);
}
```

BUSCA BINÁRIA

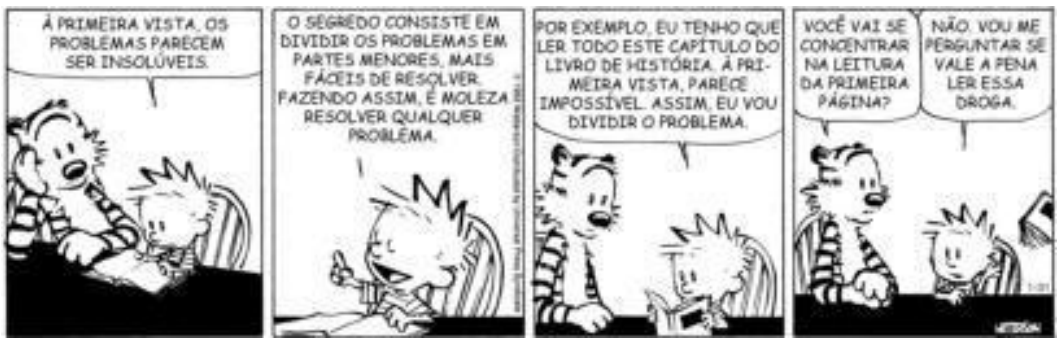
A pesquisa ou busca binária (em inglês *binary search algorithm* ou *binary chop*) é um algoritmo de busca em vetores que requer acesso aleatório aos elementos do mesmo. Ela parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca (divisão e conquista) comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Exercício:

Faça uma rotina recursiva que pesquise um elemento em um vetor ordenado utilizando busca binária. A rotina retornará o índice do elemento no vetor, caso este se encontre no vetor, -1 caso contrário.

```
int BuscaBin (int *v,int esq, int dir, int num) {
    int meio = (esq+dir)/2;
    if (esq > dir)
        return -1;
    if (v[meio] == num)
      return meio;
    if (num > v[meio])
       return BuscaBin (v,meio+1,dir,num);
    return BuscaBin (v,esq,meio-1,num);
}
```

O paradigma da busca binária serve de base para o conhecido "método da divisão e conquista", que pode ser usado para resolver eficientemente muitos problemas computacionais.



Exercícios

1 A seguinte função recursiva pretende encontrar o valor de um elemento máximo de um vetor (não ordenado) *v* de índice inicial *e* e índice final *d*, supondo e $\leqslant$ d.

```
float max (int e, int d, float *v) {
     int m;
     float maxe, maxd;
     if (e == d)
        return v[d];
     m = (e + d)/2;
     maxe = max (e, m, v);
     maxd = max (m+1, d, v);
     if (maxe >= maxd)
        return maxe;
     return maxd;
}
```

A função está correta? Ela é mais rápida que a versão iterativa? Quantas vezes a função chama a si mesma?

2 Escreva uma função eficiente que receba inteiros positivos k e n e calcule k^n. Quantas multiplicações sua função executa?