DevTitans

Busca

Busca em Vetor

Busca Linear

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

E se os elementos estiverem em um vetor ordenado, há alguma solução mais eficiente?

Busca Linear

```
int busca (int n, int* vet, int elem)
{
  int i;
  for (i=0; i<n; i++)
    if (elem == vet[i])
      return i;
    else if (vet[i] > elem)
      return -1;
  return -1;
}
```

É um pouco mais eficiente, mas no pior caso vai ter que percorrer todo o vetor

Busca Binária

```
int busca_bin (int n, int* vet, int elem)
{
  int ini = 0;
  int fim = n-1;
  int meio;
  while (ini <= fim) {
    meio = (ini + fim) / 2;
    if (elem < vet[meio])
      fim = meio - 1;
    else if (elem > vet[meio])
      ini = meio + 1;
    else
      return meio;
  }
  return -1;
}
```

Busca Binária Recursiva

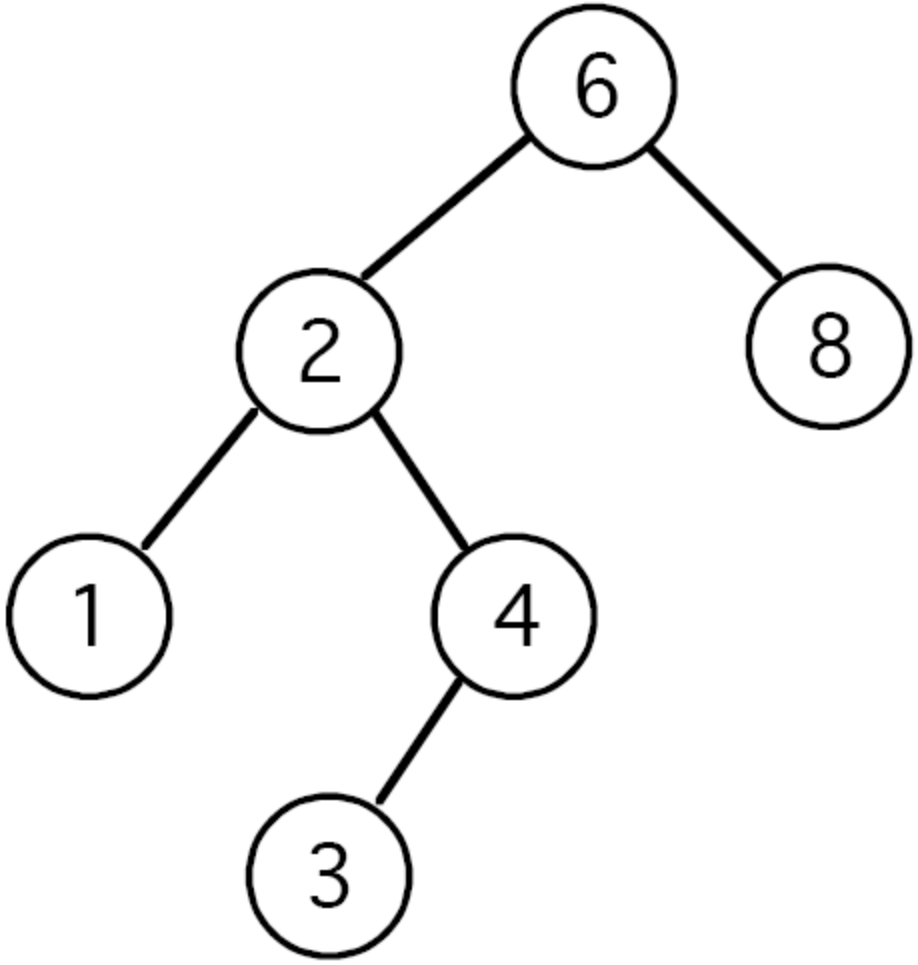
```
int busca_bin_rec (int n, int* vet, int elem)
{
  if (n <= 0) return 0;
  else {
    int meio = (n - 1) / 2;
    if (elem < vet[meio])
      return busca_bin_rec(meio,vet,elem);
    else if (elem > vet[meio])
      return busca_bin_rec(n-1-meio,&vet[meio+1],elem);
    else
      return 1;
  }
}
```

Árvore Binária de Busca

Introdução

- As árvores binárias que serão consideradas nesta seção têm uma propriedade fundamental:
  - *o valor associado à raiz é sempre maior que o valor associado a qualquer nó da sub-árvore à esquerda (sae), e é sempre menor que o valor associado a qualquer nó da sub-árvore à direita (sad)*
- Essa propriedade garante que, quando a árvore é percorrida em ordem infixada (*sae - raiz - sad*), os valores são encontrados em ordem crescente

Introdução



Tipo da Árvore Binária

```
struct arv {
   int info;
   struct arv* esq;
   struct arv* dir;
};

typedef struct arv Arv;
```

Operação de Busca

```
Arv* busca (Arv* r, int v)
{
  if (r == NULL) return NULL;
  else if (r->info > v) return busca(r->esq,v);
  else if (r->info < v) return busca(r->dir,v);
  else return r;
}
```

Operação de Inserção

```
Arv* insere (Arv* a, int v)
{
  if (a==NULL) {
    a = (Arv*)malloc(sizeof(Arv));
    a->info = v;
    a->esq = a->dir = NULL;
  }
  else if (v < a->info)
    a->esq = insere(a->esq,v);
  else
    a->dir = insere(a->dir,v);
  return a;
}
```