Árvore Binária de Pesquisa Balanceada

Inserção Binária

Implementação com arrays 1D

Introdução

A pesquisa de um elemento numa ABP implica

- Fazer uma travessia da árvore até se encontrar o elemento procurado, ou
- Concluir a sua inexistência

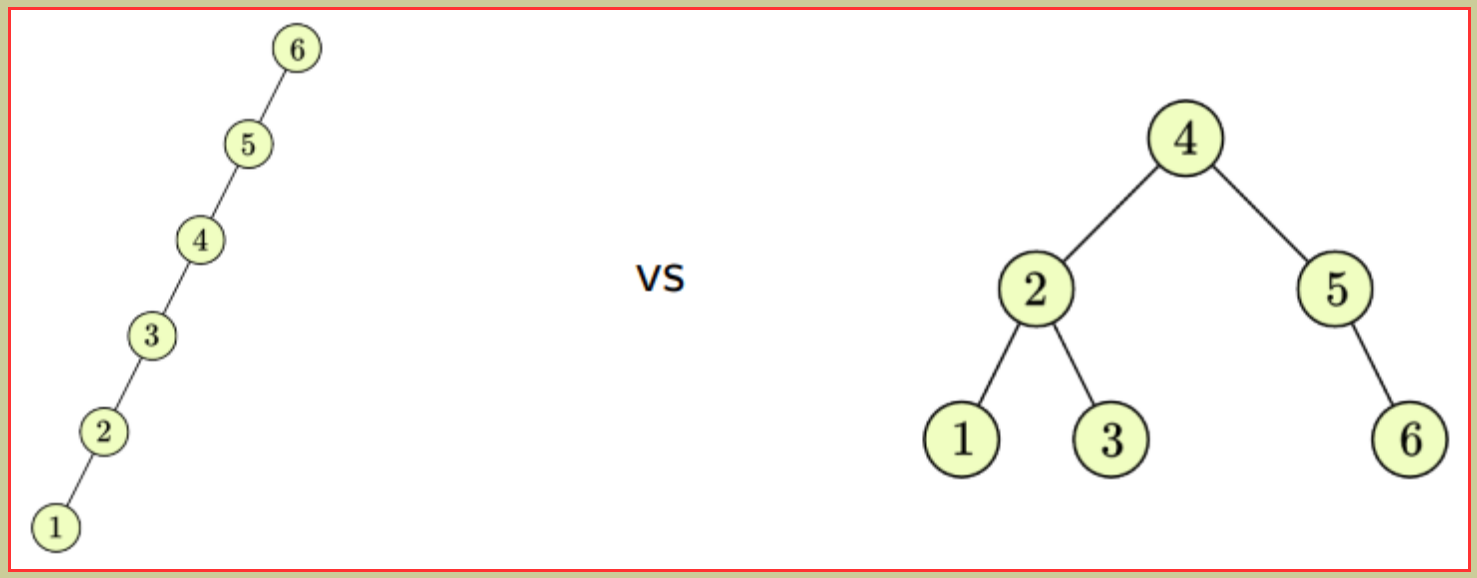
As travessias de uma árvore fazem-se em profundidade, pelo que

- Quanto *maior* é a *altura* de uma árvore *mais demorado* será o processo de pesquisa de um elemento, no pior caso e no caso médio

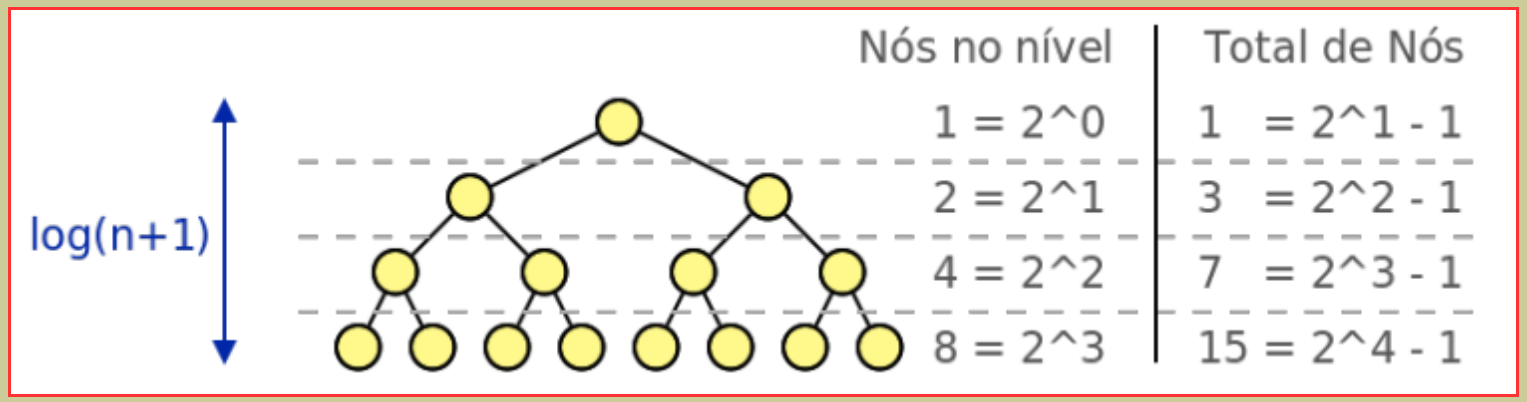
Assim sendo,

- É desejável que as árvores estejam balanceadas (equilibradas) em altura, para que os algoritmos de manipulação de árvores tenham eficiência logarítmica

Árvore não balanceada vs. árvore balanceada



Numa árvore equilibrada com n nodos, a altura calcula-se em O(log(n))



n = 15 (árvore cheia) ⇒ níveis = log(15+1) = 4 (2^4 = 16) ⇒ altura = 3

Pergunta 1

- Dado um conjunto de elementos, qual a **ordem de inserção** daqueles elementos numa árvore binária de pesquisa para que esta fique o mais balanceada possível?

Resposta 1

- Guardar os elementos numa lista (array 1D) e ordená-la por ordem crescente
- Aplicar a **inserção binária** da seguinte forma:
  - inserir o elemento do meio como raiz da ABP
  - dividir a lista restante em duas metades naquele elemento
    - metade da esquerda fica na subárvore esquerda da raiz
    - metada da direita fica na subárvore direita da raiz
  - inserir os restantes elementos de cada metada da mesma forma

Pergunta 2

- E se os elementos não forem conhecidos todos à partida?
- Ou seja, e se os elementos foram inseridos e removidos dinamicamente?

Resposta 2

- Existem várias estratégias com a mesma eficiência (complexidade), tais como:
  - Árvore AVL
  - Árvore Vermelha-Preto (Red-Black Tree)

Inserção Binária

Forma de equilibrar uma ABP em altura

- Garante um balanceamento perfeito
  - árvore com altura mínima (com o número mínimo de níveis)
- Mecanismo de balanceamento
  - 1º: armazenar os elementos da árvore numa sequência linear (lista) ordenada, usando a travessia em *em-ordem*
  - 2ª: construir uma nova árvore fazendo uma inserção "binária" dos elementos da sequência

Definição de árvore balanceada

- Considera-se uma ABP balanceada se a sua altura for o valor mínimo possível, atendendo à quantidade de nós da árvore (todos os níveis completos, com exceção do ultimo que pode não estar completo)
  - se uma ABP não vazia tem altura k (logo, k+1 níveis), então a quantidade de nós da ABP não pode ser inferior a 2^k: numNodos $\geq 2^{altura}$

Operação: criar ABP balanceada usando o método de inserção binária

Entrada: uma ABP T

Saída: a ABP T balanceada

```
PNodoABP  criarABPBalanceadaIB (PNodoABP  T){
  INFOABP *Lista;
  int  N = 0, Num;  // obrigatorio iniciar N a 0
  if (T == NULL)
     return T;
  if (arvoreBalanceada(T) == 1)  // árvore balanceada
     return T;
  Num = numeroNodosABP(T);
  Lista = (INFOABP *) malloc(Num * sizeof(INFOABP));
  if (Lista == NULL)
     return T;
  criarSequenciaEmOrdem(T, Lista, &N);
  balancearArvore(&T, Lista, 0, N-1);
  return T;
}
```

Operação auxiliar: verificar se uma ABP está balanceada

Entrada: uma ABP T
Saída: 1 (se T está balanceada) e 0 (se T não está balanceada)

```
int arvoreBalanceada (PNodoABP T)
{
  int  altura, numNodos;
  if (T == NULL)
      return 1;
  altura = alturaABP(T);
  numNodos = numeroNodosABP(T);
  if (numNodos >= pow(2, altura))   // numNodos >= 2^altura
      return 1;
  else
      return 0;
}
```

Operação auxiliar: criar a sequência de elementos (lista) de uma ABP

Entrada: uma ABP T

Saída: um vetor do tipo INFOABP e o número de elementos do vetor

```
void criarSequenciaEmOrdem (PNodoABP T, INFOABP *L, int *N)
{
  if (T != NULL){
     criarSequenciaEmOrdem(T→Esquerda, L, N);
     L[*N] = T→Elemento;
     *N = (*N) + 1;
     criarSequenciaEmOrdem(T→Direita, L, N);
  }
}
```

Operação auxiliar: construir uma ABP balanceada com elementos duma lista

Entrada: lista de elementos do tipo **INFOABP** e os índices **inicial** e **final** da lista
Saída: uma ABP balanceada

```
void balancearArvore (PNodoABP *T, INFOABP *L, int inicio, int fim)
{
  int  medio;
  if (inicio > fim)
      return ;
  if (inicio == fim){
      *T = inserirABP(*T, L[inicio]);
      return ;
  }
  medio = (inicio + fim) / 2;
  *T = inserirABP(*T, L[medio]);
  balancearArvore(T, L, inicio, medio-1);
  balancearArvore(T, L, medio+1, fim);
}
```