MC3305
Algoritmos e Estruturas de Dados II

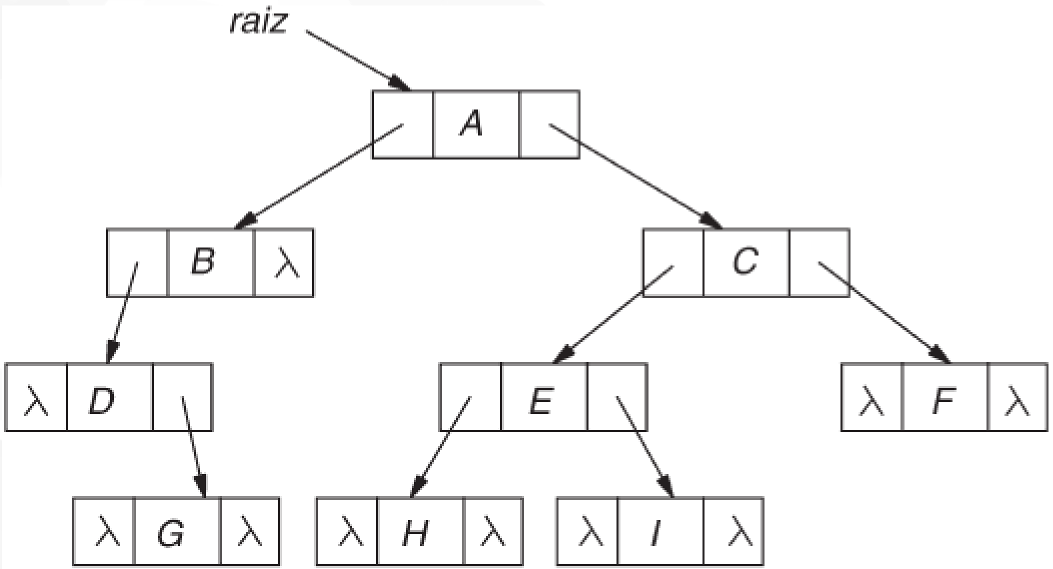
Aula 08 – Árvores Binárias de Busca

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

2Q-2015

Representação de uma árvore binária

λ =NULL



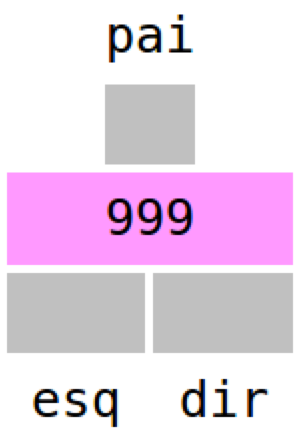
Para uma árvore binária de n vértices:
São requeridas 2n+1 unidades de memória para sua representação

n+1 unidades de memória são iguais a NULL.
Por que não aproveitar esse espaço de memoria?

Nós, filhos e pais

```
struct cel {
   int          conteudo; /
   struct cel *esq;
   struct cel *dir;
};
typedef struct cel no; /
```


```
struct cel {
   int          conteudo;
   struct cel *pai;
   struct cel *esq;
   struct cel *dir;
};
typedef struct cel no;
```



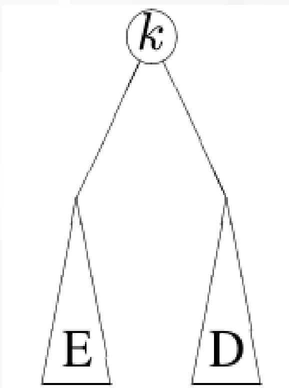
Nós, filos e pais

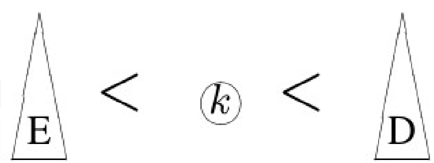
```c
void preenchePaiDadoFilho(no *pai, no *filho) {
    if (filho!=NULL) {
        filho->pai = pai;
        preenchePaiDadoFilho(filho, filho->esq);
        preenchePaiDadoFilho(filho, filho->dir);
    }
}

void preenchePai(no *r) {
    preenchePaiDadoFilho(r, r);
}
```

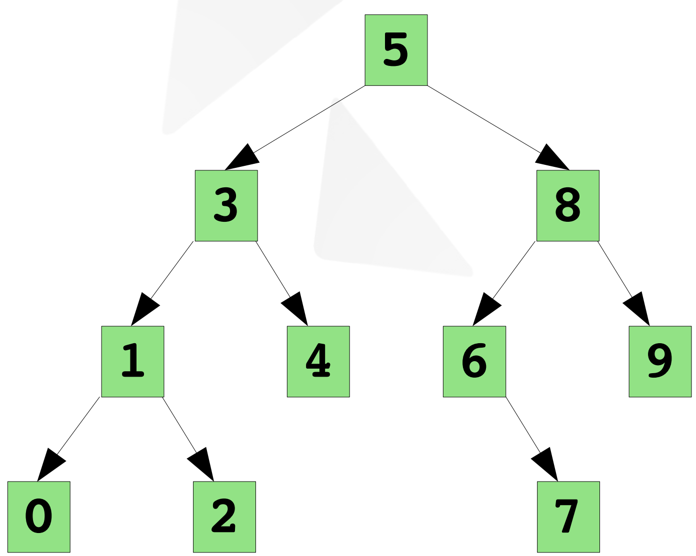
```c
// Bruna Gomes
void preenchePai(no *r) {
    if (r!=NULL) {
        if (r->esq!=NULL){
            r->esq->pai = r;
            preenchePai(r->esq);
        }
        if(r->dir != NULL){
            r->dir->pai = r;
            preenchePai(r->dir);
        }
    }
    if(r->pai == NULL){
        r->pai = r;
    }
}
```

```c
// Augusto Abreu
void preenchePai(no *r) {
    r->pai = r;
    preenche(r);
}

void preenche(no *r) {
    if (r->esq != NULL) {
        (r->esq)->pai = r;
        preenche(r->esq);
    }
    if (r->dir != NULL) {
        (r->dir)->pai = r;
        preenche(r->dir);
    }
}
```

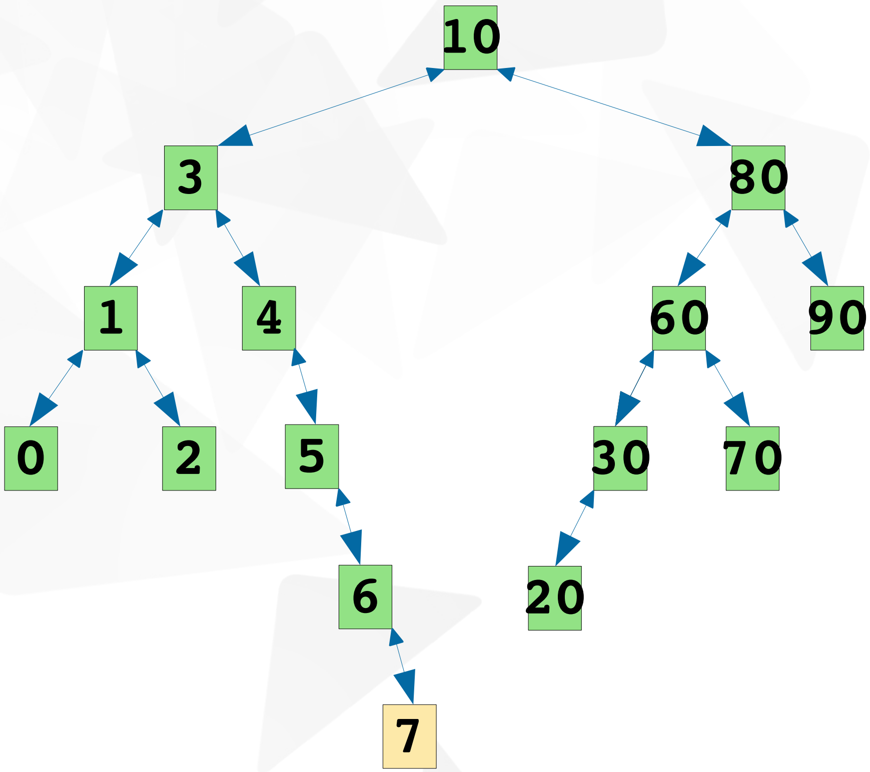
Árvores binárias de busca



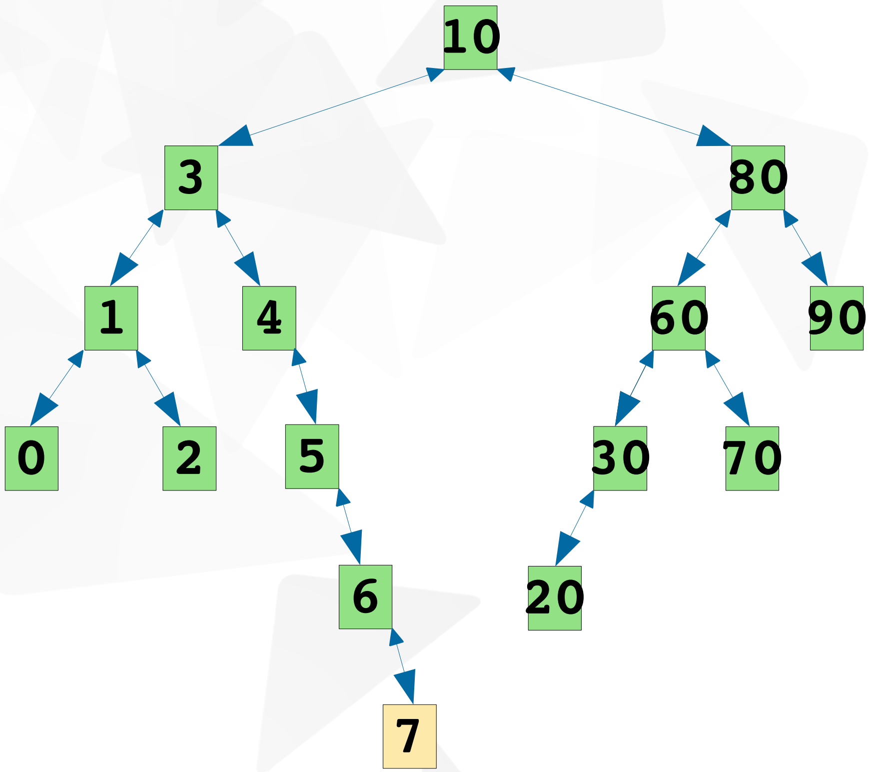




Em uma ABB
a ordem e-r-d das chaves
é crescente!



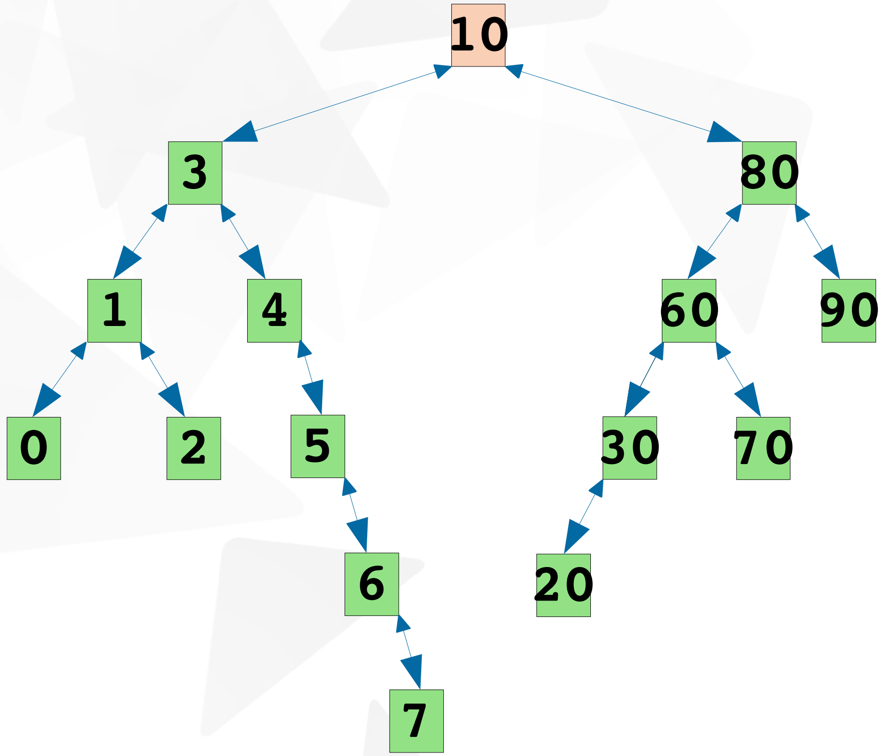

```
Folhas da arvore:
0        2        7        20       70       90
Altura da arvore:
5
```




```
no* primeiroErd(no *r) {
    while (r->esq!=NULL)
        r = r->esq;
    return r;
}
```

```
no* ultimoErd(no *r) {
    while (r->dir!=NULL)
        r = r->dir;
    return r;
}
```

Seguinte e anterior / sucessor e predecessor (repositório do Tidia)



Seguinte de 1 é 2.
Anterior de 1 é 0.

Seguinte de 7 é 10.
Anterior de 7 é 6.

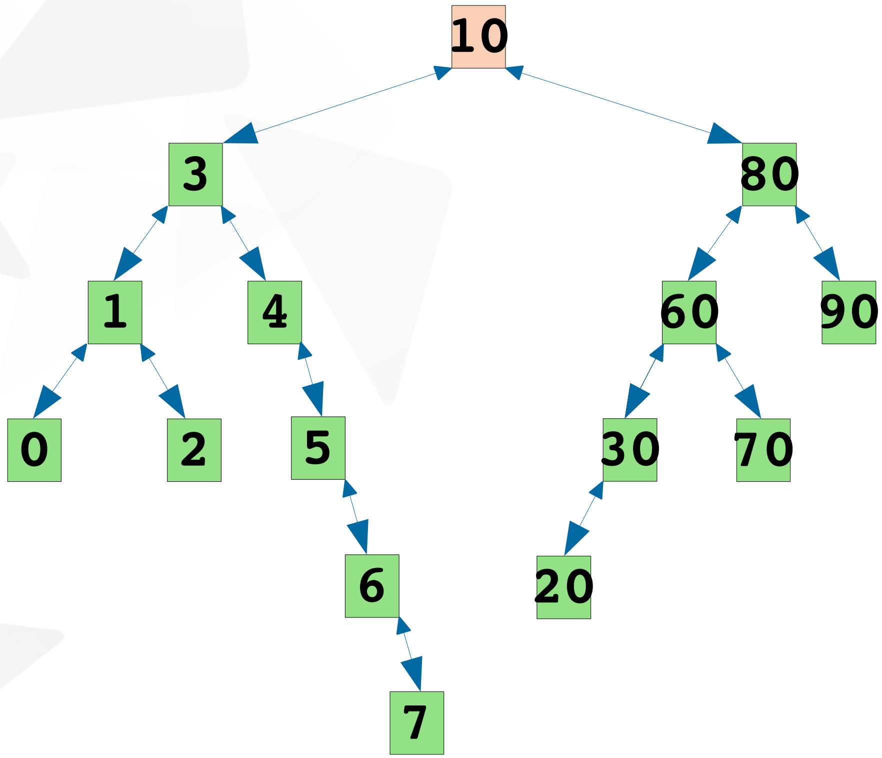
Seguinte de 20 é 30.
Anterior de 20 é 10.

Seguinte de 10 é 20.
Anterior de 10 é 7.

Escreva uma função que permita calcular o endereço do nó seguinte e do nó anterior na ordem e-r-d.

```
no* anterior (no* x)
no* seguinte (no* x)
```

Seguinte e anterior / sucessor e predecessor (repositório do Tidia)

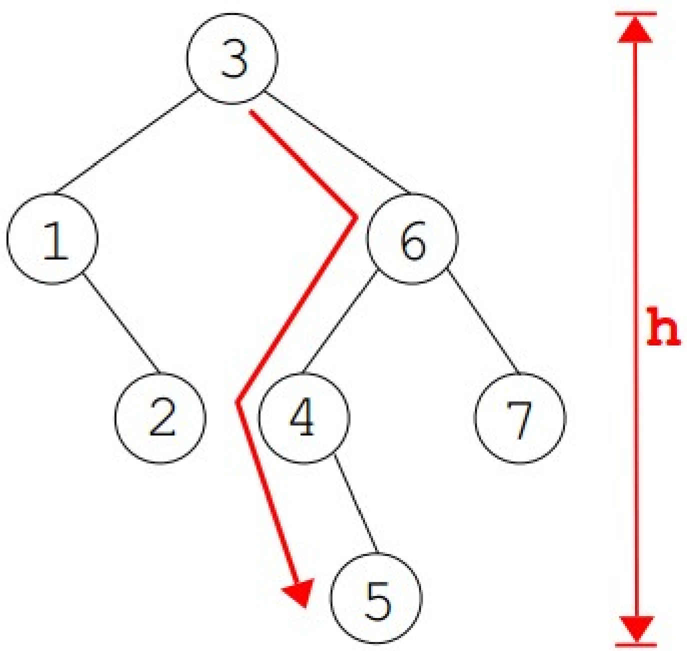



```
no* seguinte (no *x) {
    if (x->dir!=NULL) {
        no* y = x->dir;
        while(y->esq!=NULL)
            y = y->esq;
        return y;
    }
    while(x->pai!=x && x->pai->dir==x)
        x = x->pai;
    return x->pai;
}
```

```
no* anterior (no *x) {
    if (x->esq!=NULL) {
        no* y = x->esq;
        while(y->dir!=NULL)
            y = y->dir;
        return y;
    }
    while(x->pai!=x && x->pai->esq==x)
        x = x->pai;
    return x->pai;
}
```

Complexidade de busca em uma ABB

- Busca em ABB = **caminho da raiz até a chave desejada** (ou até a folha, caso a chave não exista)



Pior caso:

Maior caminho até a folha = altura da árvore

Complexidade: $O(h)$

Uma árvore binária balanceada é aquela com altura $O(\lg n)$

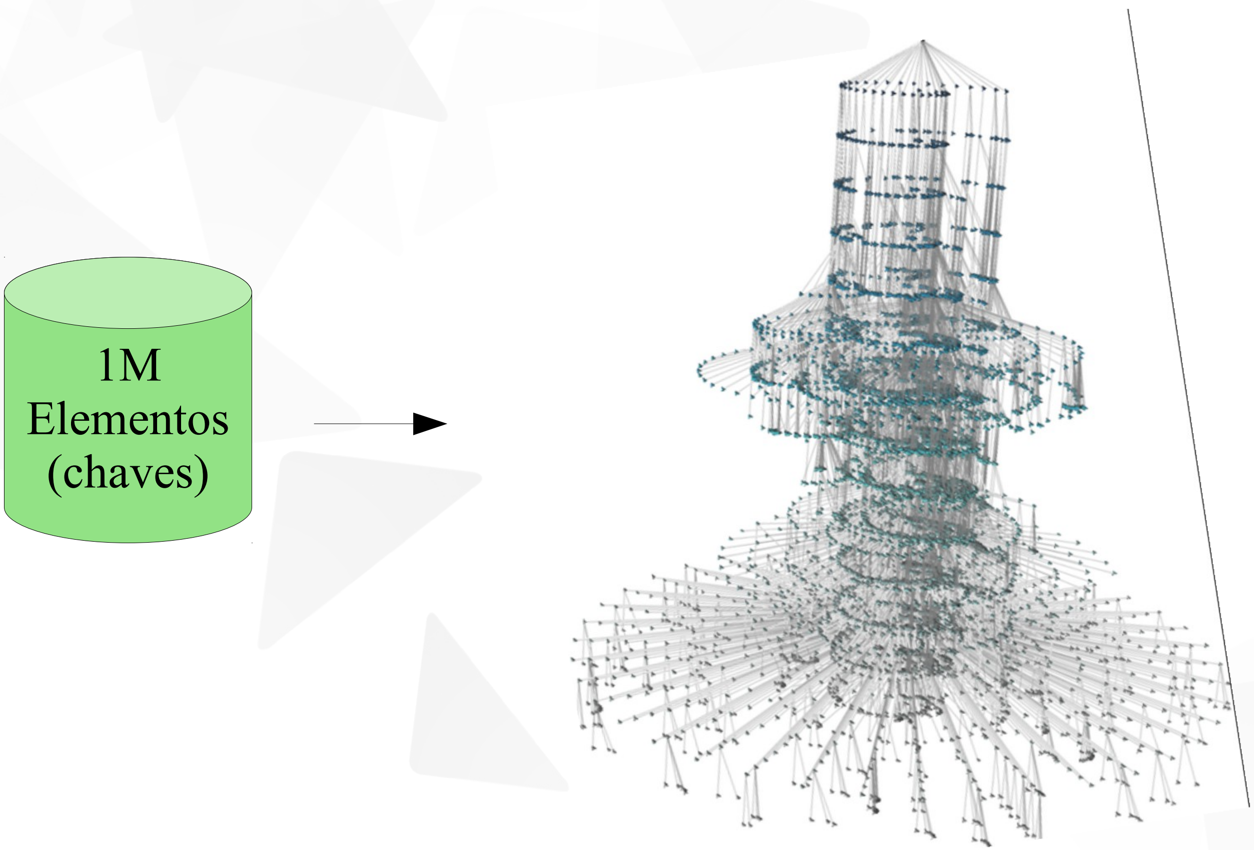
Complexidade de busca em uma ABB

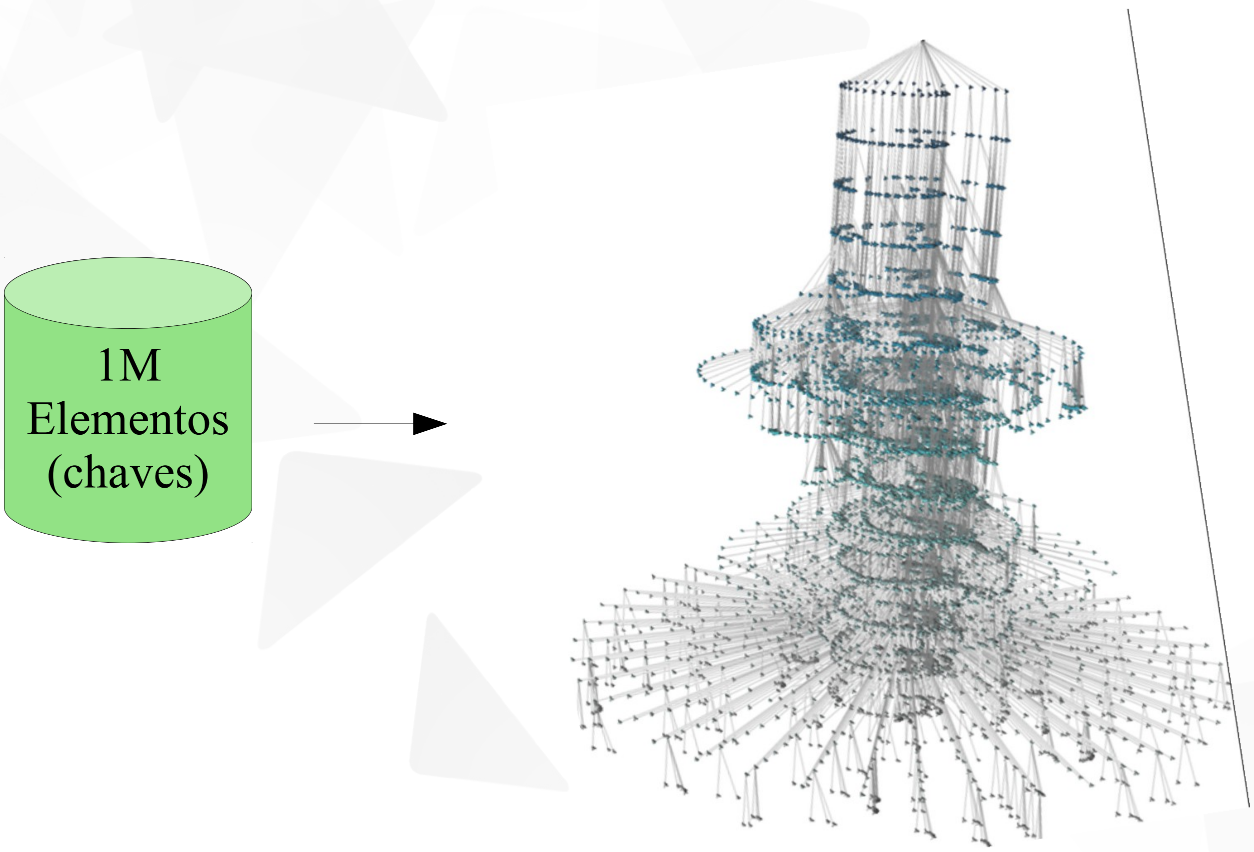
n	lg(n)
2	1
32	5
512	9
8192	13
131072	17
2097152	21
33554432	25
536870912	29
8589934592	33
137438953472	37
2199023255552	41
35184372088832	45
562949953421312	49
9007199254740990	53
144115188075856000	57
2305843009213690000	61
36893488147419100000	65
5.9029581035871E+020	69
9.4447329657393E+021	73
1.5111572745183E+023	77
2.4178516392293E+024	81
3.8685626227668E+025	85
6.1897001964269E+026	89
9.9035203142831E+027	93
1.5845632502853E+029	97
2.5353012004565E+030	101
4.0564819207303E+031	105
6.4903710731685E+032	109
1.0384593717070E+034	113
8.3076749736557E+034	116

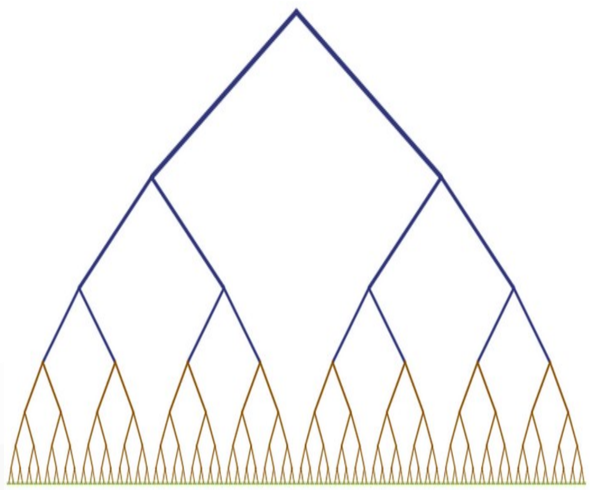
Uma árvore binária balanceada é aquela com altura $O(\lg n)$

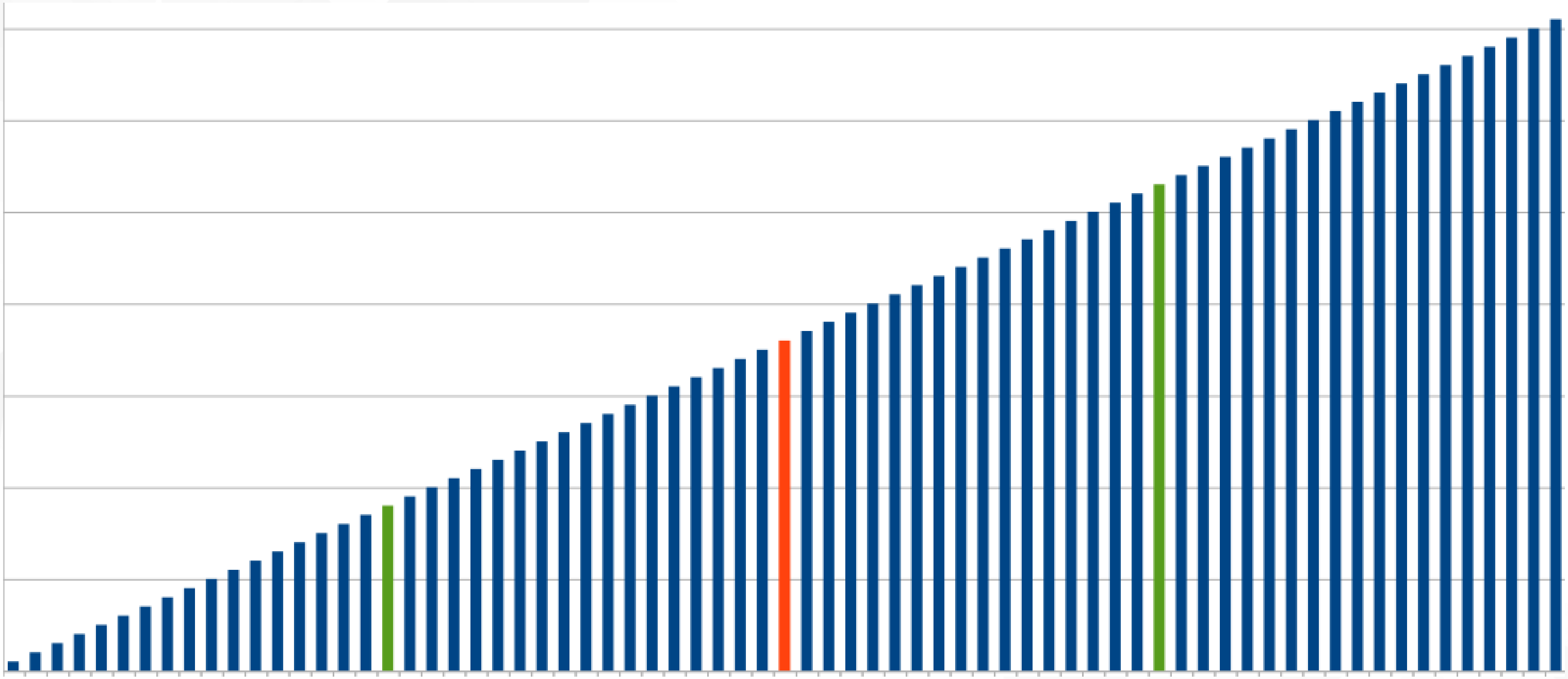
Uma árvore binária é balanceada (ou equilibrada) se, em cada um de seus nós, as subárvores esquerda e direita tiverem aproximadamente a mesma altura.

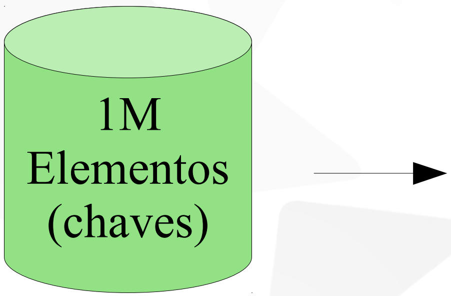
Construindo a melhor árvore binária de busca?

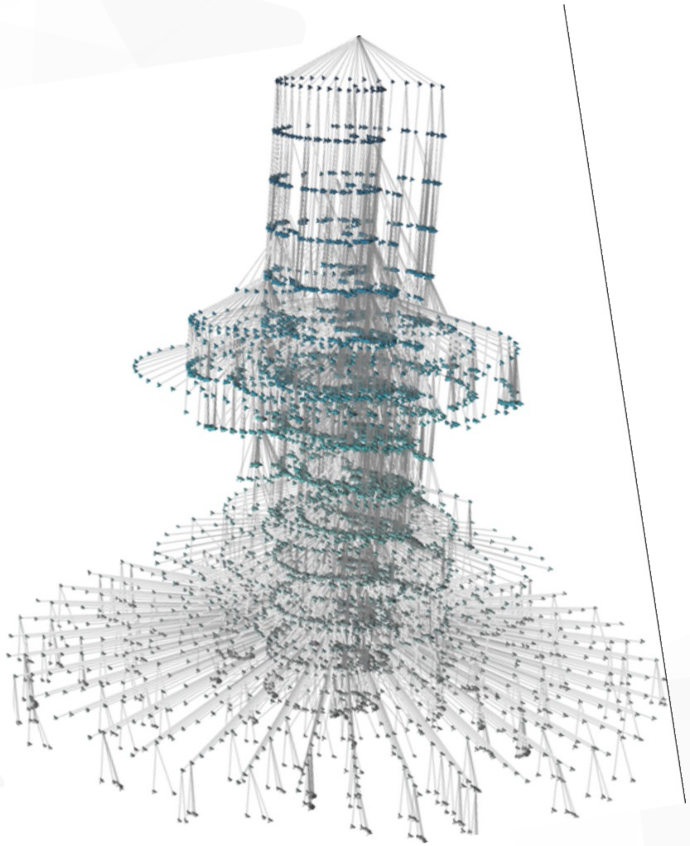

1M
Elementos
(chaves)


1M
Elementos
(chaves)









Construir a árvore: $O(n \lg(n))$

Árvore binária de busca ótima

Árvore binária de busca ótima

- Em diferentes casos, é conhecido com antecedência as chaves que serão buscadas (adicionalmente também pode se conhecer a frequência/probabilidade para cada chave).
- Uma estrutura que permita fazer uma busca de forma eficiente é uma **Árvore binária de busca ótima**.
- A melhor árvore depende das probabilidades associadas a cada chave:
  - Distribuição não uniforme nas chaves: a ABB não é balanceada.
  - As chaves mais buscadas deverão estar mais próximas ao topo da árvore.
- Não estamos preocupados na inserção nem na eliminação de elementos nesse tipo de ABB (versão *estática*).

Comprimento de caminho ponderado

- Denotamos por comprimento de caminho h_i de um nó k_i o número de nós encontrados desde a raiz até o nó k_i.
- Ele expressa o número de comparações realizadas para buscar uma chave no nó k_i.

Considere uma árvore binária de busca com n chaves

$$k_1 < k_2 < \ldots < k_n.$$

Suponha que se conhece a probabilidade de acesso de cada uma das chaves: sendo p_i a probabilidade de acesso à chave k_i, para $1 \leq i \leq n$:

$$\sum_{i=1}^{n} p_i = 1$$

Comprimento de caminho ponderado

O custo de busca P da árvore é expresso pelo comprimento de caminho ponderado da árvore, assim definida:

$$P = \sum_{i=1}^{n} p_i \, h_i$$

onde p_i é a probabilidade de acesso à chave k_i e h_i é o comprimento de caminho de k_i.

- Se toda chave tem igual probabilidade de ser buscada, então $p_i = 1/n$, $1 \leq i \leq n$ e teremos o comprimento de caminho médio da árvore, já visto anteriormente:

$$P = \sum_{i=1}^{n} p_i \, h_i = \sum_{i=1}^{n} \frac{h_i}{n} = \frac{1}{n} \sum_{i=1}^{n} h_i$$

Comprimento de caminho ponderado

Considere árvores binárias de busca com n chaves

$$k_1 < k_2 < \ldots < k_n.$$

Seja p_i a probabilidade de acesso à chave k_i, para $1 \leq i \leq n$.

Dentre todas tais árvores, é dita árvore binária de busca ótima aquela que minimiza o custo P:

$$P = \sum_{i=1}^{n} p_i \, h_i$$

onde h_i é o comprimento de caminho de k_i.

Árvore binária de busca ótima

Considere $n = 3$ e as 3 chaves

- $k_1 = 100$
- $k_2 = 200$
- $k_3 = 300$

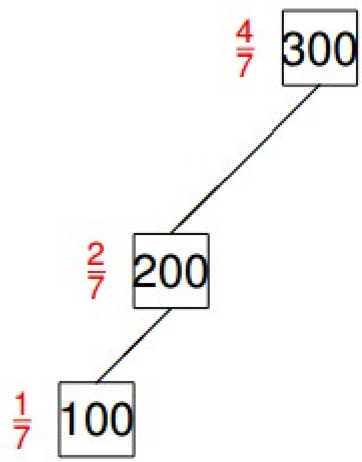
com as respectivas probabilidades de acesso

- $p_1 = \frac{1}{7}$
- $p_2 = \frac{2}{7}$
- $p_3 = \frac{4}{7}$.

Vamos ilustrar as possíveis árvores binárias de busca e seu respectivo custo P.

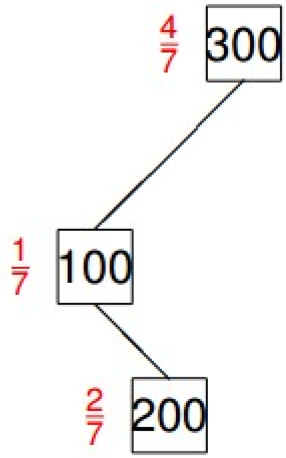
Árvore binária de busca ótima

Custo $P = 3 \times \frac{1}{7} + 2 \times \frac{2}{7} + 1 \times \frac{4}{7} = \frac{11}{7}$



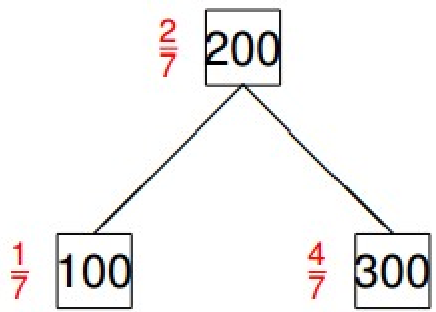
Árvore binária de busca ótima

Custo $P = 3 \times \frac{2}{7} + 2 \times \frac{1}{7} + 1 \times \frac{4}{7} = \frac{12}{7}$



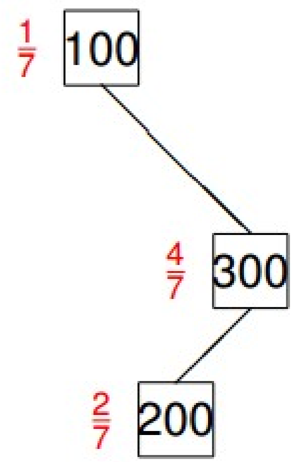
Árvore binária de busca ótima

Custo $P = 2 \times \frac{1}{7} + 2 \times \frac{4}{7} + 1 \times \frac{2}{7} = \frac{12}{7}$



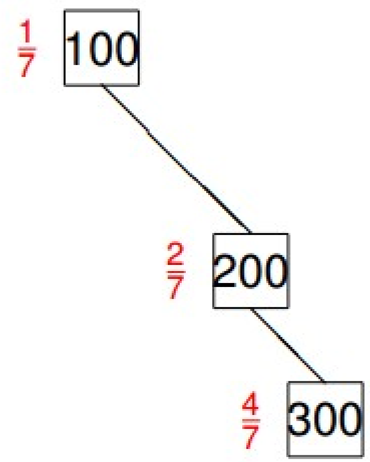
Árvore binária de busca ótima

Custo $P = 3 \times \frac{2}{7} + 2 \times \frac{4}{7} + 1 \times \frac{1}{7} = \frac{15}{7}$



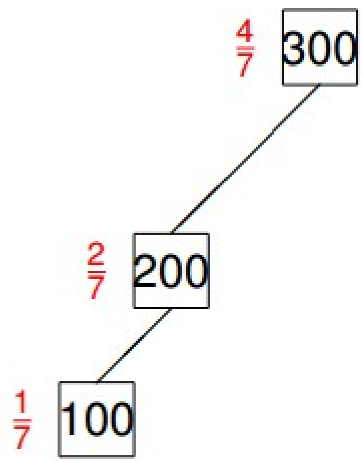
Árvore binária de busca ótima

Custo $P = 3 \times \frac{4}{7} + 2 \times \frac{2}{7} + 1 \times \frac{1}{7} = \frac{17}{7}$



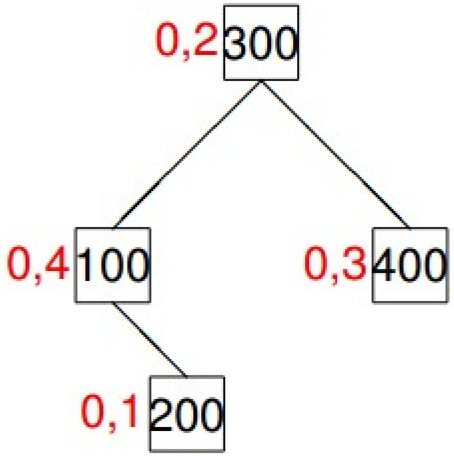
Árvore binária de busca ótima

Árvore 1 é a árvore ótima e não é balanceada.



A estratégia gulosa não funciona

- Pode parecer que basta adotar uma estratégia gulosa e começar colocando a chave de maior probabilidade de acesso na raiz.
- O exemplo mostra uma árvore ótima com as probabilidades de acesso em vermelho.
- A chave de maior probababilidade de acesso não está na raiz.



Lista 02

- Escreva um programa que **calcule a árvore binária de busca ótima**. A entrada de seu programa consiste de um número **n** de chaves a serem lidas, seguido de **n** linhas, contendo 2 inteiros: uma chave, e a sua respectiva frequência de acesso (busca).
- As chaves serão dadas em ordem crescente.
- A saida do seu programa deve conter o custo da árvore ótima bem como sua altura.

```
Entrada:
3
10 1
20 5
30 10


Saida:
Custo: 23
Altura: 3
```

Lista 02

- Material de estudo:
  - Aula do Prof. Siang: http://www.ime.usp.br/~song/mac5710/slides/07obst.pdf
  - Vídeo: https://www.youtube.com/watch?v=hgA4xxlVvfQ

→ Envio até **09/07** (23h50-Tidia)

→ Pode ser elaborado por até 2 alunos.

→ Apresentação livre de exemplos.

→ Veja no repositório do Tidia exemplos de entrada com suas respostas.