Árvores Binárias de Busca (ABB)
Notas de Aula

1 Vimos e relembramos que:

1. árvores geralmente usam alocação encadeada na implementação;
2. **árvores binárias:** para cada nó, há no máximo dois nós (esquerdo e direito);
3. **árvores binárias de busca:** se v é um nó com um filho esquerdo u e um filho direito w, então $u < v$ e $w > v$;
4. **operação de busca:** página 93, livro Szwarcfiter;
5. **operação de inserção:** página 95, livro Szwarcfiter;
6. **operação de remoção:** página 161, livro Ziviani;
7. **algoritmos para percorrer árvores**: in-ordem, pós-ordem, pré-ordem (páginas 76 e 77, livro Szwarcfiter), simétrico (conhecido como busca em largura);
8. **nível:** número de nós do caminho da raiz até o nó. O nível da raiz é, portanto, 1;
9. **altura:** número de nós do maior caminho da raiz até as folhas;
10. **árvore binária completa:** se v é um nó tal que alguma subárvore de v é vazia, então v se localiza ou no último ou no penúltimo nível da árvore (livro Szwarcfiter, pág. 69). Em outras palavras, todo nó sem alguma subárvore está no último ou penúltimo nível;

11. **árvore binária cheia:** se v é um nó com alguma de suas subárvores vazias, então v se localiza no último nível (livro Szwarcfiter, pág. 69), ou seja, toda folha está no último nível;

12. **árvore ziguezage:** árvore de altura máxima em que os nós possuem exatamente uma subárvore (página 69, livro Szwarcfiter);

13. **número de nós em cada nível da árvore:** 2^{i-1} onde i é o nível;

14. número de nós em uma árvore cheia de altura h: $n = 2^h - 1$;

15. **Provamos que:** toda árvore binária completa tem altura mínima e $h = 1 + \lfloor \log n \rfloor$ (página 71, livro Szwarcfiter).

2 Operações Básicas

Considere o nó da ABB como apresentado na Figura 1.

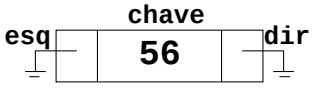

Figura 1: Representação de um nó da árvore binária de busca

2.1 Estrutura de Dados (em C/C++)

```
1 typedef struct No *Apontador;
2 typedef struct No {
3     int chave;
4     Apontador dir,esq;
5 } NoArvore;
```

2.2 Busca

```
1 int busca(int x, Apontador *pt){
2     if (*pt == NULL) {
3         return 0;
4     } else if (x == *pt → chave){
```

```
5           return 1;
6       } else if (x < *pt → chave){
7           if (*pt → esq == NULL){
8               return 2;
9           } else {
10              *pt = *pt → esq;
11              return (busca(x, *pt));
12          }
13      } else if (*pt → dir == NULL){
14          return 3;
15      } else {
16          *pt = *pt → dir;
17          return (busca(x, *pt));
18      }
19  }
```

Considere que a variável x é a chave a ser procurada e $ptraiz$ é o apontador para a raiz da árvore. A chamada inicial é: $pt = ptraiz$; $f = busca(x, \&pt)$. O valor de f indica o resultado da busca: $f = 0$ se árvore vazia; $f = 1$ se x está na árvore; $f > 1$ se x não está na árvore.

Fonte: página 93 do livro [1].

2.3 Inserção

```
1   void insere(Apontador novo, Apontador *ptraiz){ \\ insere se não está na árvore
2       Apontador pt = *ptraiz;
3       int f = busca(novo → chave,&pt);
4       if (f == 1) {
5           printf("Inserção Inválida");
6       } else if (f == 0) {
7           *ptraiz = novo;
8       } else if (f == 2) {
9           pt− > esq = novo;
10      } else {
11          pt− > dir = novo;
12      }
13  }
```

Fonte: página 95 do livro [1].

2.4 Remoção

```
1  antecessor(Apontador q, Apontador r){
2      if (r → dir! = NULL){
3          r → dir = antecessor(q, r → dir);
4      } else {
5          q = r; r = r → esq;
6      }
7      return r;
8  }
```

```
1  remove(int x, Apontador pt){
2      if (pt == NULL){
3          printf("Erro: item não encontrado");
4      } else if (x < pt → chave){
5          p → esq = remove(x, pt → esq);
6      } else if (x > pt → chave){
7          p → dir = remove(x, pt → dir);
8      } else {
9          if (pt → dir == NULL) {
10             pt = pt → esq;
11         } else if (pt → esq == NULL) {
12             pt = pt → dir;
13         } else {
14             pt → esq = antecessor(pt, pt → esq);
15         }
16     }
17     return pt;
18 }
```

Fonte: página 179 do livro [2].

3 Bibliografia Utilizada

Referências

[1] SZWARCFITER, J. L. e MARKENZON, L. *Estruturas de Dados e seus Algoritmos*, 2^a edição, LTC Editora, 1994.

[2] ZIVIANI, N. *Projeto de Algoritmos: com implementações em Pascal e C*, 2^a edição, Thomson Learning, 2009.