Estrutura de Dados

Árvore binária – Parte 2

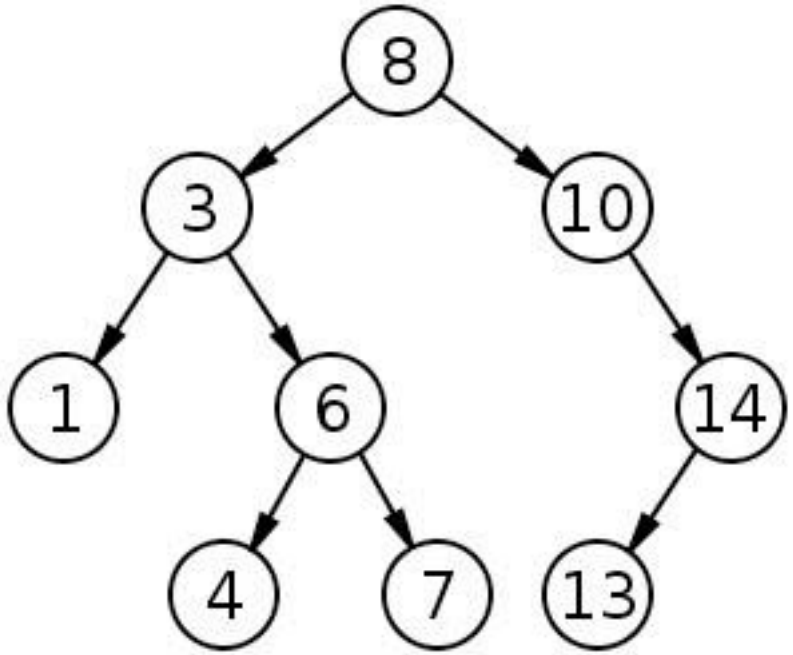
Prof. Dr. Daniel Vecchiato

Agenda

- Relembrando
- Remoção de nós
- Exercícios

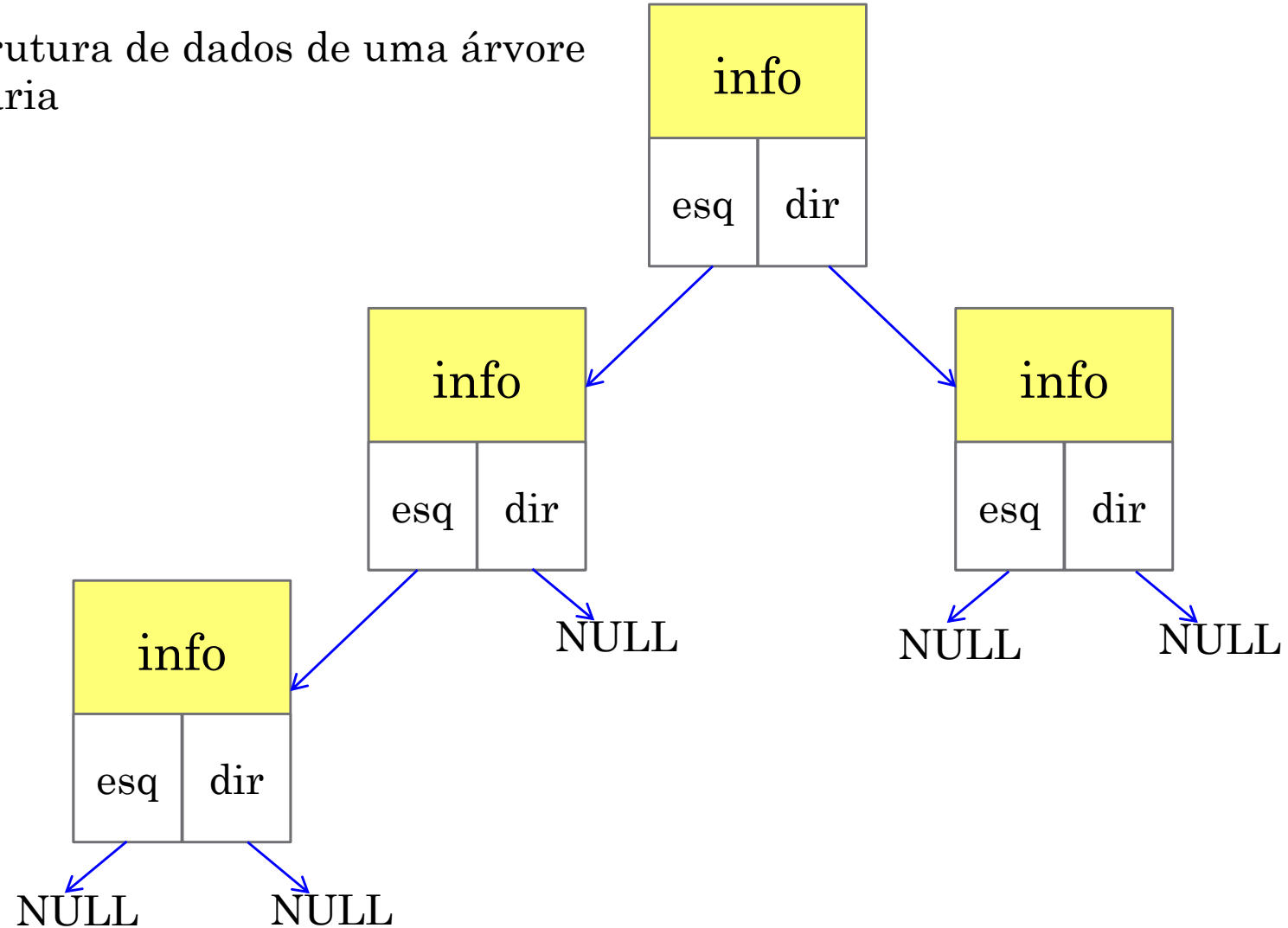
Relembrando

- Estrutura de dados de uma árvore binária



Relembrando

- Estrutura de dados de uma árvore binária



Relembrando

- Definição
  - Tipo de árvore que possui chave e 2 ponteiros para subárvores
    - além de outras informações associadas a respectiva chave
  - Em um nó com chave X:
    - As chaves na subárvore da **esquerda são menores** que X
    - As chaves na subárvore do **direita são maiores** que X

Relembrando

- Implementação básica

```
typedef struct {
	int chave;
	int informacao;
} TItem;


typedef struct No {
	TItem item;
	struct No *pEsq, *pDir;
} TNo;
```

Relembrando

- Implementação básica

```
TNo* criarNo (TItem x) {
        TNo *pAux = (TNo*) malloc (sizeof(TNo));
        pAux->item = x;
        pAux->pEsq = NULL;
        pAux->pDir = NULL;
        return pAux;
}

TNo* inserirNo (TNo *pR, TItem x) {
        if (pR == NULL)
                pR = criarNo (x);
        else
                if (x.chave < pR->item.chave)
                        pR->pEsq = inserirNo (pR->pEsq, x);
                else
                        pR->pDir = inserirNo (pR->pDir, x);
        return pR;
}
```

Relembrando

- Implementação básica

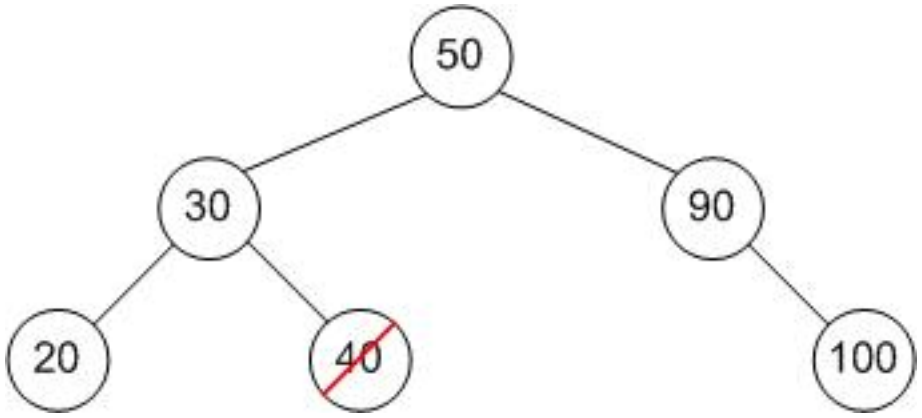
```
TItem* pesquisa (TNo* pR, int chave) {
	if (pR == NULL)
		return NULL;

	if (chave < pR->item.chave)
		return pesquisa (pR->pEsq, chave);

	if (chave > pR->item.chave)
		return pesquisa (pR->pDir, chave);

	return &(pR->item);
}
```

Remoção de nós

- O processo de remoção de nós não é tão simples quanto a inserção
- Deve ser considerado três casos diferentes:
  - Remoção de um nó folha (sem filhos)
  - Remoção de um nó com um único filho
  - Remoção de um nó com os dois filhos

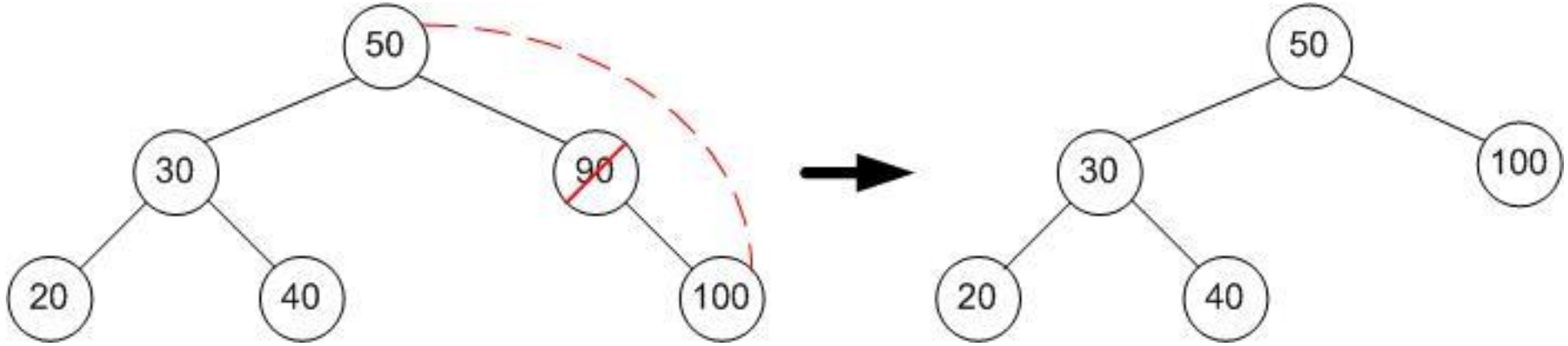
Remoção de nós

- Nó folha
  - Caso mais simples
  - Pode ser removido e atribuído NULL em seu lugar



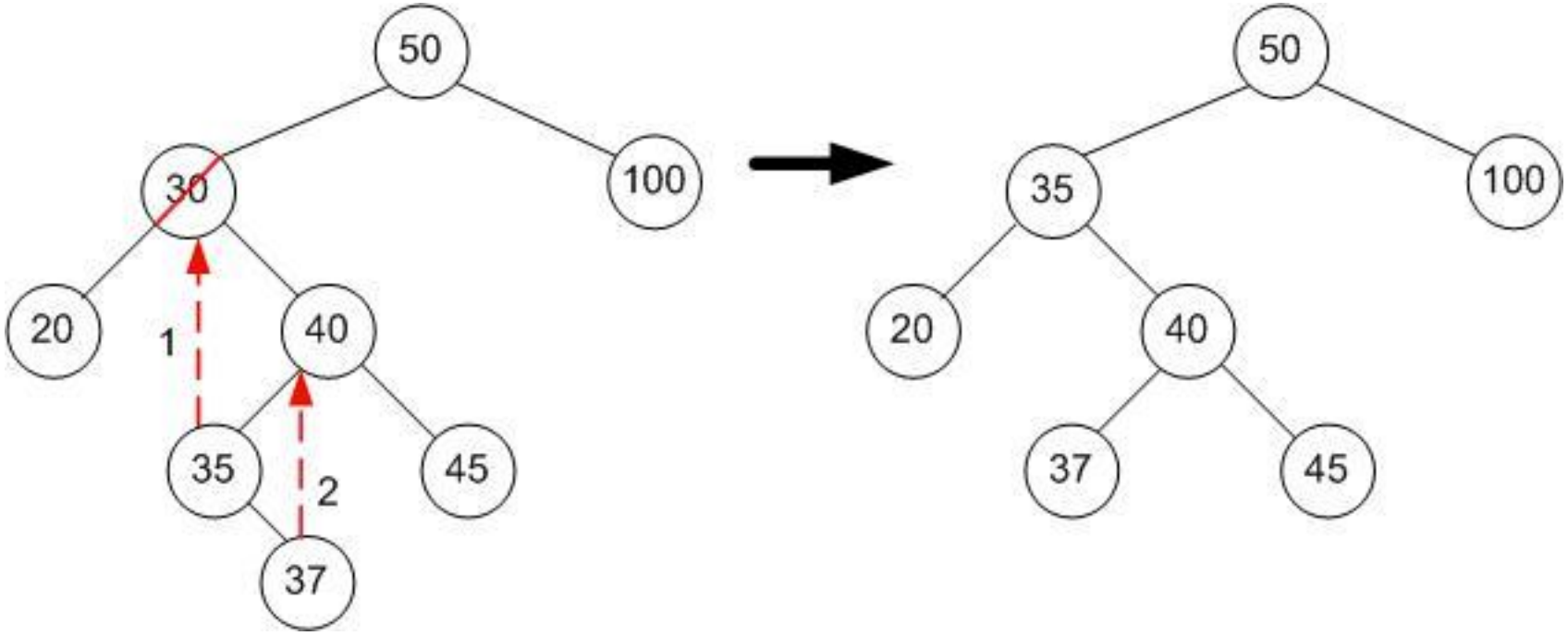
Remoção de nós

- Nó com 1 filho
  - O filho do nó excluído passa a ocupar a posição do pai



Remoção de nós

- Nó com 2 filhos
  - Deve substituir o valor do nó a ser retirado pelo valor sucessor
    - Sucessor: nó mais a esquerda da sub-árvore direita
  - Depois, remove-se o nó sucessor



Código para remoção de nós

```
TNo* removerNo (TNo *pR, int chave) {
    if (pR == NULL)
        return pR;

    if (chave < pR->item.chave)
        pR->pEsq = removerNo (pR->pEsq, chave);
    else if (chave > pR->item.chave)
        pR->pDir = removerNo (pR->pDir, chave);
    else {
        // Mesmo valor, deve ser removido este nó
         ...
    }

    return pR;
}
```

Código para remoção de nós

```
TNo* removerNo (TNo *pR, int chave) {
    ...
    else {
        // Mesmo valor, deve ser removido este nó
        // Nó com apenas um filho ou nenhum filho
        if (pR->pEsq == NULL) {
            TNo *aux = pR->pDir;
            free (pR);
            return aux;
        } else if (pR->pDir == NULL) {
            TNo *aux = pR->pEsq;
            free (pR);
            return aux;
        }
        // Nó com 2 filhos
        ...
```

Código para remoção de nós

```
TNo* removerNo (TNo *pR, int chave) {
    ...
        // Nó com 2 filhos
        TNo *aux = pR->pDir;
        while (aux->pEsq != NULL)
            aux = aux->pEsq;
        pR->item = aux->item;
        pR->pDir = removerNo (pR->pDir, aux->item.chave);
    }
    ...
}
```

Remoção de nós

```
TNo* removerNo (TNo *pR, int chave) {
    if (pR == NULL)
        return pR;

    if (chave < pR->item.chave)
        pR->pEsq = removerNo (pR->pEsq, chave);
    else if (chave > pR->item.chave)
        pR->pDir = removerNo (pR->pDir, chave);
    else {
        if (pR->pEsq == NULL) {
            TNo *aux = pR->pDir;
            free (pR);
            return aux;
        } else if (pR->pDir == NULL) {
            TNo *aux = pR->pEsq;
            free (pR);
            return aux;
        }

        TNo *aux = pR->pDir;
        while (aux->pEsq != NULL)
            aux = aux->pEsq;
        pR->item = aux->item;
        pR->pDir = removerNo (pR->pDir, aux->item.chave);
    }

    return pR;
}
```

Exercícios

- Implemente a função de remoção de um nó
- Faça funções para
  - calcular a altura da árvore
  - calcular quantos nós existem na árvore
  - calcular quantos nós folhas existem na árvore
  - verificar se duas árvores são iguais

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)
- Demaine, E., Devadas, S. Introduction to Algorithms (MIT OpenCourseWare), http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011
- http://www.ft.unicamp.br/liag/siteEd/