Estruturas de Informação
Aula 13: Árvores

08/10/2008

Fontes Bibliográficas

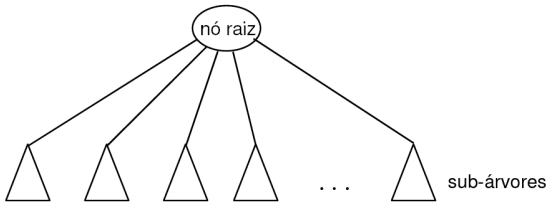
- Livros:
  - Introdução a Estruturas de Dados (Celes, Cerqueira e Rangel): Capítulo 13;
  - Projeto de Algoritmos (Nivio Ziviani): Capítulo 5;
  - Estruturas de Dados e seus Algoritmos (Szwarefiter, et. al): Capítulo 3;
  - Algorithms in C (Sedgewick): Capítulo 5;
- Slides baseados no material da PUC-Rio, disponível em http://www.inf.puc-rio.br/~inf1620/.

Introdução

- Estruturas estudadas até agora não são adequadas para representar dados que devem ser dispostos de maneira hierárquicas
  - Ex., hierarquia de pastas
  - Árvore genealógica
- Árvores são estruturas adequadas para representação de hierarquias

Definição Recursiva de Árvore

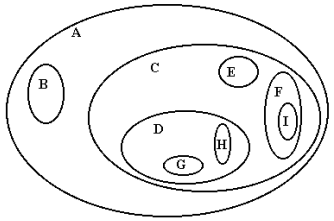
- Um conjunto de nós tal que:
  - existe um nó r, denominado *raiz,* com zero ou mais sub-árvores, cujas raízes estão ligadas a r
  - os nós raízes destas sub-árvores são os *filhos* de r
  - os *nós internos* da árvore são os nós com filhos
  - *as folhas* ou *nós externos* da árvore são os nós sem filhos



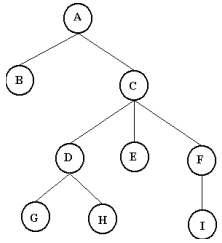
Formas de representação

- Representação por parênteses aninhados
  - (A (B) (C (D (G) (H)) (E) (F (I))))

Diagrama de Inclusão



Representação Hierárquica

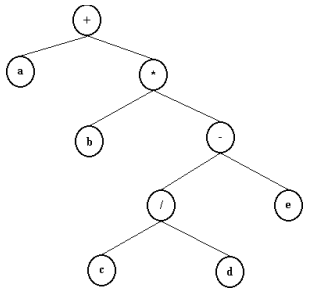


Subárvore

- Seja a árvore acima T = {A, B, ...}
- A árvore T possui duas subárvores:
  - Tb e Tc onde Tb = { B } e Tc = {C, D, ...}
- A subárvore Tc possui 3 subárvores:
  - Td, Tf e Te onde Td = {D, G, H}, Tf = {F, I}, Te = {E}
- As subárvores Tb, Te, Tg, Th, Ti possuem apenas o nó raiz e nenhuma subárvore.

Exemplo (árvore de expressão)

- Representação da expressão aritmética:
  (a + (b * (c / d - e)))



Conceitos Básicos

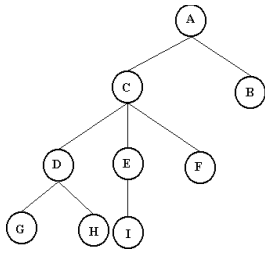
- Nós filhos, pais, tios, irmãos e avô
- Grau de saída (número de filhos de um nó)
- Nó folha (grau de saída nulo) e nó interior (grau de saída diferente de nulo)
- Grau de uma árvore (máximo grau de saída)
- Floresta (conjunto de zero ou mais árvores)

Conceitos Básicos (2)

- Caminho
  - Uma sequência de nós distintos v1, v2, ..., vk, tal que existe sempre entre nós consecutivos (isto é, entre v1 e v2, entre v2 e v3, ... , v(k-1) e vk) a relação "é filho de" ou "é pai de" é denominada um caminho na árvore.
- Comprimento do Caminho
  - Um caminho de vk vértices é obtido pela sequência de k-1 pares. O valor k-1 é o comprimento do caminho.
- Nível ou profundidade de um nó
  - número de nós do caminho da raiz até o nó.

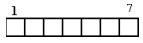
Conceitos Básicos (3)

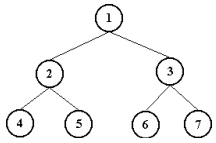
- Nível da raiz (profundidade) é 0.
- Árvore Ordenada: é aquela na qual filhos de cada nó estão ordenados. Assume-se ordenação da esquerda para a direita. Esta árvore é ordenada?



Conceitos Básicos (4)

- Árvore Cheia: Uma árvore de grau d é uma árvore cheia se possui o número máximo de nós, isto é, todos os nós têm número máximo de filhos exceto as folhas, e todas as folhas estão na mesma altura.
- Árvore cheia de grau 2: implementação sequencial.



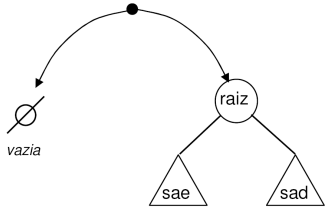


Armazenamento por nível:

posição do nó	**posição dos filhos do nó**
1	2,3
2	4,5
3	6,7
i	(2i,2i+1)

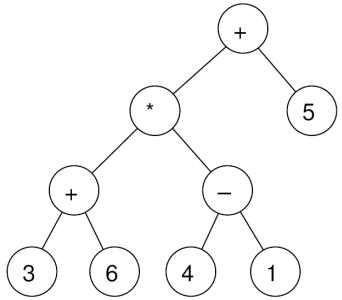
Árvore Binária

- Uma árvore em que cada nó tem zero, um ou dois filhos
- Uma árvore binária é:
  - uma árvore vazia; ou
  - um nó raiz com duas sub-árvores:
    - a subárvore da direita (sad)
    - a subárvore da esquerda (sae)



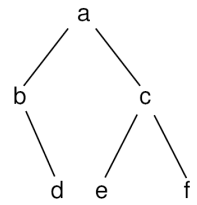
Exemplo

- Árvore binária representando expressões aritméticas binárias
  - Nós folhas representam os operandos
  - Nós internos representam os operadores
  - (3+6)*(4-1)+5



Árvores Binárias

- Notação textual
  - a árvore vazia é representada por <>
  - árvores não vazias por <raiz sae sad>
- Exemplo:
  - <a <b <> <d<><>> > <c <e<><>> <f<><>>> >



Árvores Binárias – Implementação em C

- Representação: ponteiro para o nó raiz
- Representação de um nó na árvore:
  - Estrutura em C contendo
    - A informação propriamente dita (exemplo: um caractere, ou inteiro)
    - Dois ponteiros para as sub-árvores, à esquerda e à direita

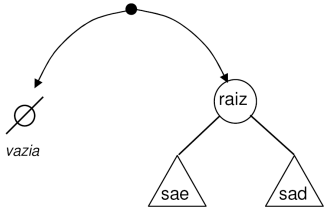
```
struct arv {
 char info;
 struct arv* esq;
 struct arv* dir;
};
```

TAD Árvores Binárias – Impl. em C (arv.h)


```
typedef struct arv Arv;
//Cria uma árvore vazia
Arv* arv_criavazia (void);
//cria uma árvore com a informação do nó raiz c, e
//com subárvore esquerda e e subárvore direita d
Arv* arv_cria (char c, Arv* e, Arv* d);
//libera o espaço de memória ocupado pela árvore a
Arv* arv_libera (Arv* a);
//retorna true se a árvore estiver vazia e false
//caso contrário
int arv_vazia (Arv* a);
//indica a ocorrência (1) ou não (0) do caracter c
int arv_pertence (Arv* a, char c);
//imprime as informações dos nós da árvore
void arv_imprime (Arv* a);
```

TAD Árvores Binárias – Implementação em C

- Implementação das funções:
  - implementação em geral recursiva
  - usa a definição recursiva da estrutura
- Uma árvore binária é:
  - uma árvore vazia; ou
  - um nó raiz com duas sub-árvores:
    - a sub-árvore da direita (sad)
    - a sub-árvore da esquerda (sae)



TAD Árvores Binárias – Implementação em C

- função arv_criavazia
  - cria uma árvore vazia

```
Arv* arv_criavazia (void){
  return NULL;
}
```

TAD Árvores Binárias – Implementação em C

- função arv_cria
  - cria um nó raiz dadas a informação e as duas sub-árvores, a da esquerda e a da direita
  - retorna o endereço do nó raiz criado

```
Arv* arv_cria (char c, Arv* sae, Arv* sad){
  Arv* p=(Arv*)malloc(sizeof(Arv));
  p->info = c;
  p->esq = sae;
  p->dir = sad;
  return p;
}
```

TAD Árvores Binárias – Implementação em C

- arv_criavazia e arv_cria
  - as duas funções para a criação de árvores representam os dois casos da definição recursiva de árvore binária:
    - uma árvore binária Arv* a;
      - é vazia a=arv_criavazia()
      - é composta por uma raiz e duas sub-árvores a=arv_cria(c,sae,sad);

TAD Árvores Binárias – Implementação em C

- função arv_libera
  - libera memória alocada pela estrutura da árvore
    - as sub-árvores devem ser liberadas antes de se liberar o nó raiz
  - retorna uma árvore vazia, representada por NULL

```
Arv* arv_libera (Arv* a){
  if (!arv_vazia(a)){
    arv_libera(a->esq); /* libera sae */
    arv_libera(a->dir); /* libera sad */
    free(a); /* libera raiz */
  }
  return NULL;
}
```

TAD Árvores Binárias – Implementação em C

- função arv_vazia
  - indica se uma árvore é ou não vazia

```
int arv_vazia (Arv* a){
  return a==NULL;
}
```

TAD Árvores Binárias – Implementação em C

- função arv_pertence
  - verifica a ocorrência de um caractere c em um dos nós
  - retorna um valor booleano (1 ou 0) indicando a ocorrência ou não do caractere na árvore

```
int arv_pertence (Arv* a, char c){
  if (arv_vazia(a))
     return 0; /* árvore vazia: não encontrou */
  else
     return a->info==c ||
     arv_pertence(a->esq,c) ||
     arv_pertence(a->dir,c);
}
```

TAD Árvores Binárias – Implementação em C

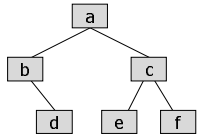
- função arv_imprime
  - percorre recursivamente a árvore, visitando todos os nós e imprimindo sua informação

```
void arv_imprime (Arv* a){
  if (!arv_vazia(a)){
     printf("%c ", a->info); /* mostra raiz */
     arv_imprime(a->esq); /* mostra sae */
     arv_imprime(a->dir); /* mostra sad */
  }
}
```

Exemplo

- Criar a árvore <a <b <> <d <><>> > <c <e <><> > <f <><> > > >

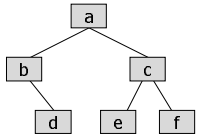
```
/* sub-árvore 'd' */
Arv* a1= arv_cria('d',arv_criavazia(),arv_criavazia());
/* sub-árvore 'b' */
Arv* a2= arv_cria('b',arv_criavazia(),a1);
/* sub-árvore 'e' */
Arv* a3= arv_cria('e',arv_criavazia(),arv_criavazia());
/* sub-árvore 'f' */
Arv* a4= arv_cria('f',arv_criavazia(),arv_criavazia());
/* sub-árvore 'c' */
Arv* a5= arv_cria('c',a3,a4);
/* árvore 'a' */
Arv* a = arv_cria('a',a2,a5 );
```



Exemplo

- Criar a árvore <a <b <> <d <><>> > <c <e <><> > <f <><> > > >

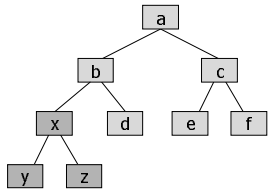
```
Arv* a = arv_cria('a',
          arv_cria('b',
              arv_criavazia(),
              arv_cria('d', arv_criavazia(), arv_criavazia())
          ),
          arv_cria('c',
             arv_cria('e', arv_criavazia(), arv_criavazia()),
             arv_cria('f', arv_criavazia(), arv_criavazia())
          )
);
```



Exemplo

- Acrescenta nós x, y e z

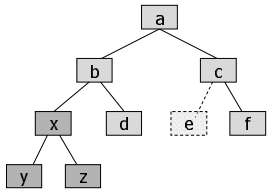
```
a->esq->esq =
  arv_cria('x',
     arv_cria('y',
          arv_criavazia(),
          arv_criavazia()),
     arv_cria('z',
          arv_criavazia(),
          arv_criavazia())
  );
```



Exemplo

- Libera nós

```
a->dir->esq = arv_libera(a->dir->esq);
```



Próxima Aula

- Travessia e busca em uma árvore binária.