CES-11

- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

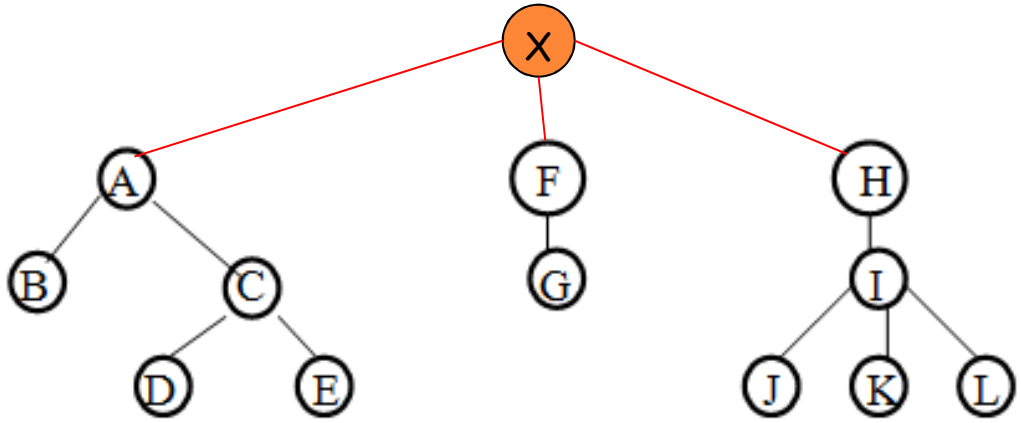
OPERAÇÕES PARA A ÁRVORE

- **noh Pai (n, A):** pai do nó n da árvore A
  - Se n é raiz, Pai é Λ (vazio)
- **noh FilhoEsquerdo (n, A):** filho esquerdo do nó n na árvore A
  - Se n é folha, FilhoEsquerdo é Λ (vazio)
- **noh IrmaoDireito (n, A):** irmão direito do nó n na árvore A
  - Se n é caçula, IrmãoDireito é Λ (vazio)
- **logic Cacula (n, A):** verifica se o nó n é caçula na árvore A

OPERAÇÕES PARA A ÁRVORE

- **tipelemento Elemento (n, A)**: conteúdo em informações do nó n da árvore A.
  - Os nós de uma árvore podem não conter informação alguma.

- **arvore Criação (x, ListaÁrvores):** coloca a informação x num novo nó n e cria uma árvore com raiz n e sub-árvores da raiz pertencentes à ListaÁrvores (Lista linear de árvores).



Operações para a árvore

- **noh Raiz (A):** nó raiz da árvore A
- **void Esvaziar (A):** tornar vazia a árvore A
- **logic Vazia (A):** verificar se a árvore A é vazia ou não
- **void FormarArvore (&A):** formar interativamente a árvore A

- Estas funções são dependentes da estrutura de dados utilizada para armazenar as árvores

Operações para a árvore

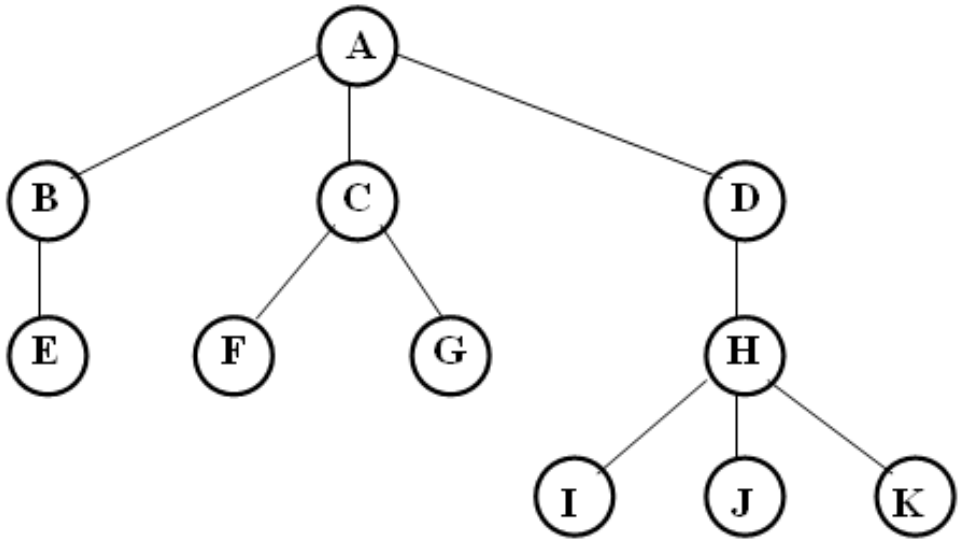
- Outras operações podem ser definidas:
  - Alterar o conteúdo de um nó
  - Fazer um novo nó ser caçula de outro nó
  - Fazer uma árvore ser uma sub-árvore de um nó de outra árvore
  - Deletar uma sub-árvore de um nó
  - Deletar um nó de uma árvore (complicado se não for folha)
  - Inserir um nó numa árvore numa dada posição

Operações para a árvore

- Exemplo 1: Rotina para ordenação pré-ordem

```
void  PreOrdem (noh n, arvore A) {
    noh c;
    Escrever (Elemento (n, A));
    c = FilhoEsquerdo (n, A);
    while ( c != vazio)  {
            PreOrdem (c, A);
            c = IrmaoDireito (c, A);
    }
}
```

PreOrdem (Raiz (A), A)



OPERAÇÕES PARA A ÁRVORE

- Exemplo 2: Cálculo da altura de uma árvore A

- Definição recursiva da altura de um nó n:
  - Se n é vazio, Altura (n) = -1;
  - Senão, se n é folha, Altura (n) = 0;
  - Senão: Altura (n) = 1 + max (Alturas dos filhos de n)

Altura (Raiz (A), A)

Operações para a árvore

- Exemplo 2: Cálculo da altura de uma árvore A

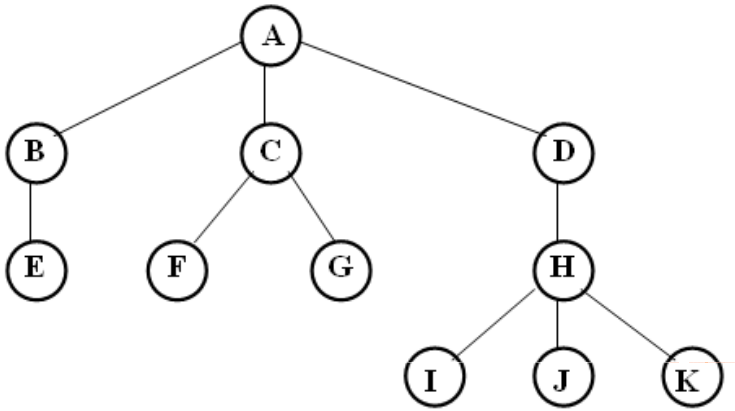
```
int Altura (noh n, arvore A) {
    int maxalt, aux; noh f;
    if (n == vazio) return -1;
    f = FilhoEsquerdo (n, A);
    if (f == vazio) return 0;
    for (maxalt = 0; f != vazio; f = IrmaoDireito (f, A)) {
        aux = Altura (f, A);
        if (aux > maxalt) maxalt = aux;
    }
  return maxlt + 1;  }
```

CES-11

- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

ESTRUTURA CONTÍGUA

- Os nós são armazenados num vetor, numa ordem convencionada (**Pré-ordem**)
- noh é o índice da célula no vetor de células
- Armazena-se o grau do nó, isto é seu número de filhos



Propositalmente vazio, a fim de começar com nó 1

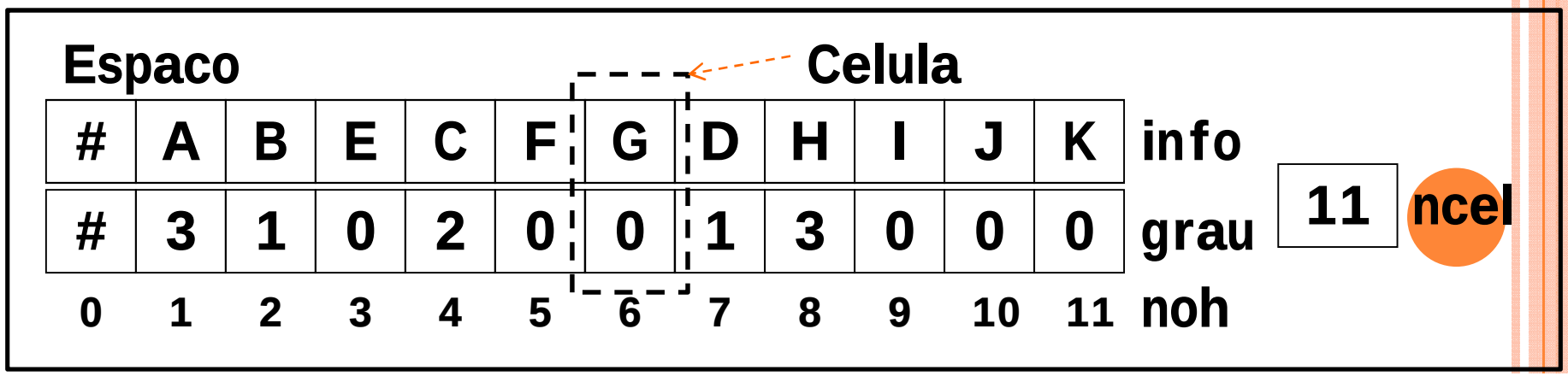
Espaco **Celula**

	#	A	B	E	C	F	G	D	H	I	J	K
info	#	A	B	E	C	F	G	D	H	I	J	K
grau	#	3	1	0	2	0	0	1	3	0	0	0
noh	0	1	2	3	4	5	6	7	8	9	10	11

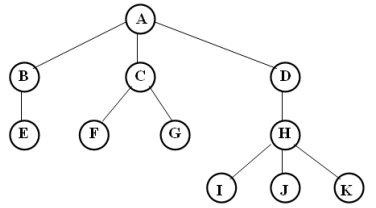
11 ncel

ESTRUTURA CONTÍGUA

```
const int max = 100;
const int vazio = 0;
const int erro= -9999;
typedef int noh;
struct celula {char info; int grau;};
struct arvore {celula Espaco [101]; int ncel;};
```



ESTRUTURA CONTÍGUA



Qual o nó raiz?

```
noh Raiz (arvore A) {
  if (A.ncel == 0) return vazio;
  return 1;
}
```

A árvore está vazia?

```
logic ArvVazia (arvore A) {
      if (A.ncel == 0) return TRUE;
      else return FALSE;
}
```

Esvaziar a árvore

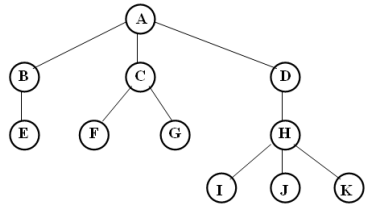
```
void Esvaziar (arvore *A) {A->ncel = 0;}
```

Espaco

#	A	B	E	C	F	G	D	H	I	J	K	info
#	3	1	0	2	0	0	1	3	0	0	0	grau
0	1	2	3	4	5	6	7	8	9	10	11	noh

11 ncel

ESTRUTURA CONTÍGUA



Qual a informação de um dado nó n?

```
char Elemento (noh n, arvore A) {
      if (n <= 0 || n > A.ncel)
            return erro;
      return A.Espaco[n].info;
}
```

Qual o filho esquerdo de um dado nó n?

```
noh FilhoEsquerdo (noh n, arvore A) {
  if (n <= 0 || n > A.ncel) return erro;
  if (A.Espaco[n].grau == 0) return vazio;
  else return n + 1;
}
```

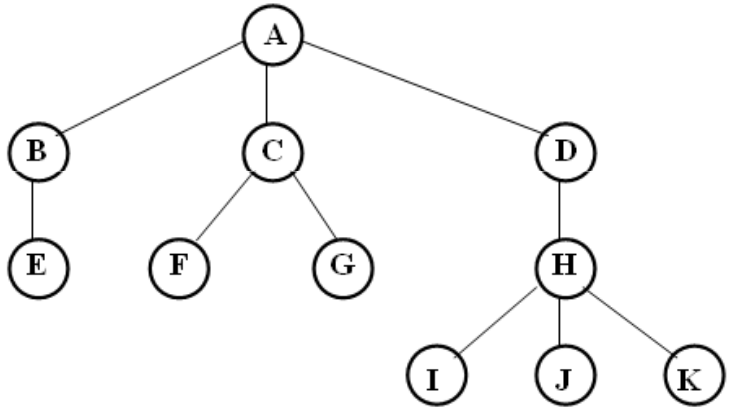
Espaco

#	A	B	E	C	F	G	D	H	I	J	K	info
#	3	1	0	2	0	0	1	3	0	0	0	grau
0	1	2	3	4	5	6	7	8	9	10	11	noh

11 ncel

ESTRUTURA CONTÍGUA

Quem é o pai de um dado nó n?



- Ex: Pai do nó 7 é o 1
- O pai de **n** é um nó à sua esquerda
- Entre **n** e seu pai estão todos os irmãos de **n** mais à esquerda e seus descendentes

  Código não imediato!

Espaco

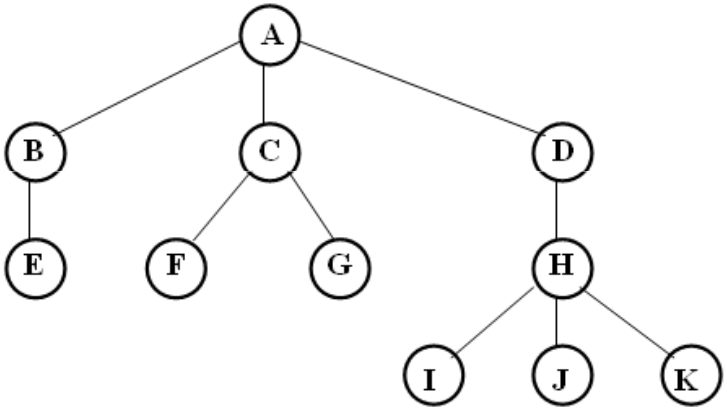
#	A	B	E	C	F	G	D	H	I	J	K	info
#	3	1	0	2	0	0	1	3	0	0	0	grau
0	1	2	3	4	5	6	7	8	9	10	11	noh

11

ncel

ESTRUTURA CONTÍGUA

Quem é o irmão direito um dado nó n?



- Ex: Irmão direito do nó 4 é o 7
- Se **n** não for caçula, seu irmão direito é um nó à sua direita

  Entre **n** e seu irmão direito estão todos os descendentes próprios de **n**

  Código não imediato!

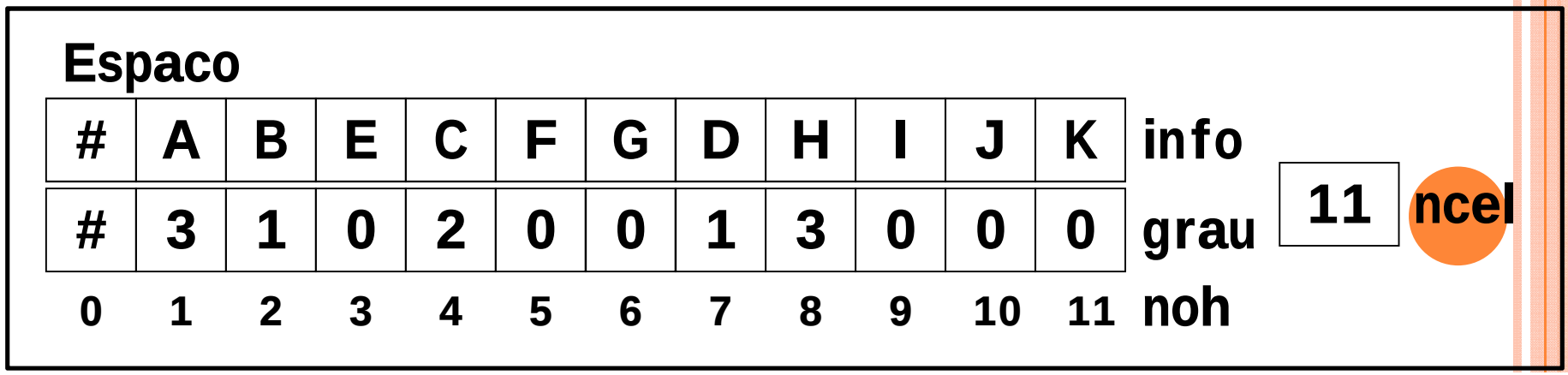
Espaco

#	A	B	E	C	F	G	D	H	I	J	K	info
#	3	1	0	2	0	0	1	3	0	0	0	grau
0	1	2	3	4	5	6	7	8	9	10	11	noh

11 ncel

Estrutura contígua

- Pontos fortes
  - Gasta relativamente pouca memória
  - Serve para armazenamento permanente
- Ponto fraco
  - Operações ineficientes

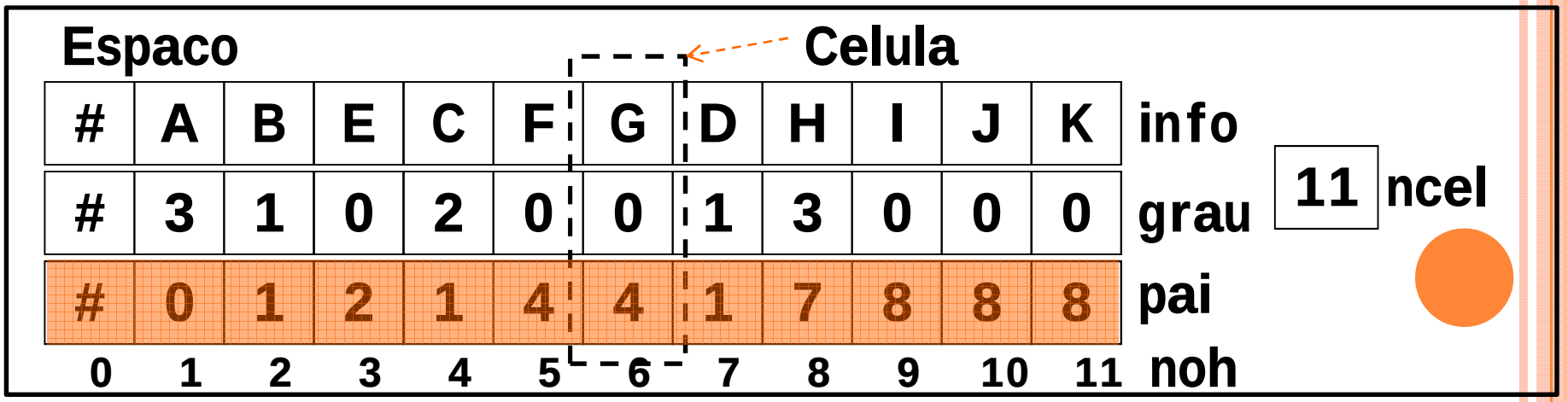


CES-11

- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

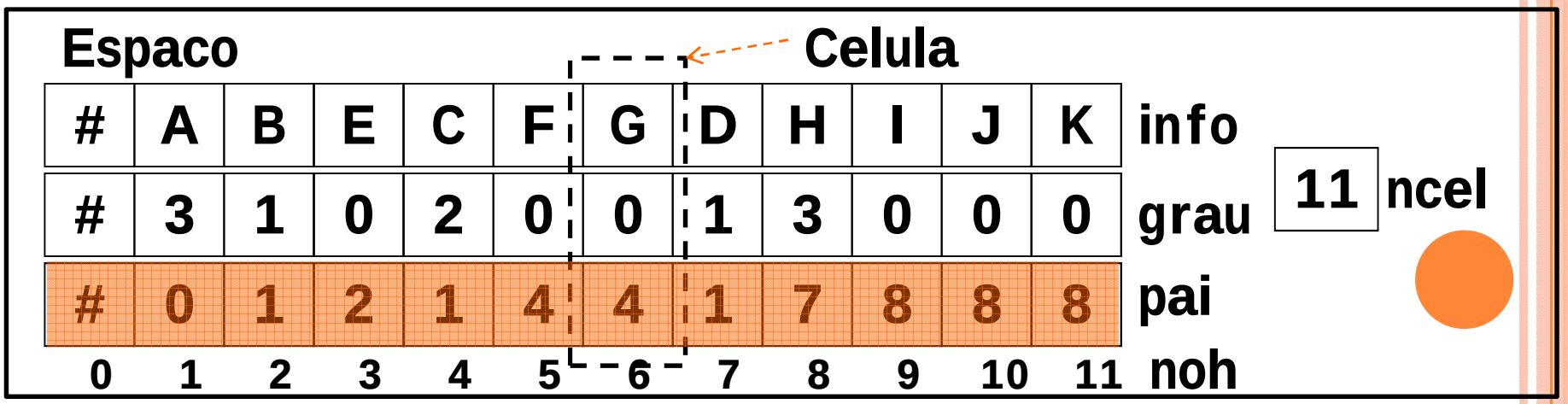
ESTRUTURA CONTÍGUA MELHORADA

- A estrutura contígua pode ser sofisticada incluindo-se nas células informações de caráter operacional
  - No caso: informação do Pai de cada nó



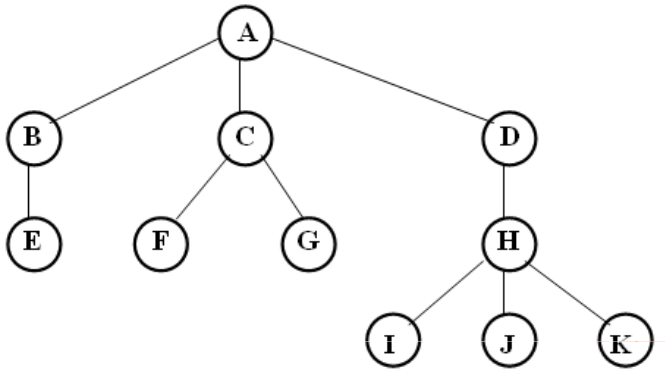
ESTRUTURA CONTÍGUA MELHORADA

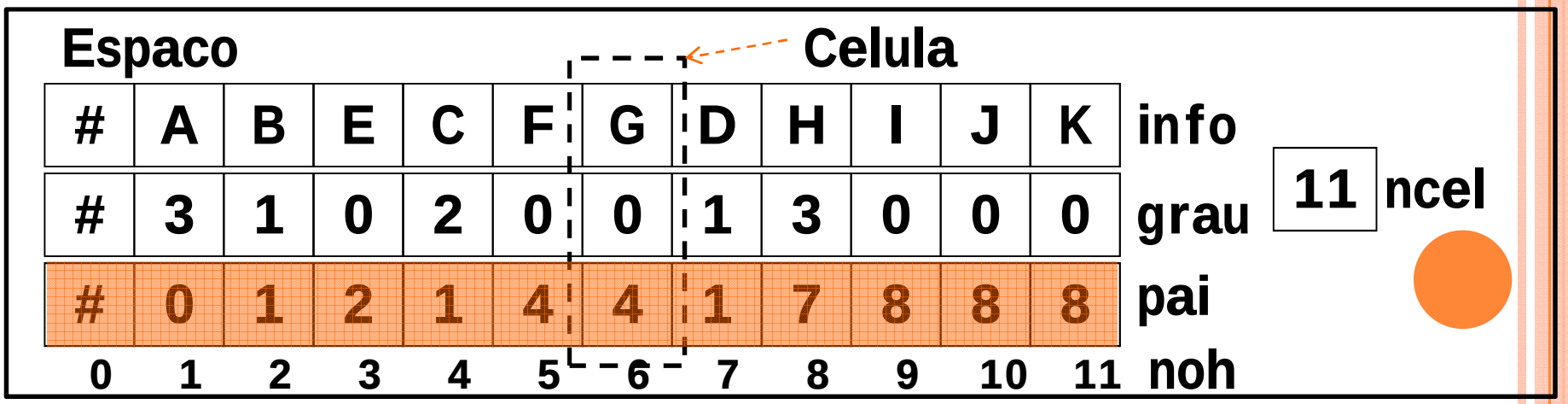
```
const int max = 100;
const int vazio = 0;
const int absurdo = -9999;
typedef int noh;
struct celula  {char info; int grau, pai;};
struct arvore {celula Espaco [101]; int ncel;};
```



Estrutura contígua melhorada

- Quem é o **pai** de um dado nó n?
  - Ex: Pai do nó 7 é o 1
  - Agora é elementar, pois a informação já consta na célula
- Quem é o **irmão direito** de um dado nó n?
  - Ex: Irmão direito do nó 4 é o 7
  - Basta encontrar o primeiro nó à direita de **n** que tenha o mesmo pai de **n**



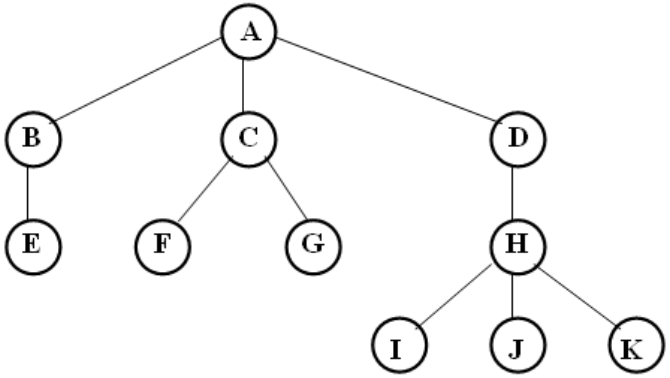


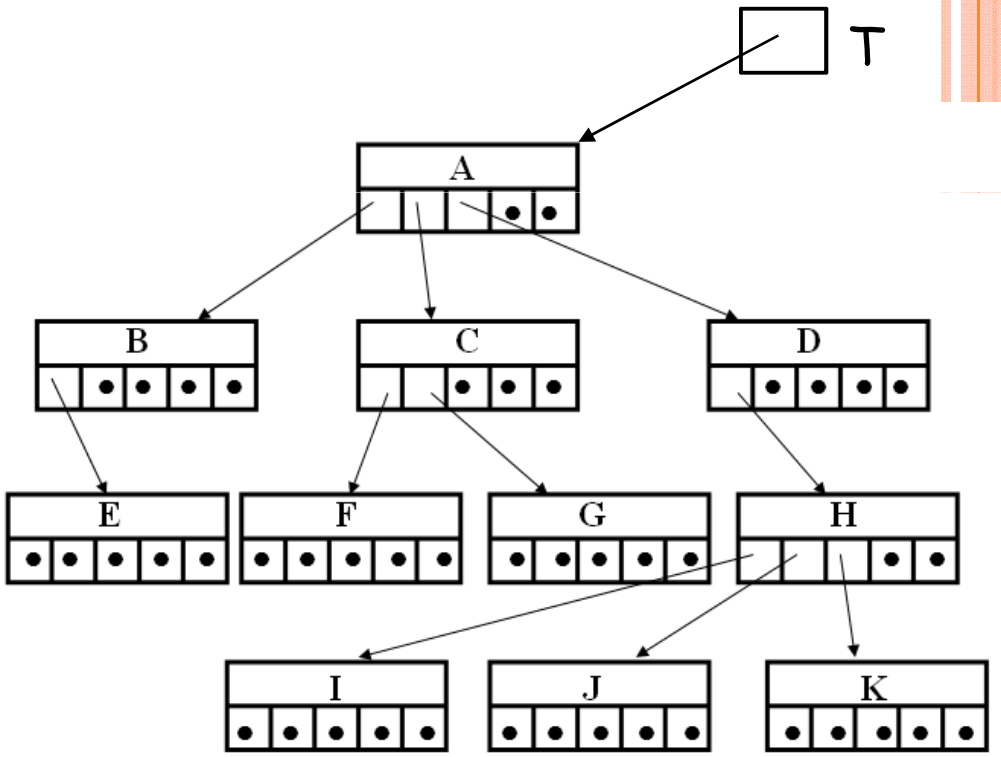
CES-11

- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

ESTRUTURA ENCADEADA DIRETA

- Declarações:
  - **Nó:** ponteiro para célula raiz
  - **Árvore:** ponteiro para a célula da raiz

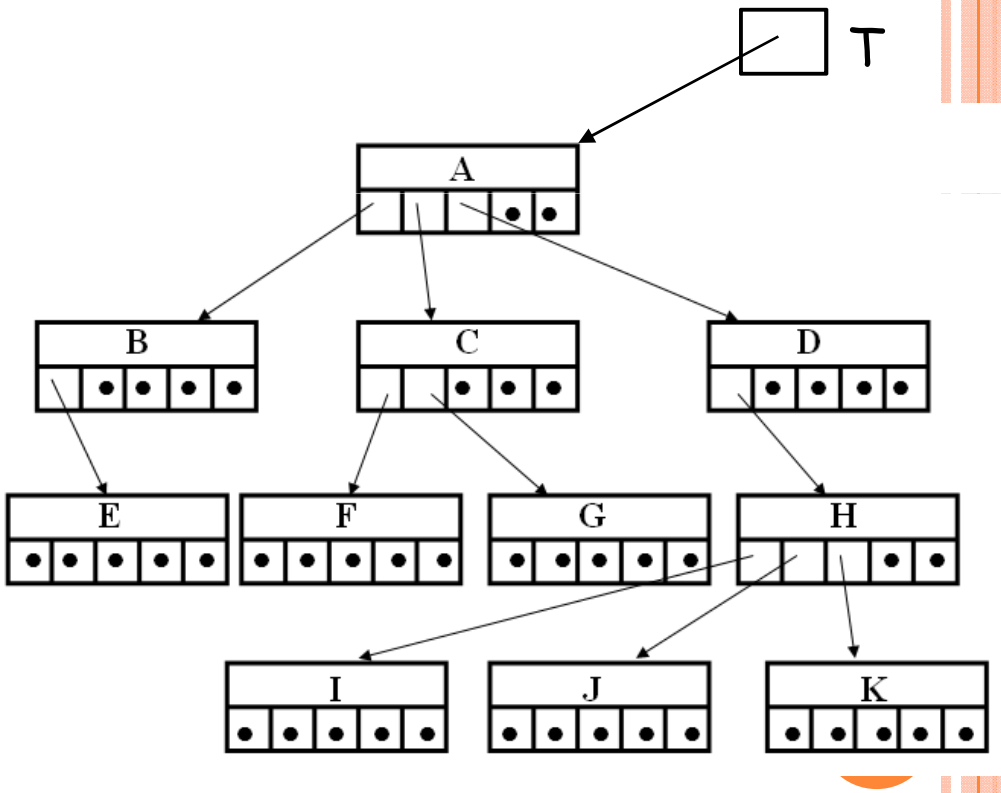




ESTRUTURA ENCADEADA DIRETA

- Pontos fracos:
  - Usa muitos ponteiros desnecessários. Ex: folhas
  - Limita o número de filhos

```
const int maxfilhos = 5;
typedef celula *noh;
typedef celula *arvore;
struct celula {
   informacao info;
   noh Filhos[maxfilhos];
};
arvore T;
```

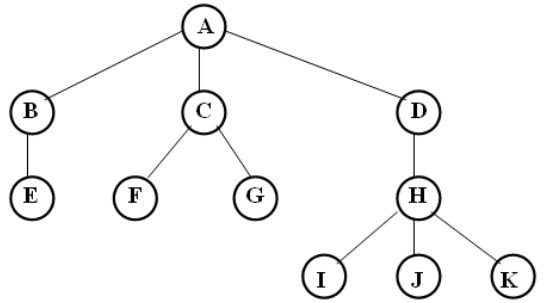


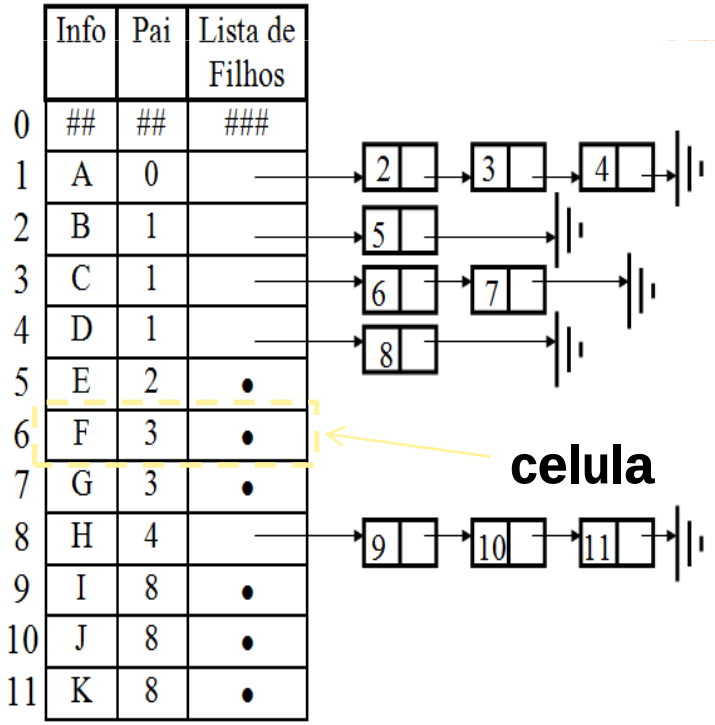
CES-11

- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

ESTRUTURA COM LISTA DE FILHOS

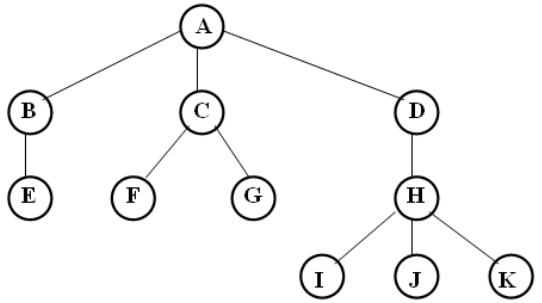
- Nó: índice num vetor de células
  - Não necessariamente a raiz fica no nó 1
  - Os nós podem ficar embaralhados no vetor, sem ordenação (pré-ordem, etc.)
- Filhos: ordenados da esquerda para a direita numa lista
  - Estrutura das listas: contígua, encadeada, etc

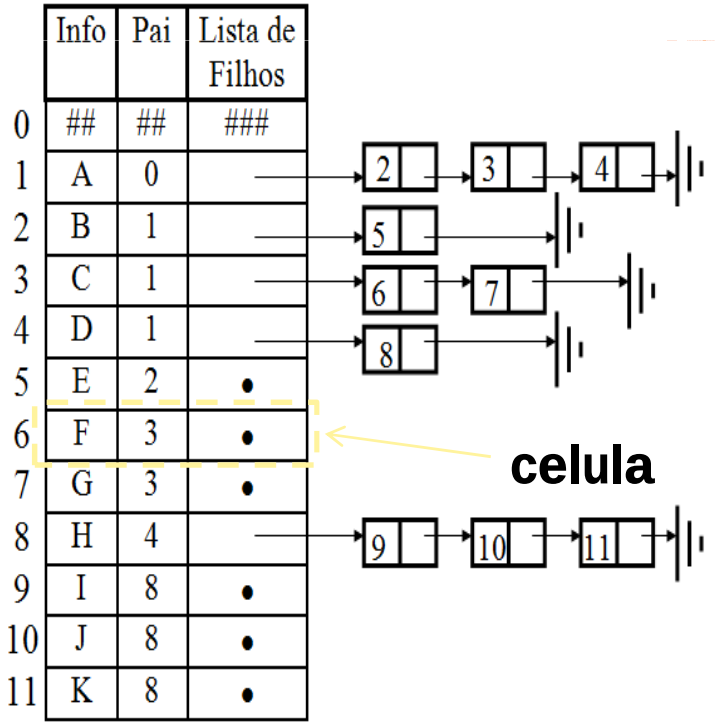




ESTRUTURA COM LISTA DE FILHOS

```
const int vazio = 0;
typedef int noh;
typedef celfilho *lista;
typedef celfilho *posicao;
struct celfilho  {
      noh filho;
      celfilho *prox;
};
struct celula {
      char info;
      noh pai;
      lista list;
};
struct arvore  {
      noh Raiz;
      celula Espaco[51];
      int ncel;
};
```

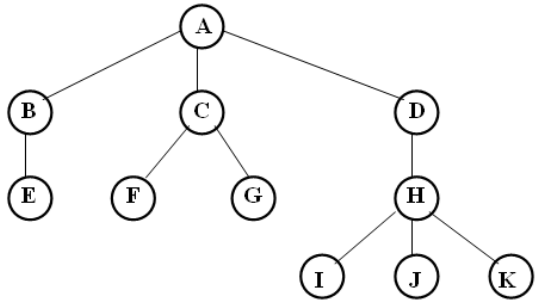




ESTRUTURA COM LISTA DE FILHOS

Qual o nó raiz?

```
noh raiz (arvore A) {
    if (A.ncel == 0)
        return vazio;
    return A.raiz;
}
```

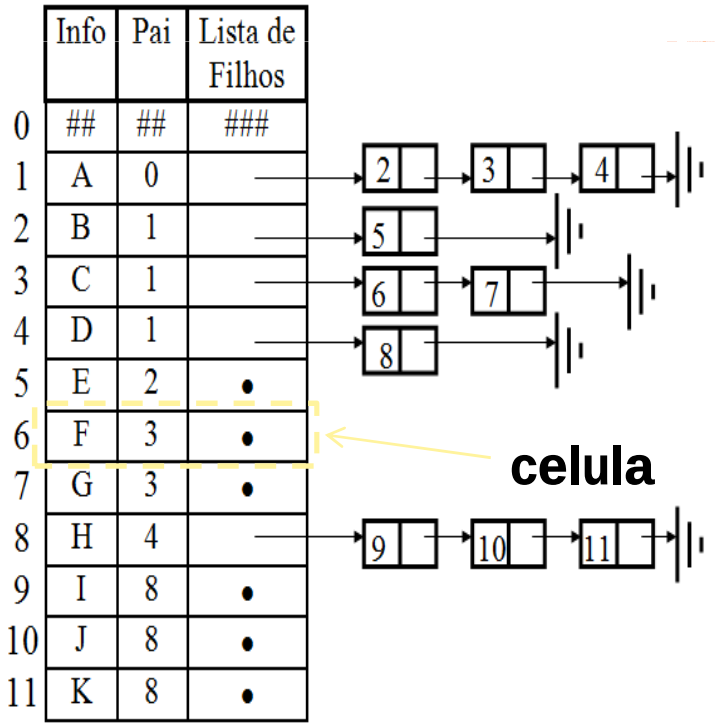


Esvaziar a árvore

```
void Esvaziar (arvore &A){
  //Liberar todas as
  //células de  filhos;
  A.raiz = vazio;
  A.ncel = 0;
}
```

Qual a informação de um dado nó n?

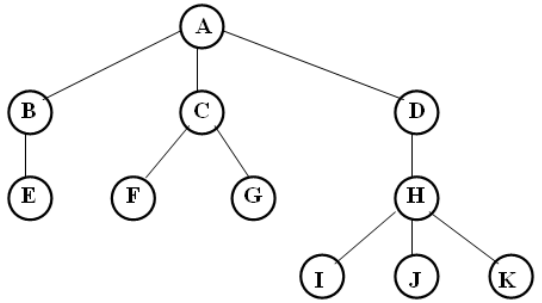
```
char Elemento (noh
n, arvore A) {
    return
A.Espaco[n].info;
}
```



ESTRUTURA COM LISTA DE FILHOS

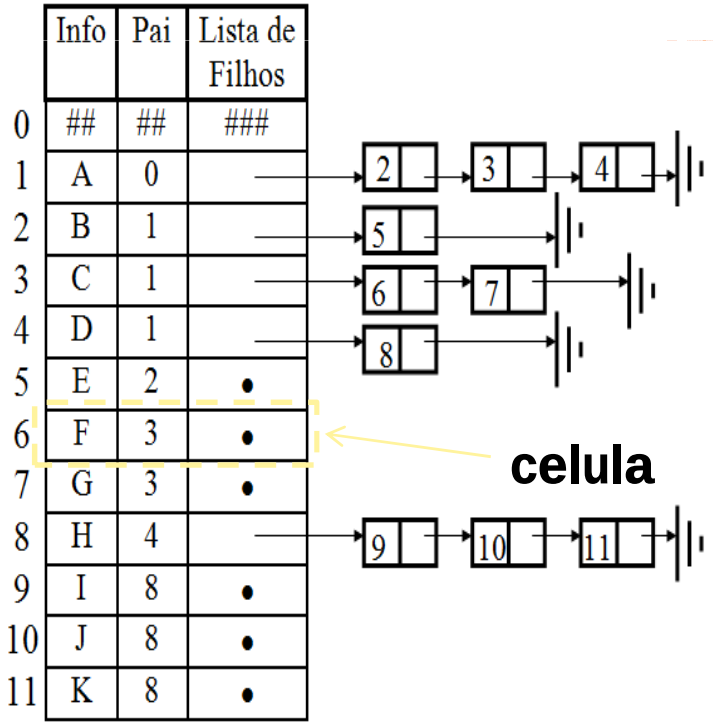
Qual o pai de um dado nó n?

```
noh Pai (noh n, arvore A) {
        return A.Espaco[n].pai;
}
```



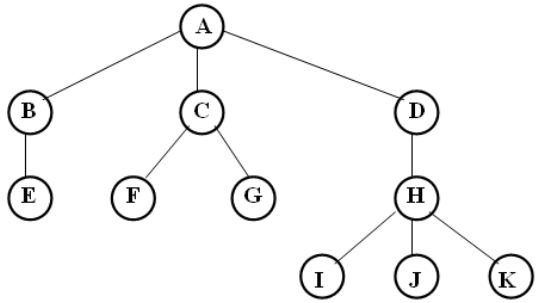
Qual o filho esquerdo de um dado nó n?

```
noh FilhoEsquerdo (noh n, arvore A) {
    if (A.Espaco[n].list == NULL)
        return vazio;
    else
        return A.Espaco[n].list->filho;
}
```

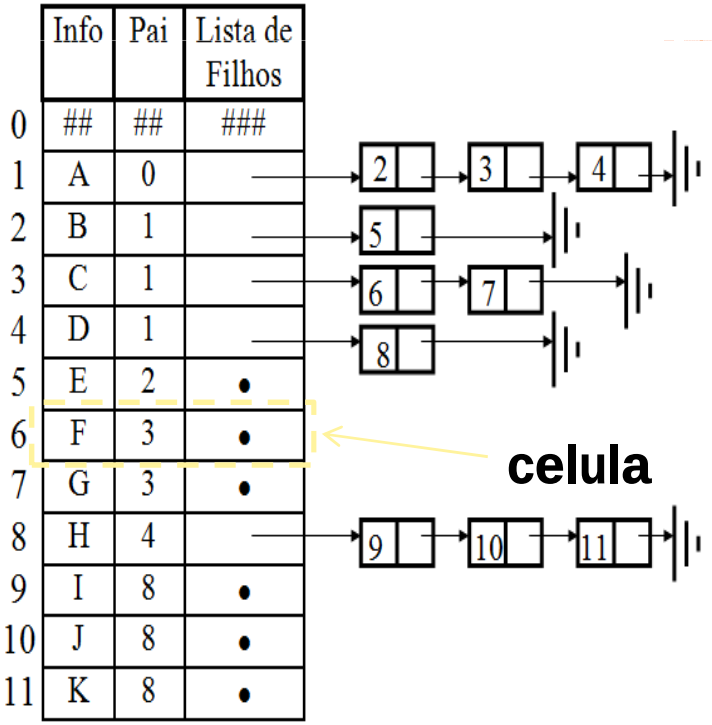


ESTRUTURA COM LISTA DE FILHOS

Qual o irmão direito de um dado nó n?




```
noh IrmaoDireito (noh n, arvore A)
{
   noh pai; posicao idir, p;
   if (n == A.Raiz) return vazio;
   pai = A.Espaco[n].pai;
   p = A.Espaco[pai].list;
   while (p->filho != n)
      p = p->prox;
   idir = p->prox;
   if (idir == NULL) return vazio;
   else return idir->filho;
}
```



Estrutura com lista de filhos

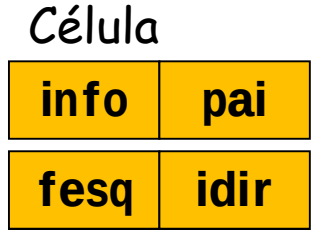
- Pontos Fortes
  - Gasta relativamente pouca memória
  - Maioria das operações com código simples e eficiente
- Pontos Fracos
  - Operação Irmão-direito complexa ...O(n)
  - Limitação no número de nós (lista de nós em estrutura contígua)

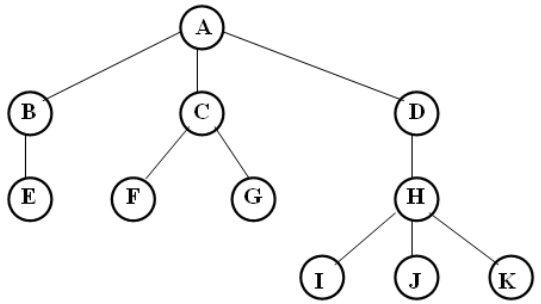
CES-11

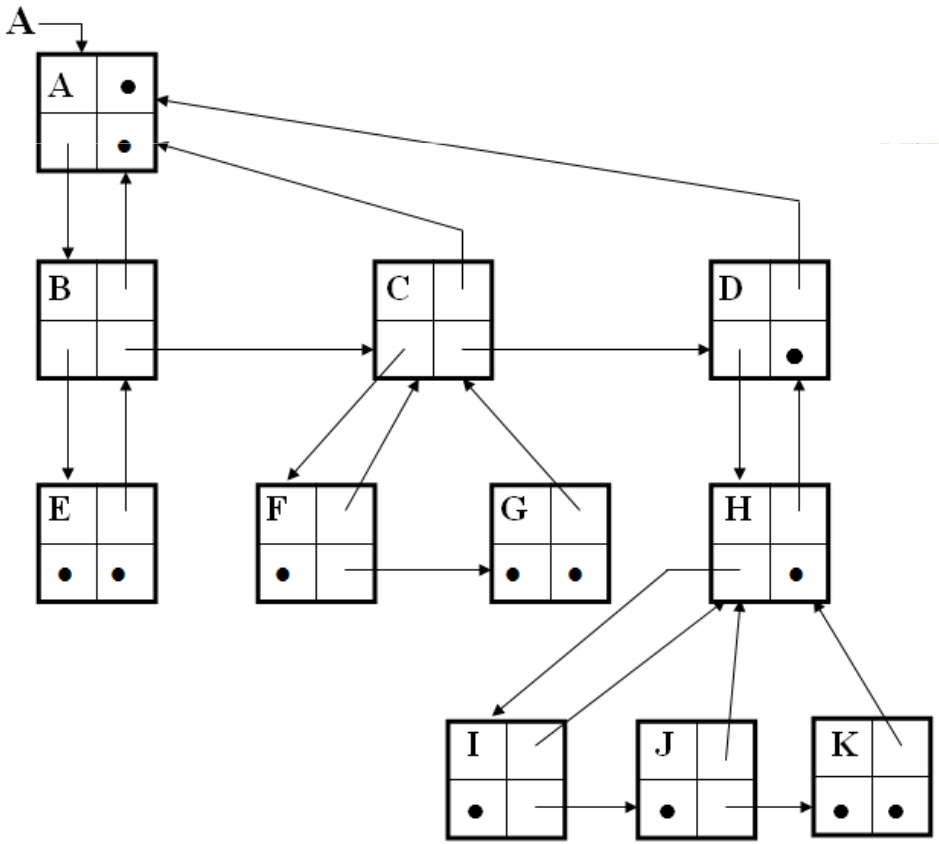
- Árvores
  - Operações para a árvore
  - Estruturas para a árvore
    - Estrutura contígua
    - Estrutura contígua melhorada
    - Estrutura encadeada direta
    - Estrutura com listas de filhos
    - Estrutura com encadeamento de pais e irmãos

ESTRUTURA COM ENCADEAMENTO DE PAIS E IRMÃOS

```
const celula *vazio = NULL;
typedef celula *noh;
typedef celula *arvore;

struct celula {
    char info;
    noh pai, fesq, idir;
};
arvore A1, A2, A3;
```





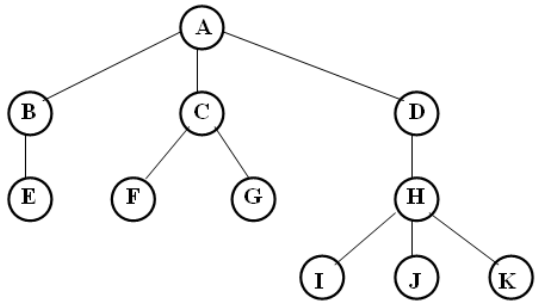


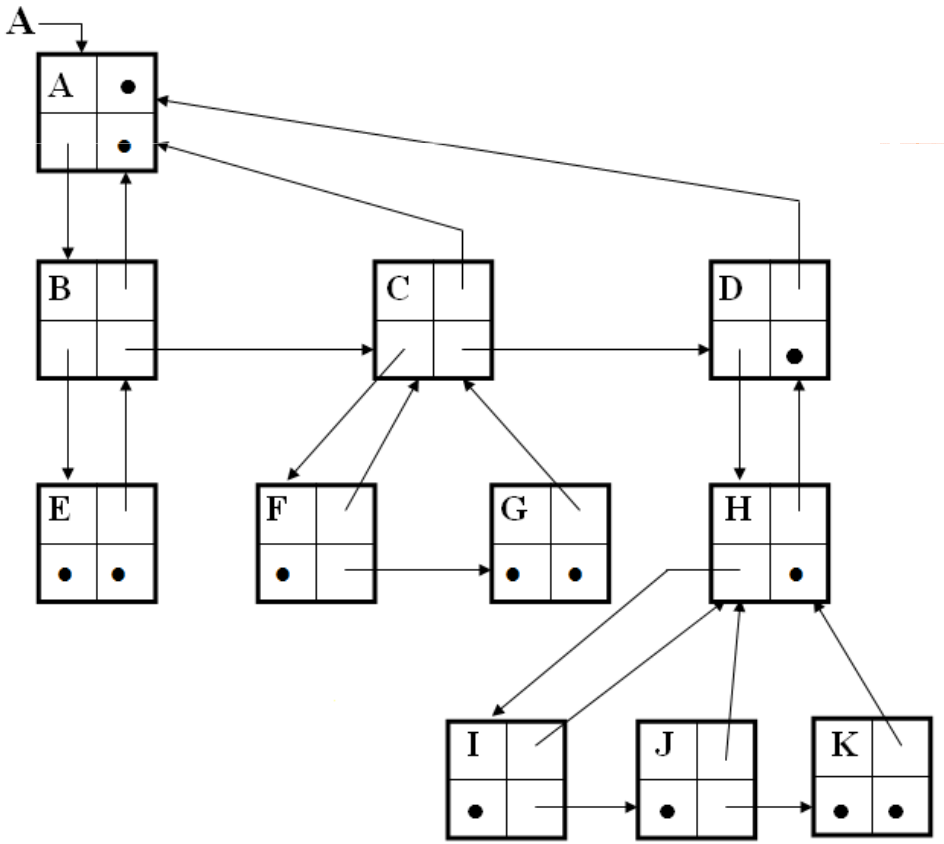
ESTRUTURA COM ENCADEAMENTO DE PAIS E IRMÃOS

```
FilhoEsquerdo (noh n,arvore A):
    n->fesq;

IrmãoDireito (noh n,arvore A):
    n->idir;

Elemento (noh n,arvore A):
    n->info;

Pai(noh n,arvore A):
    n->pai;

Raiz (arvore A):  A;
```

Célula

info	pai
fesq	idir



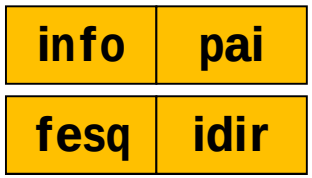


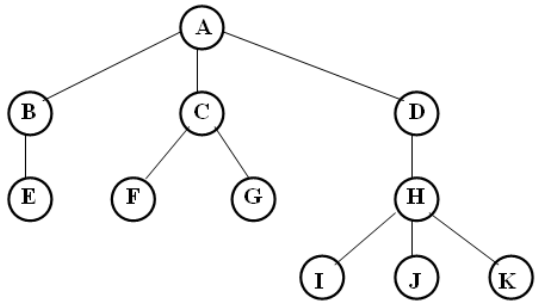
ESTRUTURA COM ENCADEAMENTO DE PAIS E IRMÃOS

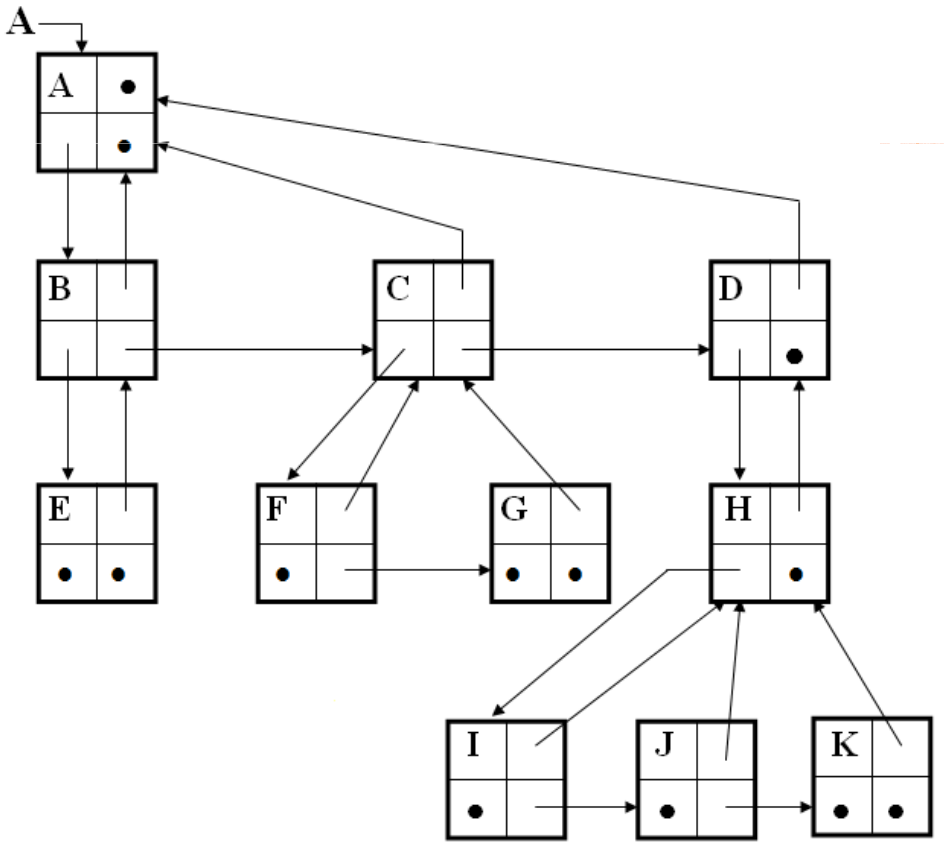
Como esvaziar a árvore?

```
void Esvaziar (noh *n) {
    if (n->fesq == NULL
    && n->idir == NULL)
        free (*n);
    else if (n->idir == NULL) {
        Esvaziar ((*n)->fesq);
        free (*n);
    }
    else {
        Esvaziar ((*n)->idir);
        if ((*n)->fesq != NULL)
            Esvaziar ((*n)->fesq);
            free (*n);
        }
        *n = NULL;
}
```

Célula

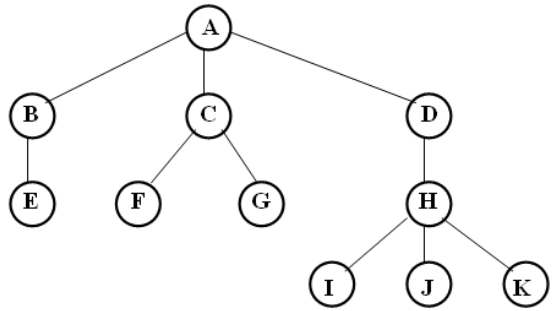






Estrutura com encadeamento de pais e irmãos

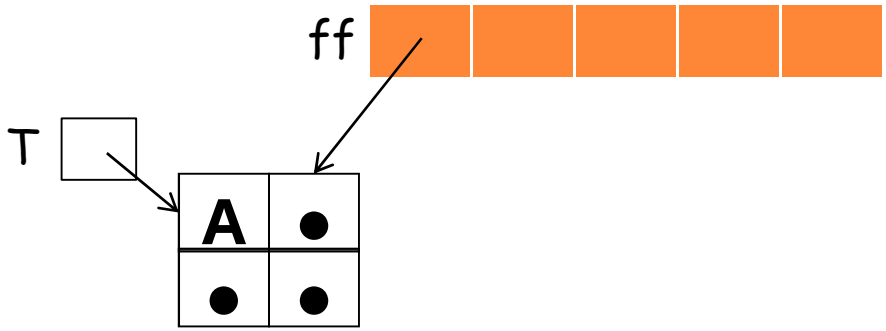
- Leitura iterativa de uma árvore

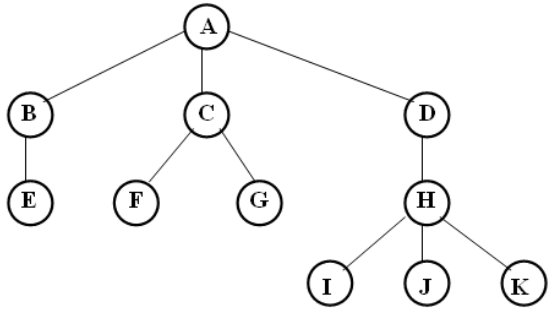



```
A arvore tem raiz? (s/n): s

Digite a informacao da raiz: A

Quantos filhos tem A? 3
1.o filho de A: B
2.o filho de A: C
3.o filho de A: D

Quantos filhos tem B? 1
1.o filho de B: E

Quantos filhos tem C? 2
1.o filho de C: F
2.o filho de C: G

Quantos filhos tem D? 1
1.o filho de D: H

Quantos filhos tem E? 0

Quantos filhos tem F? 0

Quantos filhos tem G? 0

Quantos filhos tem H? 3
1.o filho de H: I
2.o filho de H: J
3.o filho de H: K

Quantos filhos tem I? 0

Quantos filhos tem J? 0

Quantos filhos tem K? 0
```

Estrutura com encadeamento de pais e irmãos

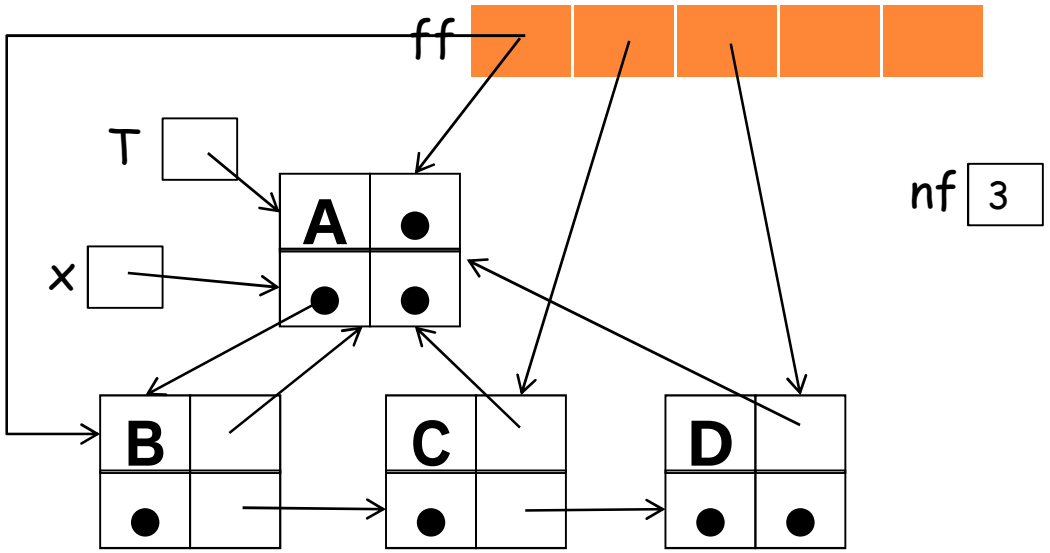
- Leitura iterativa de uma árvore
  - Implementação
    - Idéia: Usar uma fila **ff** para guardar os nós já introduzidos na estrutura, para os quais não se indagou ainda sobre o número e o nome de seus filhos.
    - Perguntar pela raiz, alocando-a e colocando-a em **ff**

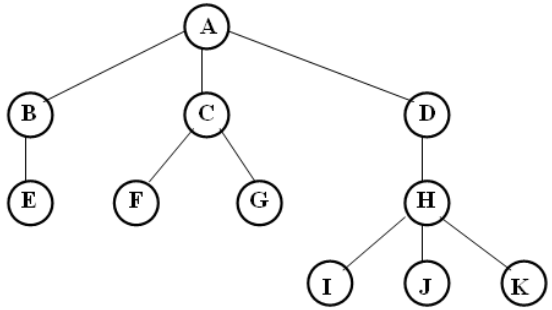




Estrutura com encadeamento de pais e irmãos

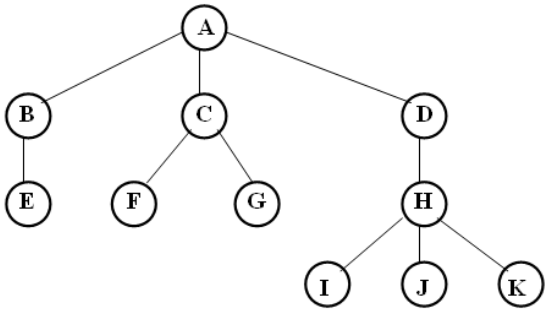
- Leitura iterativa de uma árvore
  - Implementação
    - Enquanto fila **ff** não vazia:
    - Eliminar o primeiro elemento de **ff** e copiá-lo para o nó avulso **x**
    - Perguntar quantos filhos tem este nó, guardando em **nf**
    - Alocar cada filho na árvore e colocá-lo também na fila **ff**





Estrutura com encadeamento de pais e irmãos

- Leitura iterativa de uma árvore
  - Implementação
    - O loop na fila **ff** até esvaziá-la, pois assim todos os nós da árvore foram armazenados.
    - Código em C fica como exercício.
  - Uma forma mais elegante de armazenar uma árvore é receber como entrada sua forma parentética
    - Ex: (A (B (E)) (C (F) (G)) (D (H (I) (J) (K))))



Fim