Árvores – Parte 1

Aleardo Manacero Jr.

DCCE/UNESP

Grupo de Sistemas Paralelos e Distribuídos

Árvores – uma introdução

- As listas apresentadas na aula anterior formam um conjunto de TADs extremamente importante

- Entretanto, padecem de problemas bastante severos de eficiência (velocidade de acesso) quando o número de elementos a ser armazenado cresce muito

- Um TAD que permite melhores resultados do ponto de vista de velocidade de acesso é definido através da estrutura chamada **árvore**

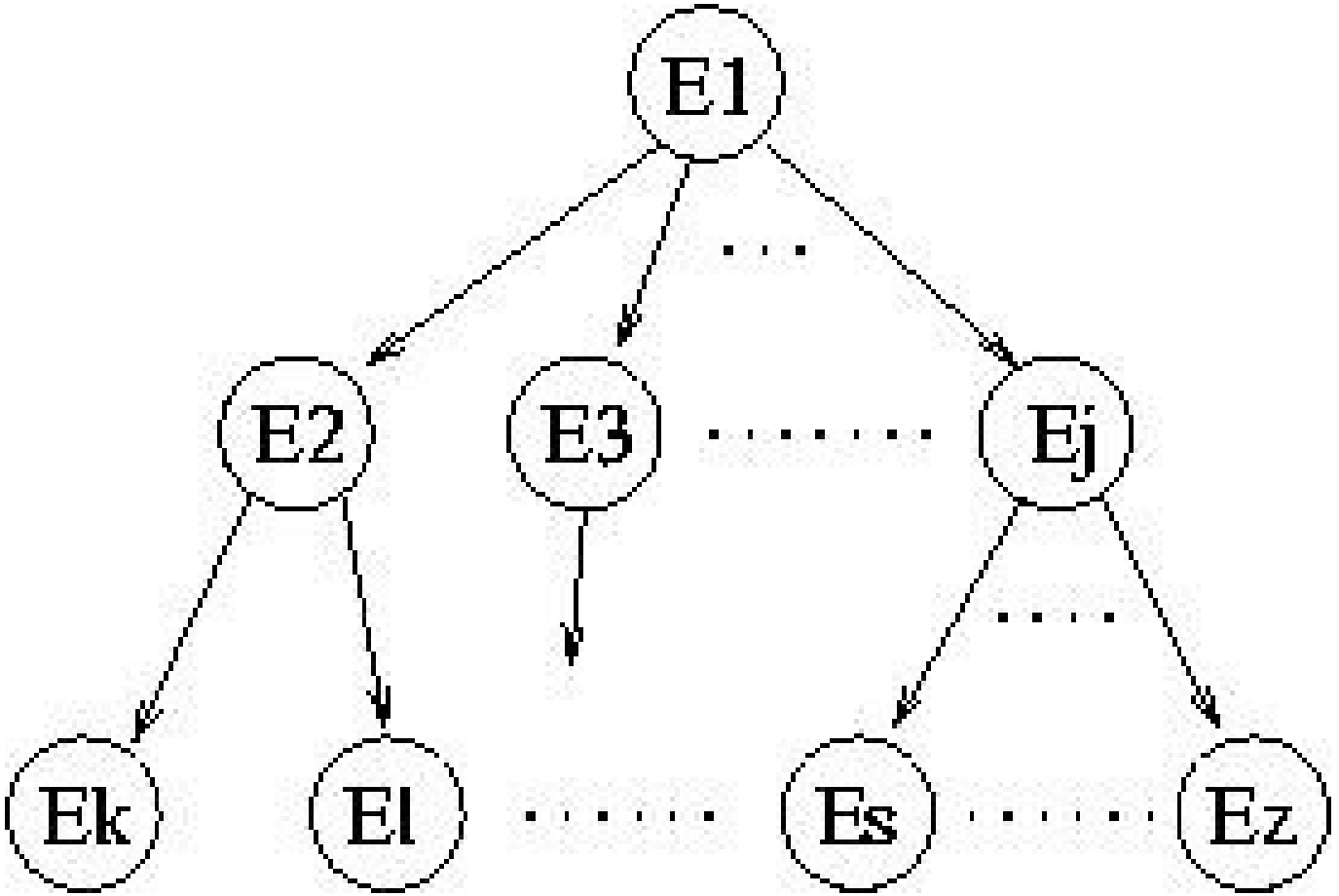
Árvores – uma introdução

- Uma árvore é, na realidade, uma especialização de uma TAD mais genérica, que é o grafo

- Árvores são grafos em que existe apenas uma origem e não se pode formar ciclos

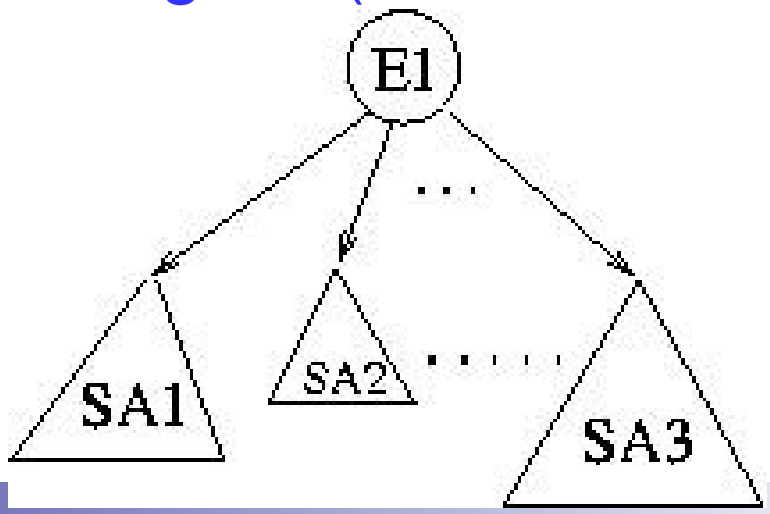
- Existem vários tipos de árvores, definidos a partir da quantidade de “filhos” que um elemento por ter e de como os elementos são arranjados dentro da árvore

Árvores – uma introdução



Árvores – uma introdução

- Numa árvore genérica com *N* elementos temos um vértice raiz (E1 no exemplo anterior) e *N-1* vértices que descendem, de algum modo, do raiz
- Cada aresta que parte do vértice raiz gera uma nova sub-árvore, que deve apresentar as mesmas características da árvore original (vértice raiz e seus descendentes)



Árvores – uma introdução

- Num TAD do tipo árvore os elementos são organizados de forma a definir um critério específico de inserção, busca e caminho ao longo da árvore

- Essa definição permite que se criem técnicas eficientes para busca (atendendo aos critérios de inserção)

Árvores – uma introdução

- Assim, para o TAD é necessário definir operadores como inserção, remoção, percurso e busca

- Dependendo das especificidades de cada árvore outros operadores são necessários, em particular o de balanceamento da árvore

Árvores – uma introdução

- Para que uma árvore seja, de fato, um mecanismo eficiente, é preciso que os seus elementos estejam distribuídos de forma relativamente homogênea pela estrutura (suas sub-árvores)
- Como as operações de inserção e remoção resultam em posições aleatórias, é preciso ter um operador capaz de manter os elementos distribuídos de forma homogênea entre as sub-árvores
- Esse é o operador de **balanceamento**

Árvores – tipos específicos

- Como dito, existem vários tipos de árvores. Os principais tipos incluem:
  - Árvores binárias
  - Árvores binárias balanceadas (AVL, rubro-negras, *splay-trees*)
  - Árvores B (e variantes)
  - Árvores *trie*
  - Árvores multidimensionais (*k-d-trees*)
- Examinaremos cada uma dessas árvores separadamente

Árvores binárias

- Árvore binária é aquela em que cada elemento tem, no máximo, dois descendentes

- Com isso podemos caracterizar três tipos de elementos: raiz, folhas e vértices intermediários

- Nessa árvore, a partir de cada vértice, escolhe-se uma sub-árvore para percorrer a partir de um critério simples de decisão, tipo “maior que”, “precede”, etc

Árvores binárias - Inserção

- O processo de inserção numa árvore binária ocorre sempre com a geração de um novo nó folha

- A localização do ponto da árvore em que se dará a inserção depende do critério de escolha da sub-árvore, sendo um processo tipicamente recursivo (embora sua implementação não precise ser assim)

Árvores binárias - Inserção

- O processo de inserção parte do nó raiz (supondo que ele já exista, caso contrário ele será o elemento a ser inserido)

- A partir do critério específico de ordenação da árvore decide-se qual sub-árvore será percorrida

- Dentro da nova sub-árvore repete-se o procedimento considerando-se a raiz dessa nova sub-árvore, até que se chegue a uma sub-árvore vazia (sem raiz)

Árvores binárias - Remoção

- O processo de remoção é mais elaborado, uma vez que envolve (possivelmente) a junção de duas sub-árvores numa única sub-árvore

- Isso implica então em três possíveis situações para remoção de um elemento da árvore, segundo ele:
  - Possua duas sub-árvores descendentes
  - Possua uma sub-árvore descendente
  - Não possua sub-árvores descendentes

Árvores binárias - Remoção

- Para os casos com um ou nenhum descendente o processo é simples

- Quando o nó a ser retirado tiver apenas uma sub-árvore descendente basta fazer o “pai” desse nó como novo “pai” dessa sub-árvore

- Quando o nó não tiver descendentes basta remove-lo diretamente, fazendo com que seu “pai” deixe de ter filhos naquela sub-árvore

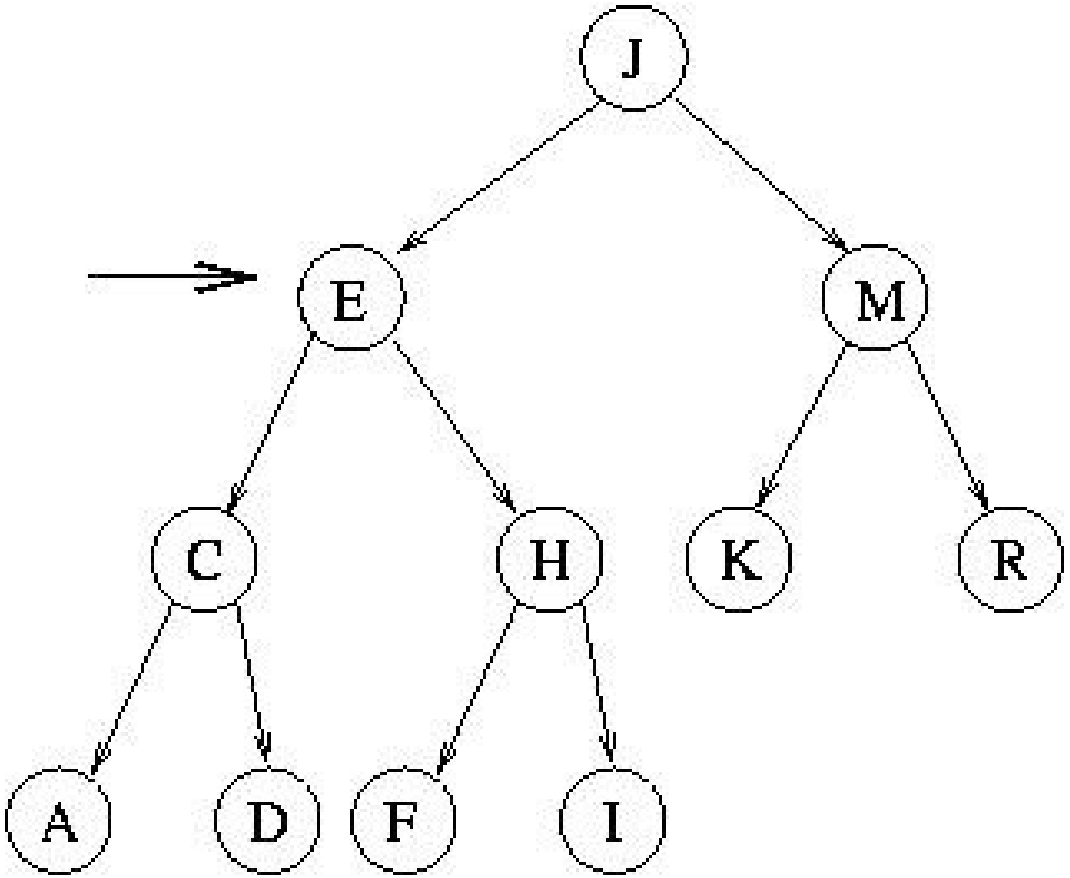
Árvores binárias - Remoção

- Quando o nó a ser retirado tiver duas sub-árvores como descendentes, então o processo não é tão imediato (embora simples)

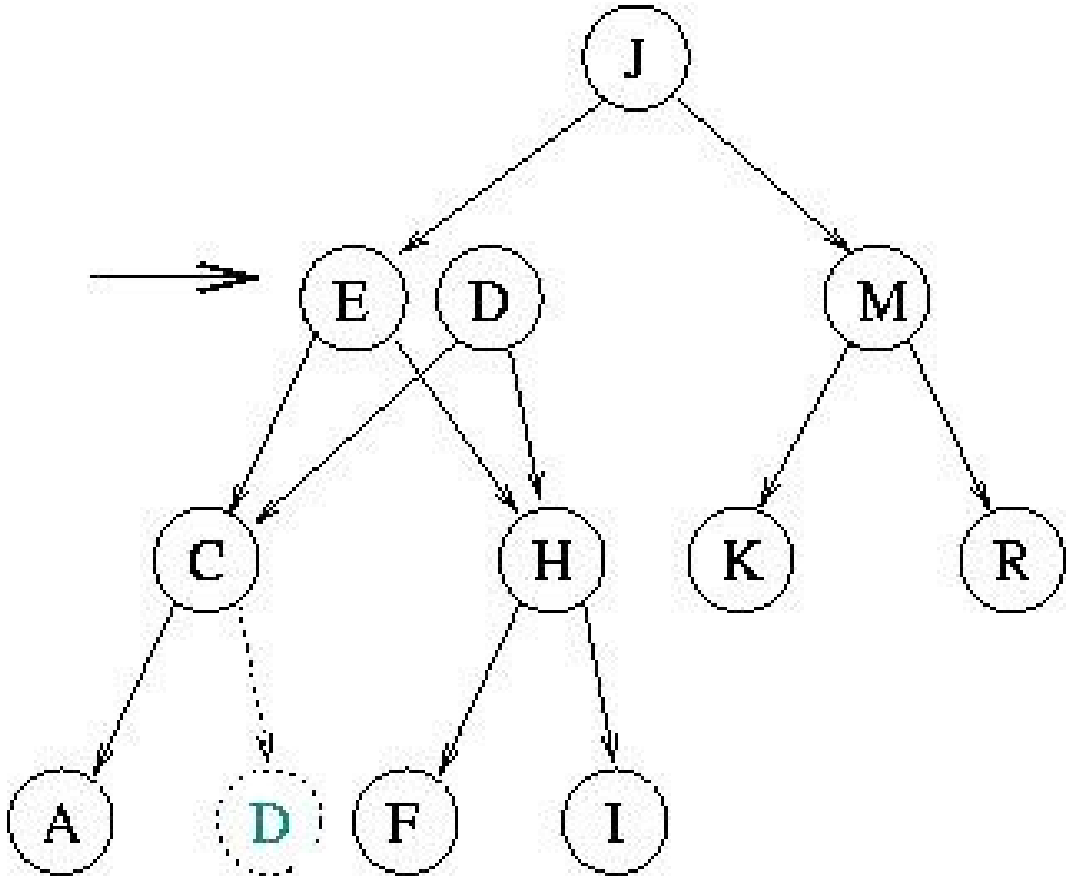
- A idéia aqui é substituí-lo pelo vértice folha que o anteceda imediatamente com a aplicação da regra de ordenação existente

- Outra possibilidade é usar o vértice folha que o imediatamente suceda

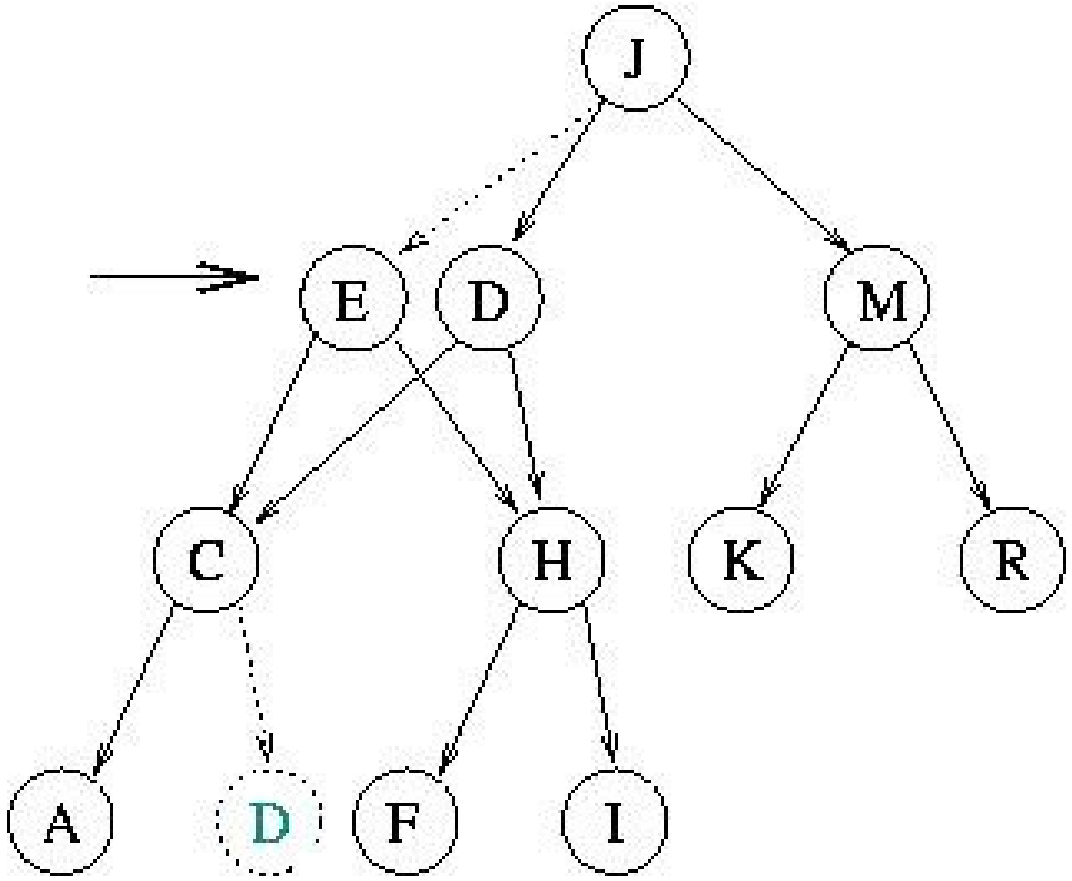
Árvores binárias - Remoção



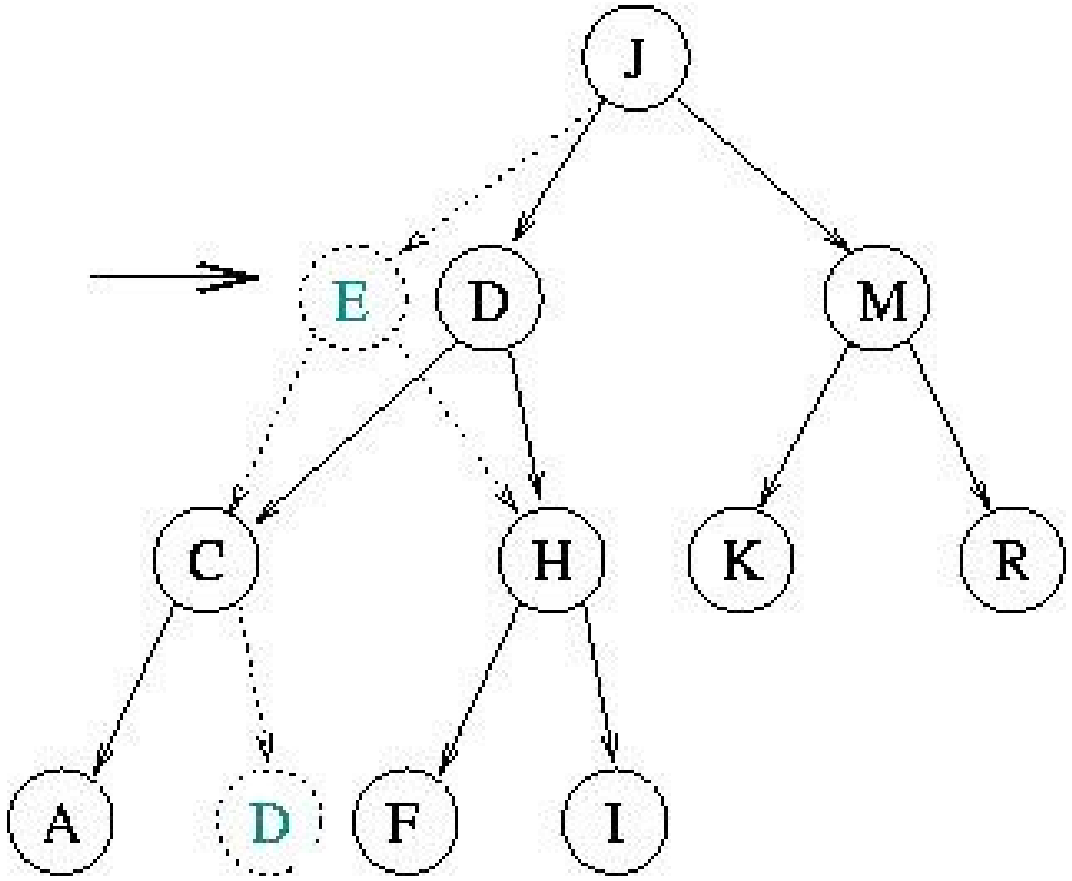
Árvores binárias - Remoção



Árvores binárias - Remoção



Árvores binárias - Remoção



Árvores binárias - Percurso

- Tanto a operação de inserção quanto a de remoção implicam em percorrer a árvore até identificar o ponto em que a operação de fato ocorrerá

- Além disso, a operação de busca por um elemento na árvore também depende de percorrer trechos da árvore

- Assim, é preciso especificar como uma árvore binária pode ser percorrida e o que isso implica do ponto de vista da aplicação de árvores binárias

Árvores binárias - Percurso

- Árvores podem ser percorridas de diversas formas:
  - Em ordem direta
  - Em ordem inversa
  - Em profundidade
  - Em largura

- A escolha por uma ou outra forma de percurso depende fundamentalmente da aplicação da árvore

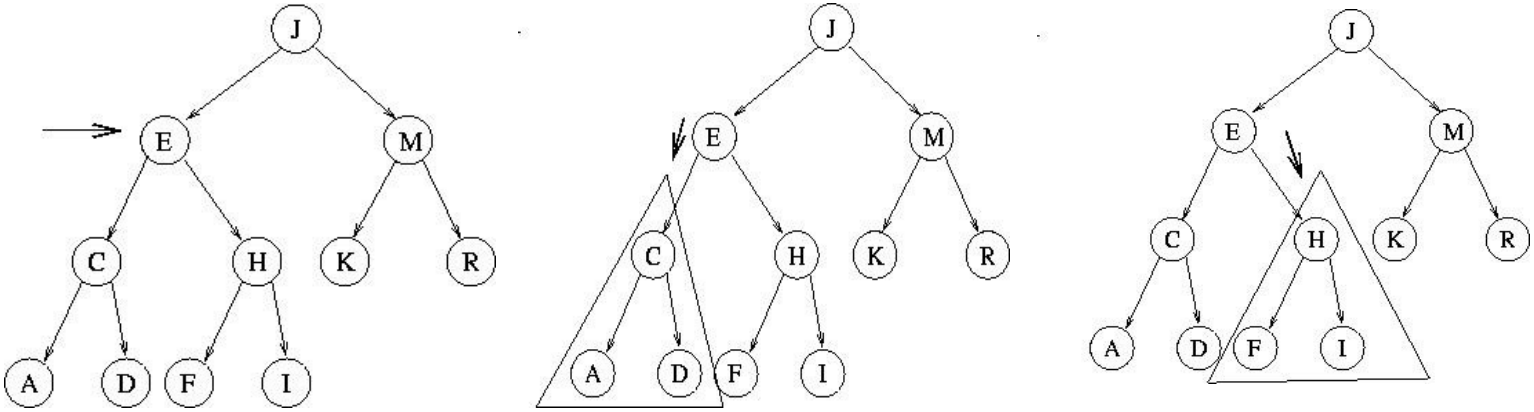
Árvores binárias - Percurso

- Um exemplo clássico é o uso de árvores de decisão em inteligência artificial, gerando técnicas de busca em largura, em profundidade e variações dessas técnicas (algoritmo A*, por exemplo)

- Apresentaremos aqui os percursos tradicionais para árvores binárias de busca, que são os percursos em ordem, pré-ordem e pós-ordem

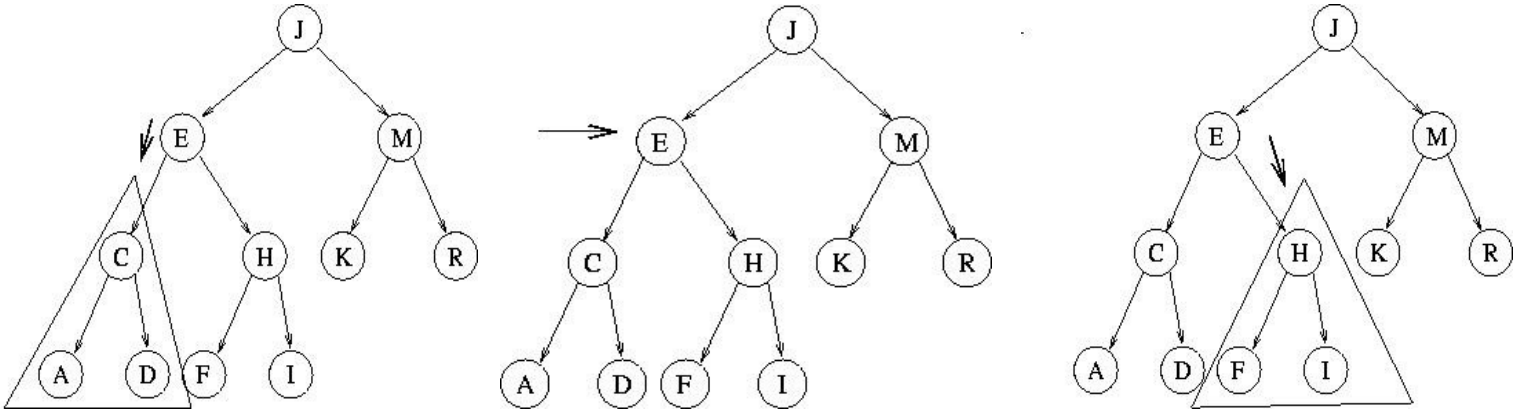
Árvores binárias – Percurso pré-ordem

- Nesse percurso caminha-se na árvore visitando inicialmente um nó, depois visita-se a sua sub-árvore a esquerda, passando-se finalmente para sua sub-árvore direita



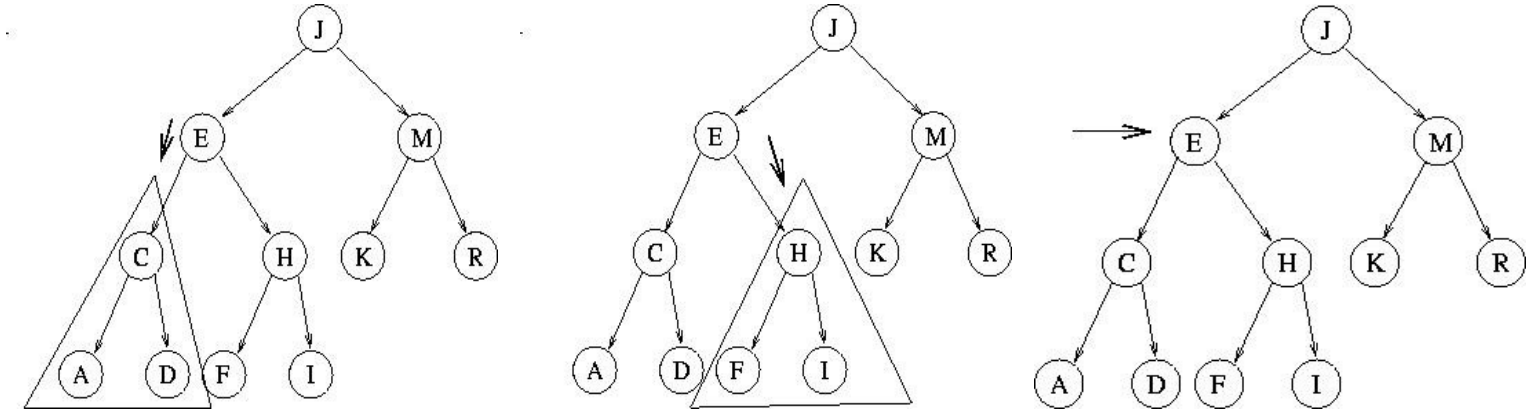
Árvores binárias – Percurso em ordem

- Nesse percurso caminha-se na árvore visitando primeiramente a sub-árvore esquerda do nó, depois se visita o próprio nó, passando-se finalmente para sua sub-árvore direita



Árvores binárias – Percurso pós-ordem

- Nesse percurso caminha-se na árvore visitando primeiramente a sub-árvore esquerda do nó, passando para sua sub-árvore direita, terminando na visita do próprio nó



Árvores binárias - Percurso

- Qualquer das técnicas de percurso demanda o uso de pilhas para o armazenamento temporário dos nós ou sub-árvores ainda não visitados

- Uma forma de evitar o uso dessas pilhas e fazer o uso de **árvores alinhadas**, em que para cada nó que não possua uma de suas sub-árvores gera-se uma ligação com o nó que estaria no topo da pilha de percurso

Árvores binárias - Implementação

- Com matrizes (ou vetor de vetores)
  - Usa-se um vetor para guardar o conteúdo efetivo do nó
  - Usam-se dois outros vetores para armazenar os índices dos nós raízes de cada uma das sub-árvores desse nó
  - Se o nó não possuir sub-árvore, então o vetor correspondente recebe o valor –1
  - O nó raiz da árvore ocupa a primeira posição do vetor ou, opcionalmente, é apontado por uma variável externa que marca o índice da posição ocupada pela raiz

Árvores binárias - Implementação

- Com ponteiros
  - Cada nó da árvore é representado por uma estrutura contendo tanto as informações do nó, quanto os apontadores para as suas sub-árvores
  - Se um nó não possuir uma sub-árvore, o ponteiro respectivo é marcado como nulo
  - A raiz é identificada através de um ponteiro específico

Árvores binárias - Implementação

- Árvores alinhadas
  - A implementação desse tipo de árvore é feita simplesmente com a troca dos indicadores de inexistência de sub-árvore por ponteiros indicando o próximo nó a ser visitado