Breve Revisão

Programa

Um programa pode ser visto como a especificação formal da solução de um problema.

N. Wirth expressa em sua equação

programa = algoritmo + estruturas de dados

Como vimos em AED I, a qualidade da solução de um problema depende, entre outros fatores, da forma como estão organizados os dados relevantes.

UNIVASF

Estruturas de Dados

Estruturas de dados são formas genéricas de se estruturar dados/informação de modo a serem registradas e processadas pelo computador.

Ex.:

- **vetores;**
- **matrizes;**
- **listas (pilhas e filas);**
- **árvores;**
- **grafos, etc.**

Contudo, estas só adquirem significado quando associadas a um conjunto de **operações**, que visam, de um modo geral, manipulá-las (algoritmos).

Listas, Pilhas e Filas

Visando recordá-los dos pontos principais e estabelecer um elo entre as disciplinas, faremos agora um overview dos tópicos vistos na disciplina Algoritmos e Estrutura de Dados I através um conjunto de exercícios.

O que é uma lista?

Uma lista é uma sequência de zero ou mais elementos, cada um deles sendo um valor primitivo (átomo ou nodo) ou composto. Em outras palavras, a estrutura de lista representa a ordem linear entre os elementos. Listas podem ser homogêneas ou heterogêneas, lineares ou não lineares.

Listas, Pilhas e Filas

Cite algumas operações primitivas necessárias a uma implementação de uma lista:

-criar uma nova lista;

-verificar se a lista é vazia;

-acessar o k-ésimo elemento;

-inserir um elemento como k-ésimo da lista;

-remover o k-ésimo elemento;

-etc.

Listas, Pilhas e Filas

No que tange a gerência de memória como uma lista pode ser implementada?

Com relação a gerência de memória uma lista pode ser implementada de forma sequencial ou encadeada.

Utilizando a alocação encadeada uma lista pode ser implementada de algumas formas, cite-as:

-simplesmente encadeada;

- com nó de cabeçalho;

-circular;

-duplamente encadeada;

-etc.

Disciplinas de acesso

Muitas vezes é útil impor, para manipulação de uma certa estrutura de dados, restrições quanto à visibilidade de seus componentes ou quanto à ordem que deve ser respeitada para se efetuarem operações, como inserções ou retiradas, por exemplo. Isto ajuda na modelagem de certos processos que ocorrem no mundo real.

Com o tempo e a prática, foram identificadas algumas disciplinas de acesso aplicadas a estruturas de dados, úteis em diversas aplicações. Dois casos dos mais importantes são casos particulares de listas com disciplinas de acesso, denominados: filas e pilhas.

UNIVASF

Defina uma fila.

Uma fila é uma lista com restrições de acesso, em que as operações de inserção são realizadas sobre uma das extremidades, o fim da lista, enquanto operações de consulta e retirada são feitas na outra extremidade, o início da fila.

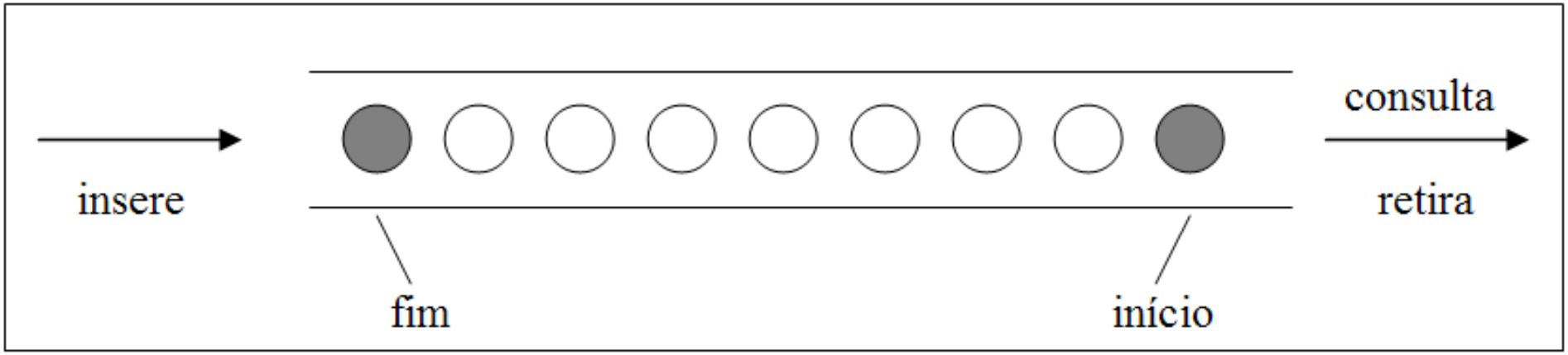
Isto leva ao critério FIFO (first in, first out) que indica que o primeiro item que entra é o primeiro a sair da estrutura. O modelo intuitivo para isto é o de uma fila para atendimento em um guichê, na qual são atendidas as pessoas pela ordem de chegada.

UNIVASF

Listas, Pilhas e Filas

O atendente só tem contato com (só pode consultar) o primeiro (ou o mais antigo) da fila. Novos pretendentes ao serviço entram no fim da fila.

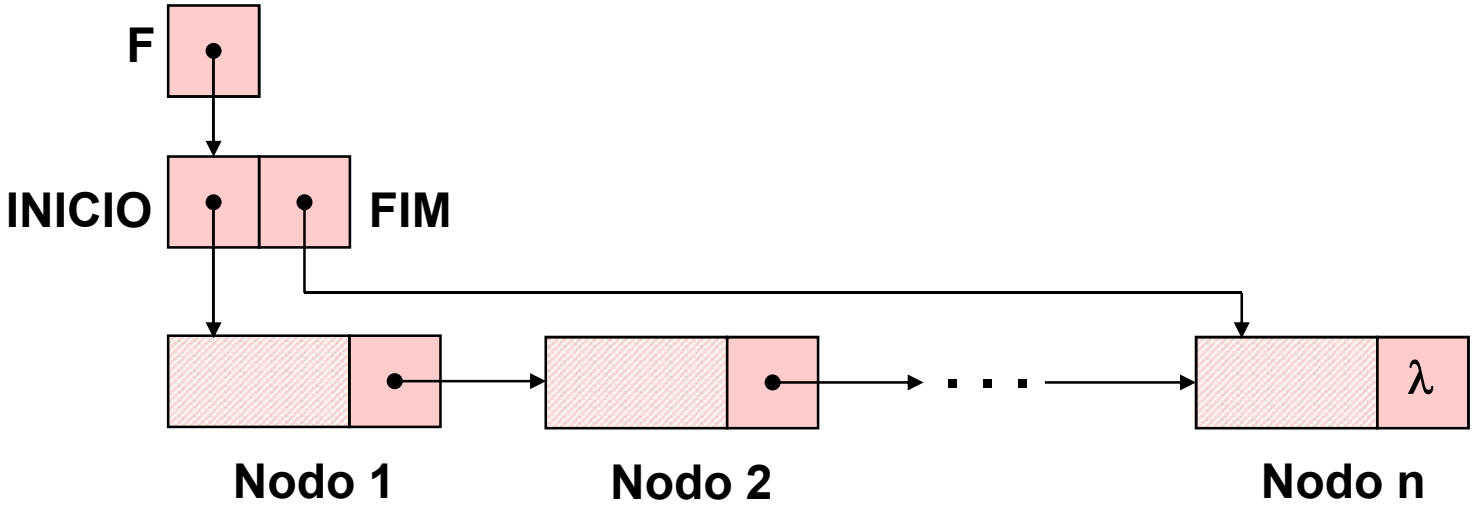
No modelo formal, não há opção de abandono da fila: somente o primeiro pode ser retirado (sair da fila).



Listas, Pilhas e Filas

Especifique uma forma simples e eficiente de armazenar em memória uma fila.

Podemos utilizar uma lista simplesmente encadeada com um nó de cabeçalho (descritor) contendo um ponteiro para o primeiro elemento e outro para o último elemento da fila.



Exemplo, utilizando a linguagem C, da definição de um TAD FILA_ENC (de valores inteiros).

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef struct
{
   NODO *INICIO;
   NODO *FIM;
}DESCRITOR;
typedef DESCRITOR * FILA_ENC;
void cria_fila (FILA_ENC *);
int eh_vazia (FILA_ENC);
void ins (FILA_ENC, int);
int cons (FILA_ENC);
void ret (FILA_ENC);
int cons_ret (FILA_ENC);
```

Defina uma pilha.

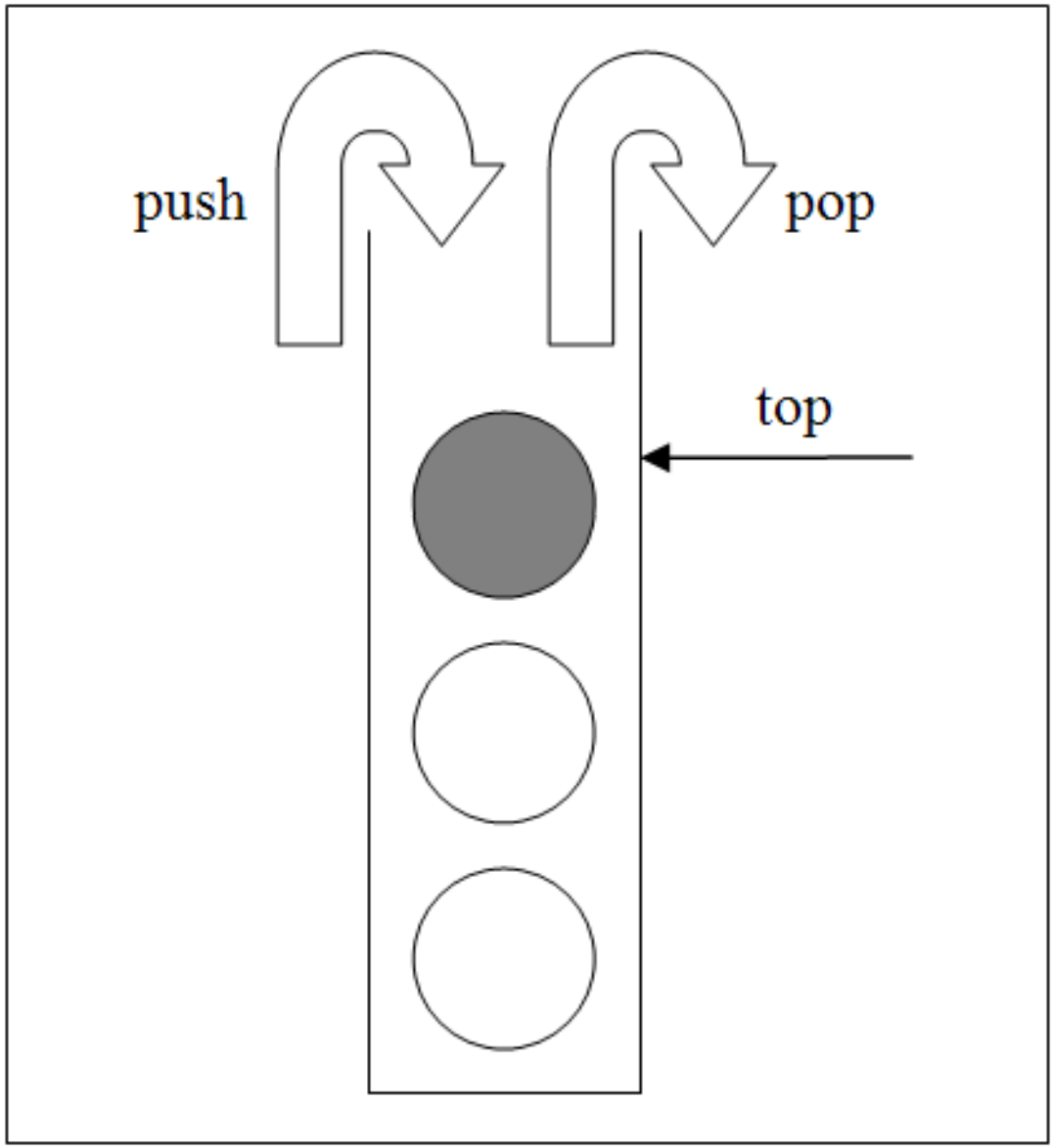
Uma pilha é uma lista com restrições de acesso, onde todas as operações só podem ser aplicadas sobre uma das extremidades da lista, denominada topo da pilha.

Com isso estabelece-se o critério LIFO (last in, first out), que indica que o último item que entra é o primeiro a sair.

O modelo intuitivo para isto é o de uma pilha de pratos, ou livros, etc, na qual só se pode visualizar (consultar) o último empilhado e este é o único que pode ser retirado. E também qualquer novo empilhamento (inserção) se fará sobre o último da pilha.

UNIVASF

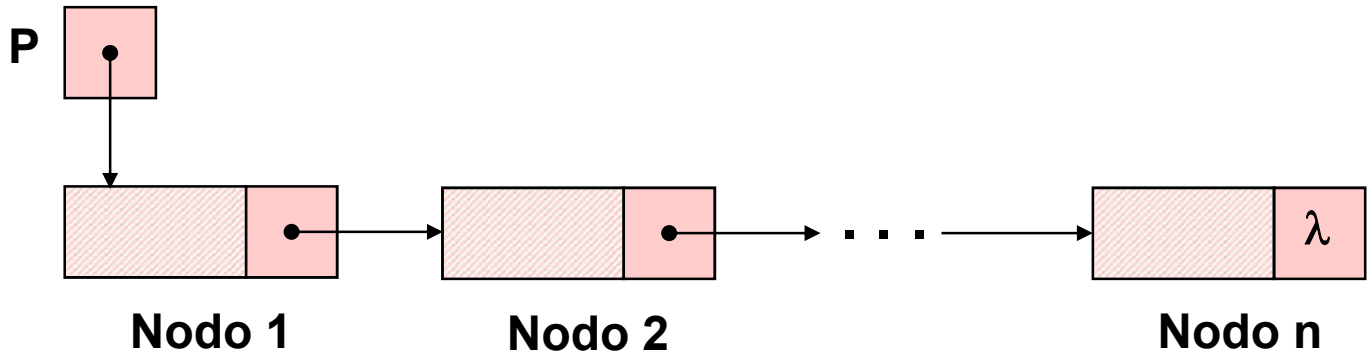
Listas, Pilhas e Filas



Listas, Pilhas e Filas

Especifique uma forma simples e eficiente de armazenar em memória uma fila.

Uma pilha pode ser armazenada em uma lista simplesmente encadeada, sem perda de eficiência.



A implementação das operações é trivial.

Para fazer uma inserção, basta alocar um nodo para o novo valor, ligá-lo ao primeiro nodo da lista e fazer o ponteiro apontar para o novo nodo.

Uma retirada exige apenas que o ponteiro passe a apontar para o segundo nodo da lista (ou ser anulado, se houver apenas um nodo). Uma consulta exige apenas a recuperação do valor do primeiro nodo.

Desta forma, podemos definir o TAD PILHA_ENC (de valores inteiros) da seguinte forma.

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * PILHA_ENC;
void cria_pilha (PILHA_ENC *);
int eh_vazia (PILHA_ENC);
void push (PILHA_ENC *, int);
int top (PILHA_ENC);
void pop (PILHA_ENC *);
int top_pop (PILHA_ENC *);
```

Árvores

Qual a carência das pilhas e filas?

Estas são de difícil utilização para a representação hierárquica de elementos.

Devido a…

Serem limitadas a apenas uma dimensão.

Visando eliminar esta limitação foi criado o conceito de árvore.

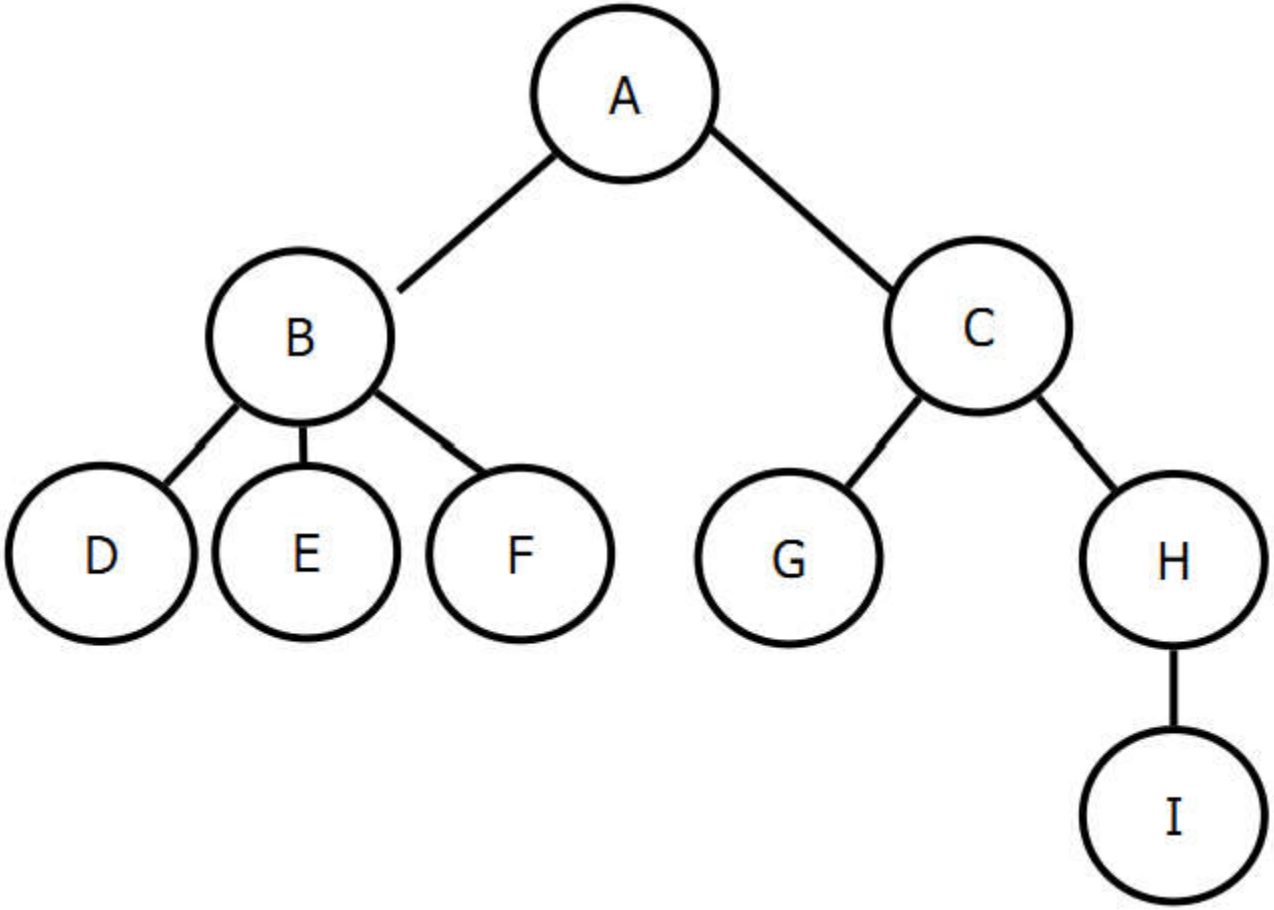
Uma definição recursiva para árvore é:

1. Uma estrutura vazia é uma árvore.

2. Se t_1, …, t_k são árvores disjuntas, então a estrutura cuja raiz tem como suas filhas as raízes de t_1, …, t_k também é uma árvore.

3. Somente estruturas geradas pelas regras 1 e 2 são árvores.

Árvores



Exemplo de árvore

Árvores

O que é uma árvore binária?

É uma árvore cujos nós têm dois filhos (possivelmente vazios) e cada filho é designado como filho à esquerda ou filho à direita.

Em uma árvore binária existem no máximo quantos nós em um determinado nível?

Existem 2^i nós no nível i+1.

Defina uma árvore binária de busca.

Também chamadas de árvore binária ordenada, é uma árvore com a seguinte propriedade: para cada nó n da árvore, todos os valores armazenados em sua subárvore à esquerda (a árvore cuja raiz é o filho à esquerda) são menores que o valor v armazenado em n, e todos os valores armazenados na subárvore à direita são maiores que v .

Árvores

Como é feita a localização de um elemento em uma árvore binária de busca?

Para cada nó, partindo da raiz, compare a chave a ser localizada com o valor armazenado no nó correntemente apontado. Se a chave for menor que o valor, vá para a subárvore à esquerda e tente novamente. Se for maior que o valor, tente a subárvore à direita. Se for o mesmo, a busca poderá parar*. A busca também é abortada se não há meios de continuar, indicando que a chave não está na árvore.

***(isto depende da árvore possuir ou não elementos/nós com o mesmo valor)**

UNIVASF

Árvores

Contudo, o percurso em uma árvore pressupõe visitar cada nó da árvore exatamente uma vez. Este processo pode ser interpretado como...

colocar todos os nós em uma linha ou a linearização da árvore.

Em uma árvore com n nós existem quantos percursos diferentes?

n! percursos diferentes.

Diante de tamanha quantidade de opções, podemos restringir este universo a duas classes de percursos. Quais seriam?

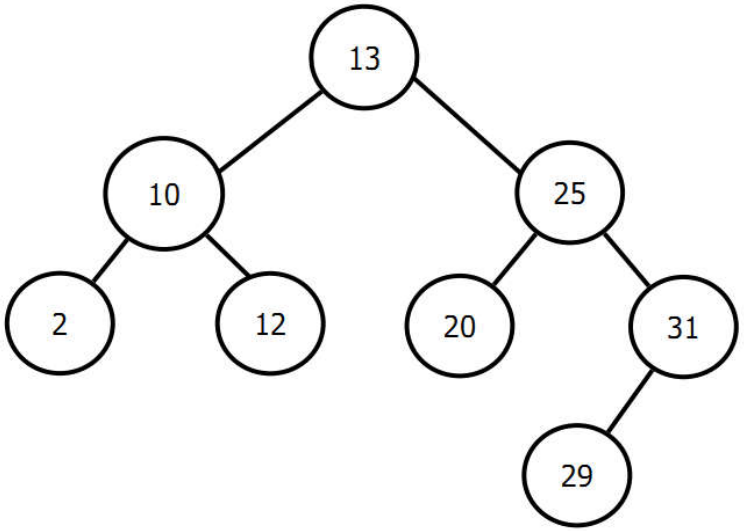
Percurso em largura e em profundidade.

UNIVASF

Árvores

Em que consiste o percurso em largura?

Consiste em visitar cada nó começando no nível mais baixo (nível da raiz) e movendo-se para as folhas nível a nível, visitando todos os nós em cada nível da esquerda para a direita (ou da direita para a esquerda).



Sequência de nós visitados:
13, 10, 25, 2, 12, 20, 31, 29

Árvores

O percurso em profundidade consiste em quais operações básicas?

V – Visitar o nó;

L – percorrer a subárvore esquerda;

R – percorrer a subárvore direita.

Existem 3! modos de organizar tais operação, porém este conjunto é reduzido a três possibilidades que seriam:

VLR – cruzamento de árvore em pré-ordem (as vezes denominado em profundidade)

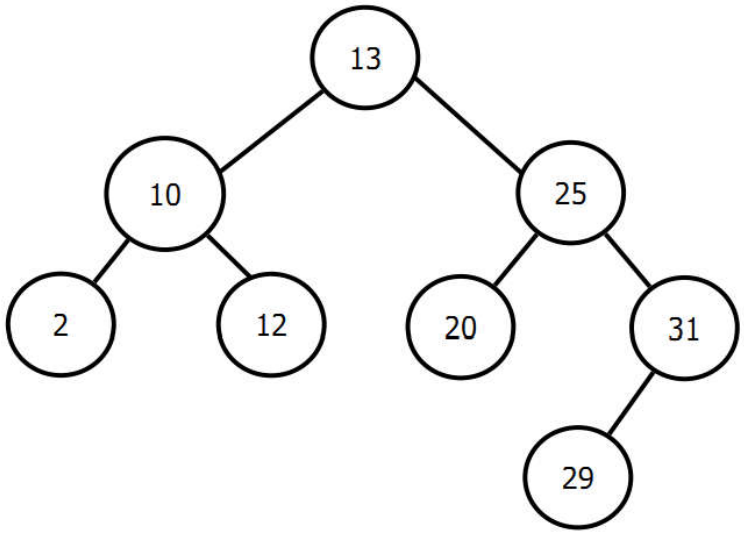
LVR – cruzamento de árvore em in-ordem (conhecido também como ordem simétrica)

LRV – cruzamento de árvore em pós-ordem

UNIVASF

Árvores

Como ficaria o percurso da árvore abaixo em pré-ordem, in-ordem e pós-ordem?



Pré-ordem
13, 10, 2, 12, 25, 20, 31, 29

In-ordem
2, 10, 12, 13, 20, 25, 29, 31

Pós-ordem
2, 12, 10, 20, 29, 31, 25, 13

Estes percursos podem ser implementados com recursividade (pilha implícita), com uma pilha explicita implementada pelo programador ou utilizando o algoritmo idealizado por Joseph M. Morris.

Árvores

A inserção de um nó em uma árvore binária de busca não representa um grande desafio.

A remoção de um nó em uma árvore binária de busca já não é tão trivial. A que é proporcional a complexidade do algoritmo de remoção?

Ao número de filhos que o nó a ser removido possui. Cite as possibilidades.

Nó sem filho – o ponteiro apropriado de seu ascendente é ajustado para nulo e a memória ocupada pelo nó é liberada.

Nó com um filho – o ponteiro apropriado de seu ascendente é ajustado para apontar para o filho do nó e a memória ocupada pelo nó é liberada.

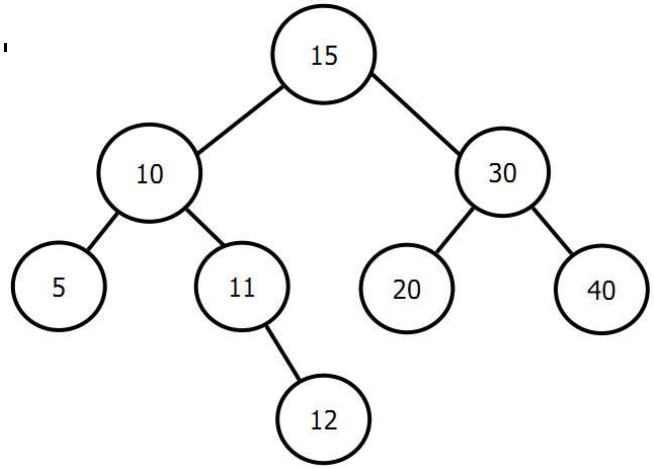
UNIVASF

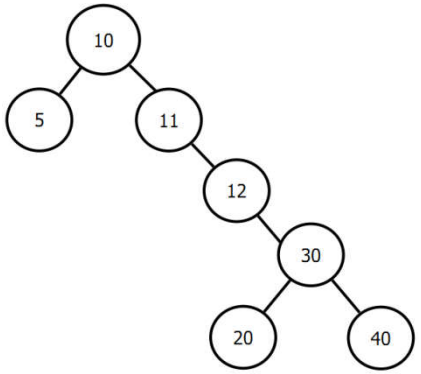
Árvores

Nó com dois filhos – para este caso nenhuma operação de apenas uma etapa pode ser executada, foram trabalhadas duas soluções para este caso.

Remoção por fusão e remoção por cópia.

Na remoção por fusão, uma das duas subárvores do nó é extraida e anexada à outra subárvore. Com base no que foi dito, analise a árvore abaixo e sugira como a remoção do nó com valor 15 se dará.





Árvore binária de busca

Outra solução é a Remoção por cópia.

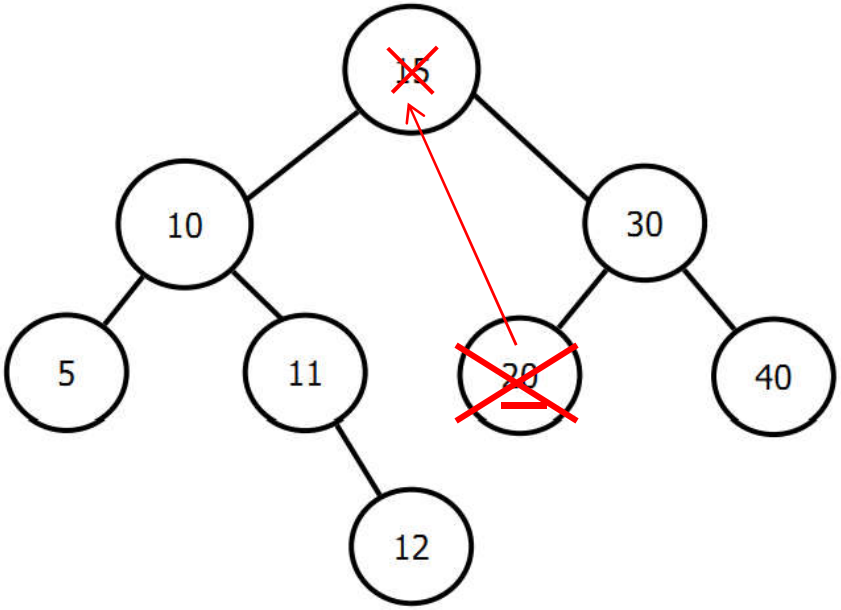
A qual propõe que um nó com dois filhos a ser removido pode ser reduzido a uma das duas situações básicas: nó com apenas um filho e nó sem nenhum filho.

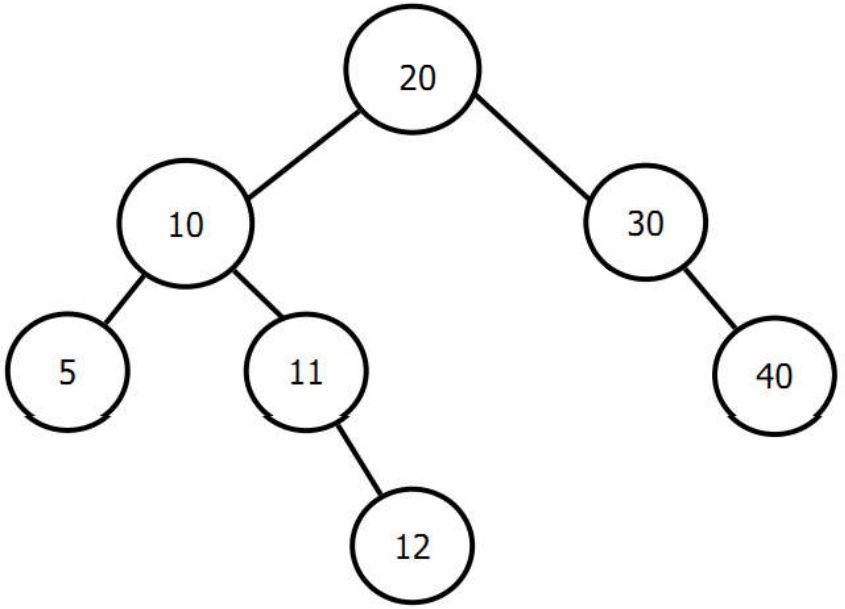
Isso é feito substituindo pela chave de seu sucessor imediato a chave que está sendo removida e em seguida removendo o nó que continha a chave do sucessor imediato.

Obs.: o sucessor imediato de um nó é o nó mais à esquerda em sua subárvore à direita.

Árvore binária de busca

Vejamos um exemplo:





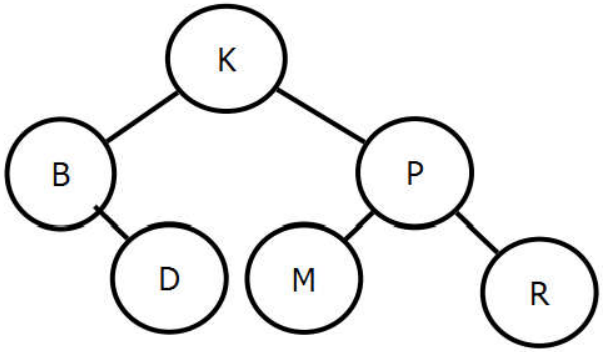
Árvore Balanceada

Além das árvores serem muito apropriadas para representar a estrutura hierárquica de um certo domínio, o processo de busca por um elemento em uma árvore tende a ser muito mais rápido do que em uma lista encadeada.

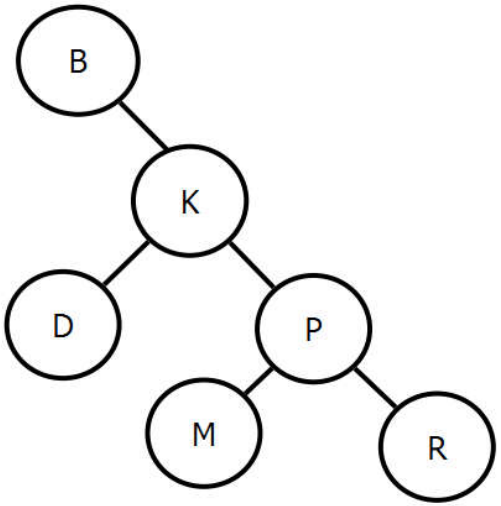
Contudo, para que efetivamente uma busca por um determinado elemento em uma árvore seja eficiente está, além de ser uma árvore binária de busca, deve ter seus nós adequadamente distribuídos.

Árvore Balanceada

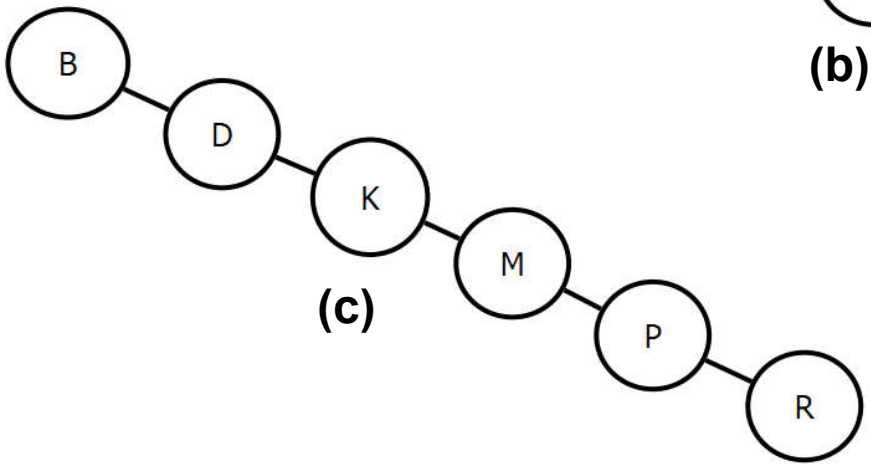
Observe as árvores a seguir:



(a)



(b)



(c)

Árvores

Com uma análise das árvores apresentadas podemos concluir que uma grande gama de configurações de árvores binárias de busca é passível de ser obtida com um mesmo conjunto de nós e que o pior caso de uma busca por um nó está diretamente relacionado com a altura da árvore.

Sendo assim, podemos estabelecer a seguinte definição: Uma árvore binária é ***balanceada em altura*** ou simplesmente ***balanceada*** se a diferença na altura de ambas as subárvores de qualquer nó na árvore é zero ou um.

Árvores

Uma definição complementar é a de árvore ***perfeitamente balanceada***.

Uma árvore binária é ***perfeitamente balanceada*** quando,além de ser balanceada, todas as suas folhas encontram-se em um ou dois níveis.

Para facilitar a visualização, uma árvore que possui 10.000 nós pode ser configurada em uma árvore com altura igual a $lg(10000) = ceil(13.289) = 14$. Ou seja, qualquer elemento é passível de ser localizado com no máximo 14 comparações se a árvore for perfeitamente balanceada.

UNIVASF

Árvores

Neste ponto cabe a seguinte pergunta: Ao se analisar uma árvore pode-se determinar qual dentre os seus nós seria uma raiz adequada para torná-la perfeitamente balanceada, qual seria este nó?

O nó cuja sua chave (valor) representa a mediana das chaves presentes nos nós que compõem a árvore, para uma árvore com número impar de nós. Ou o nó cuja sua chave (valor) representa um dentre os dois valores mais próximos da mediana das chaves presentes nos nós que compõem a árvore, para uma árvore com número par de nós.

Vimos que uma estratégia baseada nesta observação que pode ser utilizada para balancear uma árvore é:

Árvores

- Quando os dados chegarem armazene-os em um vetor;

- Após todos os dados terem sido armazenados no vetor, ordene-o;

- Agora determine como raiz o elemento do meio do vetor;

- O vetor consistirá agora em dois subvetores. O filho esquerdo da raiz será o nó com valor no meio do subvetor constituído do início do vetor até o elemento escolhido como raiz;

- Um procedimento similar é adotado para a definição do filho direito da raiz;

- Este processo se repete até não existirem mais elementos a serem retirados do vetor.

Árvores

O algoritmo apresentado possui um sério inconveniente, pois todos os dados precisam ser colocados em um vetor antes da árvore ser criada. O mesmo algoritmo também pode ser utilizado quando uma inserção ocorre sobre uma árvore já existente. Porém, com o mesmo inconveniente. Neste caso, os elementos da árvore devem ser retirados e colocado em um vetor, mesmo que a árvore binária de busca esteja desbalanceada, se for efetuado um percurso in-ordem elimina-se a necessidade de ordenar o vetor. Posteriormente o novo elemento é inserido em sua posição adequada no vetor e a função definida anteriormente pode ser utilizada.

Árvores

Vimos que existem formas mais eficientes de se balancear uma árvore.

Um exemplo é o algoritmo desenvolvido por Colin Day e posteriormente melhorado por Quentin F. Stout e Bette L. Warren. Denominado Algoritmo DSW, devido aos nomes de seus idealizadores, baseia-se em percorrer uma árvore binária de busca tornando-a uma arvore degenerada (similar a uma lista encadeada) e posteriormente percorrê-la novamente tornando-a uma árvore perfeitamente balanceada.

UNIVASF

Árvores

Seguiremos recordando conceitos de árvore AVL e árvore 234.

Os slides que versão sobre as árvores retro aludidas foram baseados nos slides gerados pela professora Elisa Maria Pivetta Cantarelli intitulados Árvores Binárias Balanceadas.

Vimos algoritmos que balanceiam árvores globalmente. Contudo, o rebalanceamento pode ocorrer localmente se alguma porção da árvore for desbalanceada por uma operação de inserção ou remoção de um elemento da árvore. Um método clássico para tal foi proposto por Adel'son-Vel'skii e Landis e o nome dado a uma árvore modificada com este método é árvore AVL.

UNIVASF

Árvores

Uma árvore AVL é uma árvore binária de busca onde a diferença em altura entre as subárvores esquerda e direita de cada nó é no máximo um (positivo ou negativo).

Esta diferença é chamada de fator de balanceamento (FB).

O FB é acrescentado a cada nó da árvore AVL.

O FB é obtido através da seguinte equação:

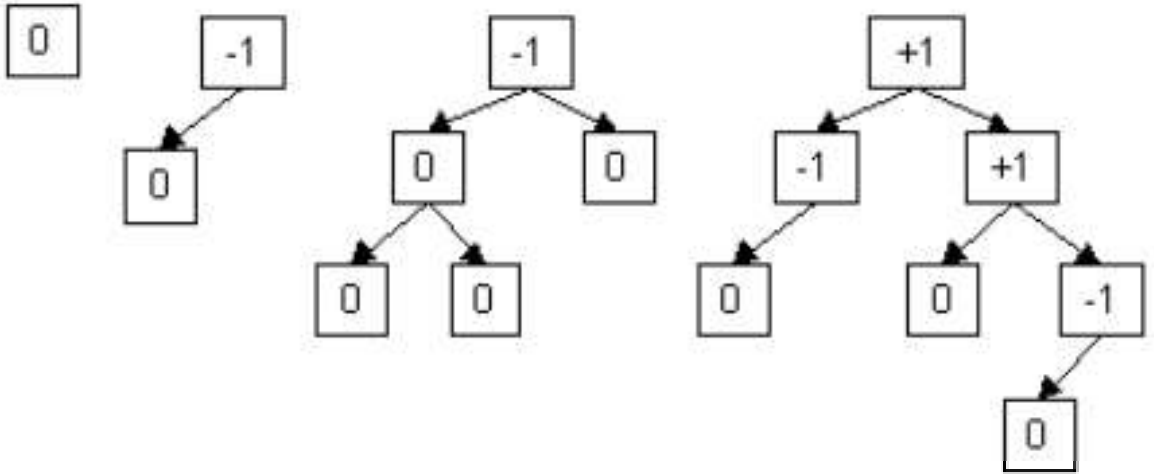
FB(nó p) = altura(subárvore direita de p) - altura(subárvore esquerda de p)

Relembraremos alguns exemplos de árvores AVL e árvores não AVL.

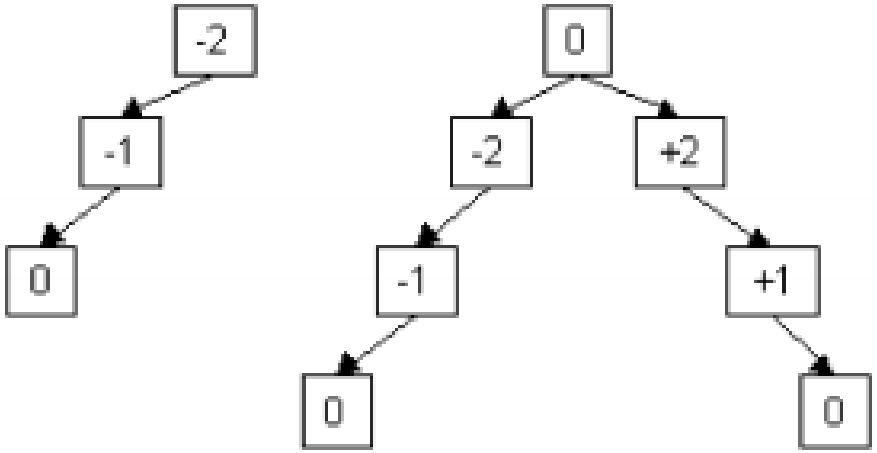
UNIVASF

Árvores

Árvores AVL:



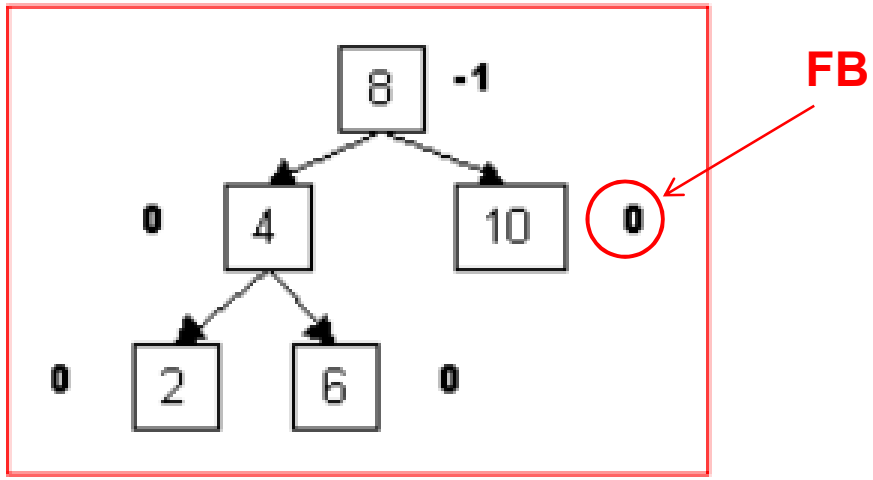
Árvores não AVL:



Árvores

Ao inserirmos um novo nó em uma árvore AVL, podemos ou não violar a propriedade de balanceamento.

Caso ocorra uma violação devemos rebalancear a árvore através da execução de operações de rotação sobre nós da árvore.



Árvore AVL

Árvores

Após uma inserção que gere um desbalanceamento na árvore AVL podemos nos deparar com duas classes de desbalanceamento.

Onde estes são identificados com base na análise dos FB's.

Ao inserirmos um novo nó devemos ajustar os FB's do nó inserido até a raiz ou até encontrarmos um fator de balanceamento inaceitável, ou seja, com valor 2 ou -2.

Quando o FB do nó filho com valor 1 possuir o mesmo sinal do FB de seu pai (nó com FB 2 ou -2) trata-se da classe de desbalanceamento 1, que requer apenas uma rotação simples para a árvore ser rebalanceada.

Árvores

Quando o FB do nó filho com valor 1 possuir sinal oposto ao FB de seu pai (nó com FB 2 ou -2) trata-se da classe de desbalanceamento 2 que requer uma rotação dupla, ou em outras palavras, duas rotações para a árvore ser rebalanceada.

Se o sinal do FB do nó que caracteriza o desbalanceamento for positivo a rotação será para a esquerda.

Se o sinal do FB do nó que caracteriza o desbalanceamento for negativo a rotação será para a direita.

UNIVASF

Árvores

Descreva uma sequência de passos para a construção de uma árvore AVL.

1. insira o novo nó normalmente, ou seja, da mesma maneira que insere-se um nó em uma árvore de busca balanceada;

2. iniciando com o nó pai do nó recém inserido, teste se a propriedade AVL foi violada, ou seja, atualize e teste se algum dos FB's passou a ser 2 ou -2. Temos duas possibilidades:

2.1. A condição AVL foi violada

2.1.1. Execute a operação de rotação conforme o caso (tipo 1 ou tipo 2);

2.1.2. Volte ao passo 1;

2.2. A condição AVL não foi violada, volte ao passo 1;