**E-mail: profa.jaqueline@gmail.com**

**Aula 15/02:**

- Métodos de ordenação;

- Tipo abstrato de dados:

Exemplo: Pilha; Fila 🡪 conceito atrelado a um processo;

- Hash 🡪 Método para criptografia

- Lista encadeada 🡪 Não possui tamanho fixo 🡪 Conceito semelhante de vetor para JavaScript;

- Lista dinâmica;

- Árvore 🡪 Estrutura não-linear 🡪 Deu origem ao banco de dados;

- Recursividade 🡪 Uma função chamando ela mesma;

**Aula 20/02:**

- Métodos de ordenação:

- Seleção; Inserção e troca 🡪 Selection sort; Bubble sort e Quick sort; Insert sort;

Bubblesort 🡪 Método mais simples de ordenação 🡪 não é muito eficiente

25 57 48 37 12 92 86 33

25 57 48 37 12 92 86 33

25 48 57 37 12 92 86 33

25 48 37 57 12 92 86 33

25 48 37 12 57 92 86 33

25 48 37 12 57 92 86 33

25 48 37 12 57 86 92 33

25 48 37 12 57 86 33 92 🡪 primeira iteração

25 48 37 12 57 86 33 92

25 48 37 12 57 86 33 92

25 37 48 12 57 86 33 92

25 37 12 48 57 86 33 92

25 37 12 48 57 86 33 92

25 37 12 48 57 86 33 92

25 37 12 48 57 33 86 92 🡪 Segunda iteração

25 37 12 48 57 33 86 92

25 37 12 48 57 33 86 92

25 12 37 48 57 33 86 92

25 12 37 48 57 33 86 92

25 12 37 48 57 33 86 92

25 12 37 48 33 57 86 92 🡪 Terceira iteração

25 12 37 48 33 57 86 92

12 25 37 48 33 57 86 92

12 25 37 48 33 57 86 92

12 25 37 48 33 57 86 92

12 25 37 33 48 57 86 92 🡪 Quarta iteração

12 25 37 33 48 57 86 92

12 25 37 33 48 57 86 92

12 25 37 33 48 57 86 92

12 25 33 37 48 57 86 92 🡪 Quinta iteração

12 25 33 37 48 57 86 92

12 25 33 37 48 57 86 92

12 25 33 37 48 57 86 92 🡪 Sexta iteração

12 25 33 37 48 57 86 92

12 25 33 37 48 57 86 92 🡪 Sétima iteração 🡪 Fim do código;

Selection sort 🡪 Busca pelo menor e troca com o primeiro da fila

10 20 12 5 8 15

5 20 12 10 8 15

5 8 12 10 20 15

5 8 10 12 20 15

5 8 10 12 20 15

5 8 10 12 15 20

25 57 48 37 12 92 86 33

12 57 48 37 25 92 86 33

12 25 48 37 57 92 86 33

12 25 33 37 57 92 86 48

12 25 33 37 57 92 86 48

12 25 33 37 48 92 86 57

12 25 33 37 48 57 86 92

12 25 33 37 48 57 86 92

Insertion sort 🡪 Insere o valor na sua posição certa;

25 57 48 37 12 92 86 33

25 57 48 37 12 92 86 33

25 48 57 37 12 92 86 33

25 37 48 57 12 92 86 33

12 25 37 48 57 92 86 33

12 25 37 48 57 92 86 33

12 25 37 48 57 86 92 33

12 25 33 37 48 57 86 92 🡪 Vetor final.

**Aula 12/03**

**Quicksort:**

Dividir o problema e ordenar, escolher um pivô, a esquerda do pivô, ficam todos os menores que ele, a direita ficam todos os maiores;

25 57 48 37 12 92 86 33 🡪 pivo

i j

25 57 48 37 12 92 86 33 🡪 troca

I j

25 33 48 37 12 92 86 57 🡪 i para

I j

25 33 48 37 12 92 86 57

I j

25 33 48 37 12 92 86 57 🡪 j para 🡪 troca

I j

25 33 12 37 48 92 86 57 🡪 Divide o vetor em 2 na recursão

25 33 12 🡪 pivo

48 92 86 57 🡪 pivô

25 33 12 🡪 j para

i j

25 33 12 🡪 troca

i j

25 12 33 🡪 divide vetor

25 12 🡪 pivô

I j

25 12 🡪 para j e troca;

12 25

Segundo vetor:

48 92 86 57 🡪 j para

I j

48 92 86 57

I j

48 57 86 92 🡪 divide o vetor de novo

48 57 86 🡪 pivô

48 57 86

I j

48 57 86 🡪 para

I j

Vetor final:

12 25 33 37 48 57 86 92 🡪 ordenado

**Aula 26/03:**

HeapSort 🡪 define o heap como sendo metade do tamanho do vetor;

ShellSort 🡪 define um gap e compara pulando esse valor pelo vetor.

Big O 🡪 calculo da ordem de complexidade de um algoritmo (2^n, n^3, n^2, nlog2n, n, log2n)

**Aula 09/04:**

Tipos abstratos de dados 🡪 Tem implícito todo funcionamento da estrutura de dados;

Fila 🡪 Inserção de elementos, (FIFO) como numa fila de supermercado;

Comandos de fila:  
- Situação: vazia (empty), cheia (full);

- Operações básicas: inserir (append ou insert) e remover (serve ou delete);

- Outras operações: limpar (clear), tamanho (size), começo (front), final (rear);

- Implementar a função clear utilizando apenas serve e empty:

function clear(){

    while(fila != empty){

        fila.serve()

    }

}

Pilha 🡪 como uma coluna, onde se insere no topo e se retira do topo 🡪 exemplo: pilha de pratos;

- Todas as inserções e remoções são feitas em uma única extremidade (topo);

- LIFO – Last In First Out;

Comandos de fila:

- Situação: vazia (empty) e cheia (full);

- Operações básicas: inserir (push) e remover (pop);

- Outras operações: limpar (clear), tamanho (size) e topo (top);

- Implementar a função clear e top utilizando push, pop e empty:

function clearPilha(){

    while(pilha != empty){

        pilha.pop()

    }

}

function top(){

    if(pilha != empty){

        x = pop()

        pilha.push(x)

    }

}

**Aula 02/05:**

Hash 🡪 função utilizada para mapear registros;

- Muito utilizado para criptografia;

Duas etapas para criação de um método de pesquisa de transformação de chave:

1 – Computar a função de transformação (hashing), a qual transforma a chave de pesquisa em um endereço da tabela;

2 – Considerar que 2 ou mais chaves poderem levar ao mesmo endereço, deve-se ter uma função de colisão;

Paradoxo do aniversario 🡪 chance de duas pessoas em uma sala fazerem aniversario no mesmo dia;

Escolha do número para função: b^i +-j; onde i e j são números inteiros pequenos e b corresponde ao numero de bytes de algum conjunto,

Hash de endereçamento aberto 🡪 Hash Linear 🡪 tudo armazenado dentro do vetor hash;

Aula 09/05:

Árvores 🡪 estrutura de dados não linear, normalmente com definição por meio de recursão;

Definição:

Nó especial chamado “raiz”;

Cada sub nó forma uma sub-arvore;

Nós sem filhos são chamados de “folhas” ou nó terminal;

Nós com filhos são chamados de “não-folha” ou não-terminal;

Grau 🡪 definido pelo nó com mais filhos;

Nivel 🡪 distancia da raiz até as folhas;

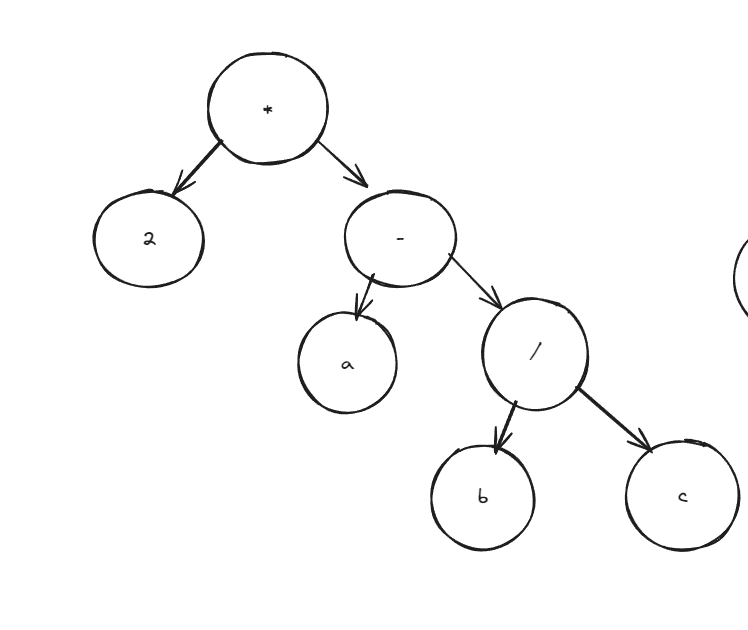
Profundidade 🡪 distancia da raiz até a folha mais abaixo;

Árvore ordenada 🡪 os ramos de cada nó são ordenados, ou seja, duas arvores iguais em nível, grau e tudo mais, são diferentes se estiverem construídas de forma diferente;

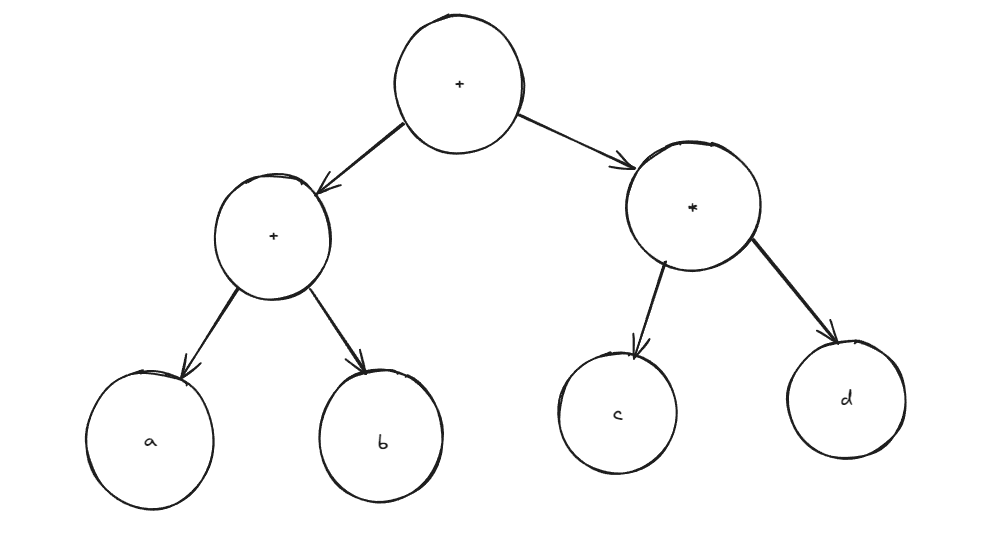
Árvore binaria 🡪 arvore ordenada de grau 2;

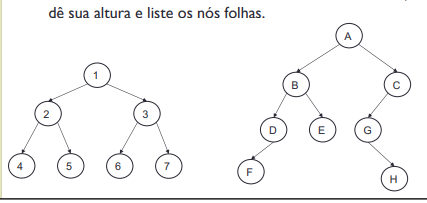
Exercicios arvore binaria:  
a) 2 \*(a –b/c)

b) a + b + c \* d

a) 

b)





Pre-ordem 🡪 1-2-4-5-3-6-7

In-ordem 🡪 4-2-5-1-6-3-7

Pos-ordem 🡪 4-5-2-6-7-3-1

Perfeitamente balanceada, altura = 2, folhas 🡪 4,5,6,7

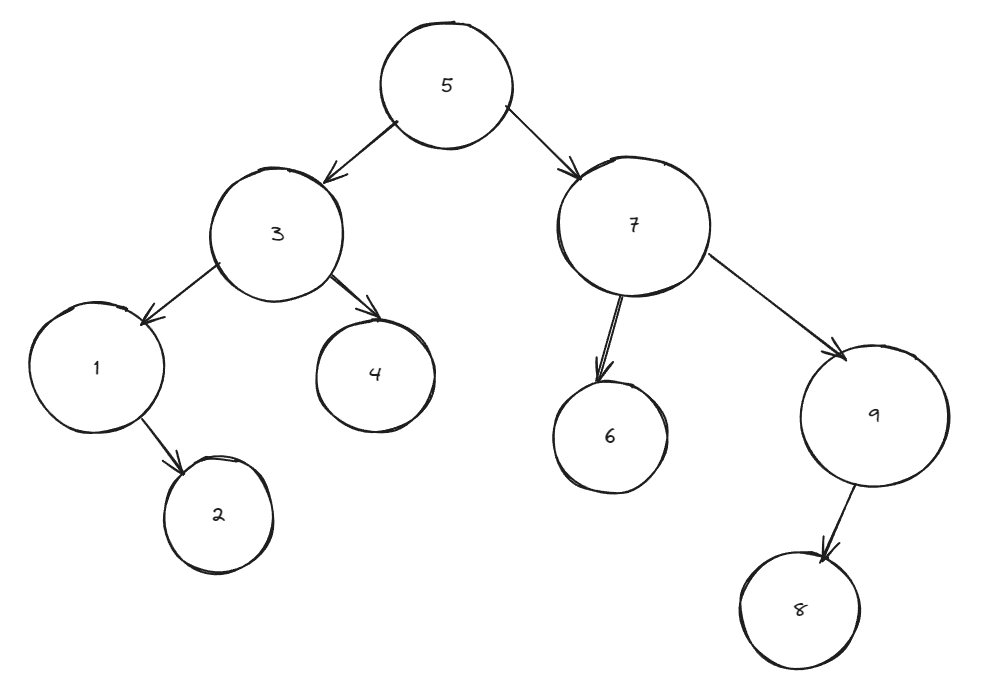
Pre-ordem 🡪 A-B-D-F-E-C-G-H

In-ordem 🡪 F-D-B-E-A-G-H-C

Pos-ordem 🡪 F-D-E-B-H-G-C-A

Não é balanceada, altura = 3; folhas 🡪 F-E-H

Binaria de busca 🡪 5-7-3-1-9-4-6-2-8



Árvore AVL 🡪 Garante que sempre estará balanceada;

Eficiência de busca 🡪 O(log2(n)) mesmo no pior caso;

Por definição, um arvore vazia é uma arvore AVL, além disso, em uma arvore AVL, suas sub arvores são sempre arvores AVL;

Altura da esquerda – altura da direita = fator de balanceamento;

Balanceamento baseado no fator de balanceamento, se os nós estiverem com o mesmo sinal, faz-se um rotação simples para algum dos lados, caso tiverem sinais trocados, faz-se uma rotação dupla, no qual a folha vira raiz;

Exemplo:

