Algoritmos e Estrutura de Dados – AC5

Professor Osvaldo Kotaro Takai

- Geovane Donizete Laera.....RA: 1902679
- Isaque Ribeiro dos Santos Junior.....RA: 1903978
- Marcelo Martinez Mesa Campos.....RA: 1905076
- Paulo Ricardo Costa da Silva.....RA: 1905013
- Vinícius da Cruz Pera.....RA: 1903144

TAD Fila de Prioridade

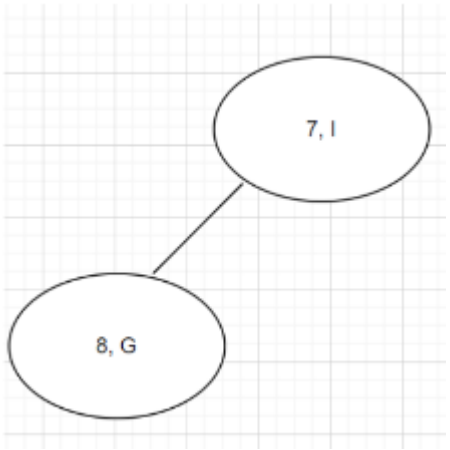
1. Implemente e teste:
    a. TAD Fila de Prioridade baseado em lista (slides de 21 a 31)
    R: // ---------- Implementação Java

    b. TAD Árvore Binária Completa (Slides de 40 a 47)
    R: // ---------- Implementação Java

    c. TAD Fila de Prioridade usando Heap (slides de 75 a 79)
    R: // ---------- Implementação Java

    d. Um método que ordene (9, 1, 3, 6, 2, 7, 8) usando TAD Fila de Prioridade usando Heap.
    R: // ---------- Implementação Java

2. Qual é a saída (desenho do heap) da seguinte sequência de métodos do TAD fila de prioridade: insert(5, A), insert(4, B), insert(7, I), insert(1, D), removeMin(), insert(3, J), insert(6, L), removeMin(), removeMin(), insert(8, G), removeMin(), insert(2, H), removeMin(), removeMin()?
    - Monte a fila de prioridade na mão e compare rodando um teste.



3. Um aeroporto quer simular o tráfego aéreo com eventos como decolagens e pousos. Os eventos têm um time-stamp com a hora em que o evento acontece. O simulador deve realizar eficientemente as duas operações fundamentais a seguir:
   - Inserir um evento com um dado time-stamp (ou seja, inserir um evento futuro);
   - Extrair o evento com menor time-stamp (ou seja, determinar o próximo evento a processar);

   Que estrutura de dados você usaria para suportar essas operações? Justifique sua resposta

   R: Fila de prioridade, porque essa estrutura de dados vai organizar crescentemente os elementos de acordo com o horário previsto, dando prioridade aos horários mais próximos do atual, e como requerido, extrair o horário do voo mais próximo de acontecer, tal como eliminar o mesmo da fila, após já haver sido realizado etc.

4. Onde pode estar armazenado o elemento com a maior chave em um heap?

   R:
   Em um heap-min, estará em algum dos nós externos (folhas).
   Em um heap-max, estará na raiz da árvore.

5. Seja T uma árvore binária completa em que **v** armazena a entrada (**p(v)**,0), onde **p(v)** é o número do nível de **v**. A árvore **T** é um heap? Justifique sua resposta.

   R: Sim, pois o formato descrito é exatamente como é em um heap: há listas dentro de uma lista, contendo um conjunto chave + valor, junto do índice do nodo, que equivale ao seu nível na árvore.

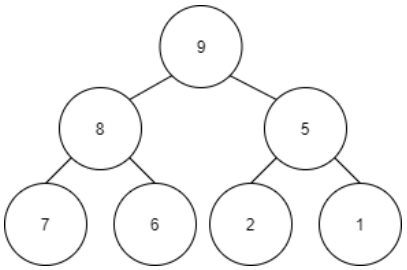
6. Explique por que não se considera o caso do filho direito de **r** ser interno e o filho esquerdo ser externo quando se descreve o processo do down-heap bubbling.

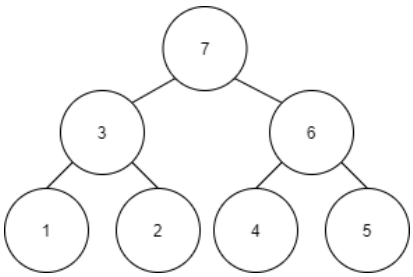
   R: Porque o filho da direita não existirá se não houver o filho da esquerda, já que o da esquerda sempre será o primogênito.

7. Existe um heap **T** armazenando sete elementos diferentes de forma que um caminhamento prefixado de **T** apresente os elementos de **T** em ordem crescente ou decrescente? E se for um caminhamento interfixado? E pós - fixado? Se sim, apresente um exemplo; caso contrário, justifique

R: Em um min-heap não, pois os caminhamentos não seguem de acordo com a ordem em que os elementos são dispostos no heap, de acordo com suas chaves.

Apenas seria possível ambos crescente e decrescente se dispostos em um max-heap, conforme exemplos:





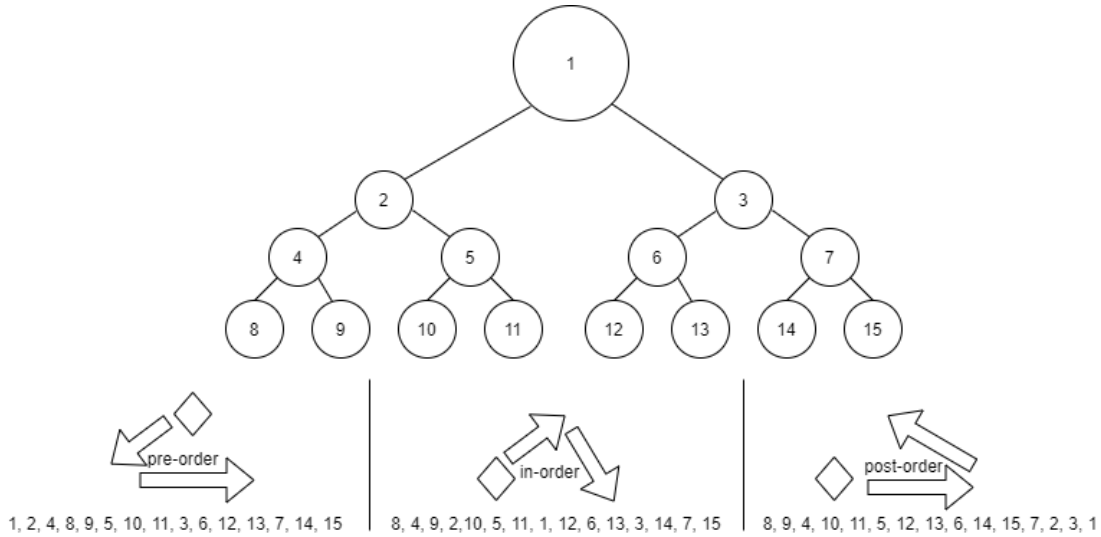
pré-order:

9, 8, 7, 6, 5, 2, 1

pós-order:

1, 2, 3, 4, 5, 6, 7.

8. Considere **H** um heap que armazena 15 elementos usando uma representação de arranjo de uma árvore binária completa. Qual é a sequência de índices da lista de arranjo que são visitados no caminhamento prefixado de H? E qual é a sequência em um caminhamento interfixado? E em um caminhamento pós-fixado?



pré-order:

1, 2, 4, 8, 9, 5, 10, 11, 3, 6, 12, 13, 7, 14, 15.

in-order:

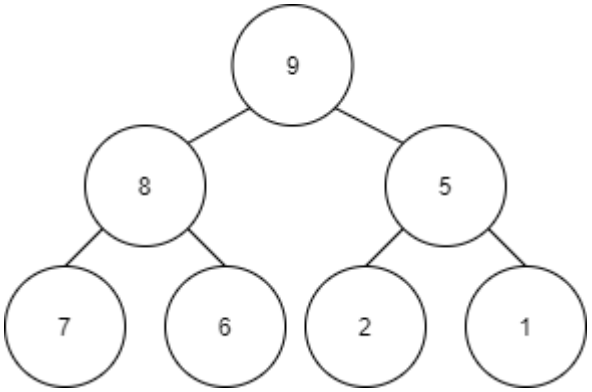
8, 4, 9, 2,10, 5, 11, 1, 12, 6, 13, 3, 14, 7, 15.

post-order:

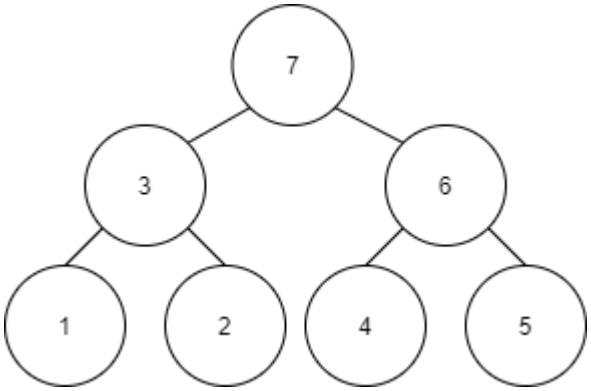
8, 9, 4, 10, 11, 5, 12, 13, 6, 14, 15, 7, 2, 3, 1.

9. Bill afirma que um caminhamento prefixado em um heap listará as chaves em ordem não-decrescente. Apresente um exemplo de um heap que prove que ele está errado.

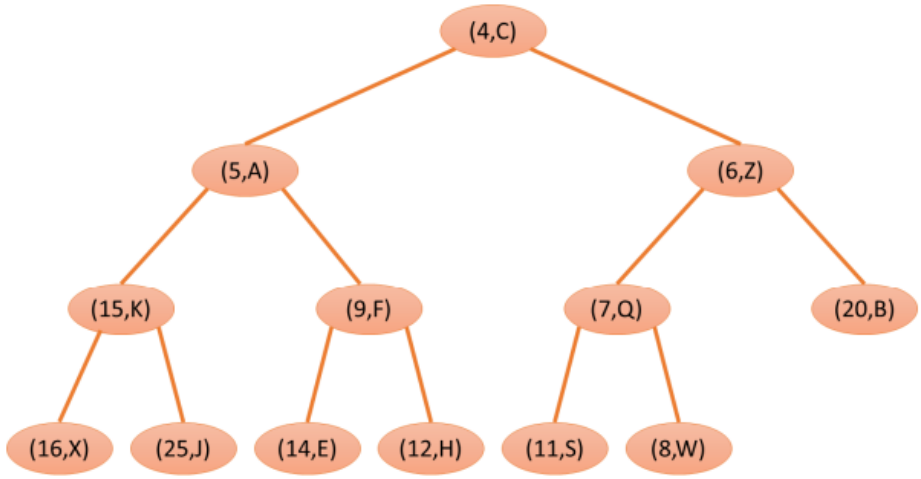
   R: O único caso em que isso é plausível, é em um max-heap:
   9, 8, 7, 6, 5, 2, 1.

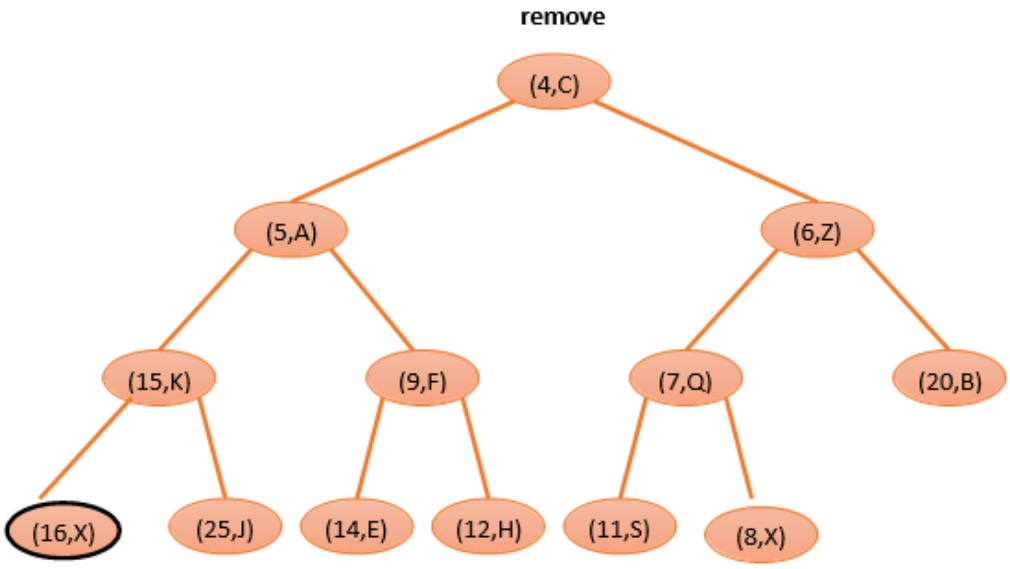


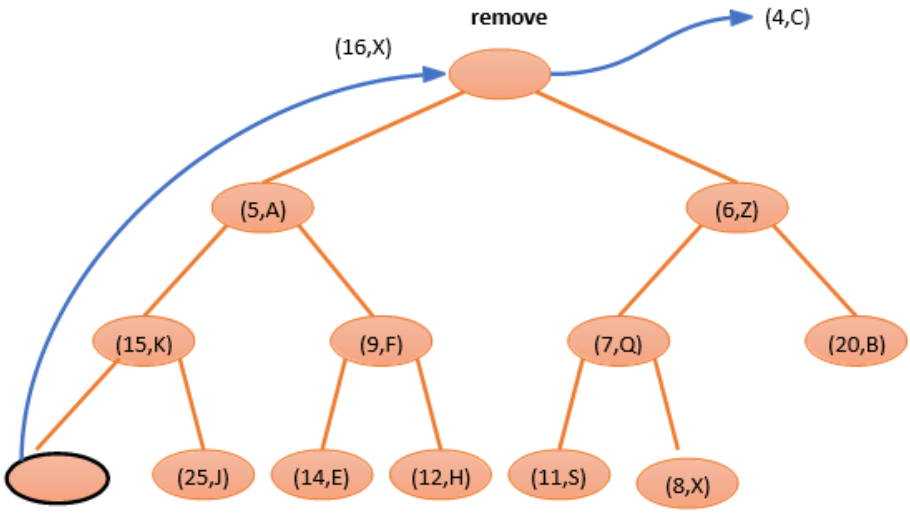
10. Hillary afirma que um caminhamento pós-fixado em um heap listará as chaves em ordem não-crescente. Apresente um exemplo de um heap que prove que ela está errada.
    R: O único caso em que isso é plausível, é em um max-heap:
    1, 2, 3, 4, 5, 6, 7.

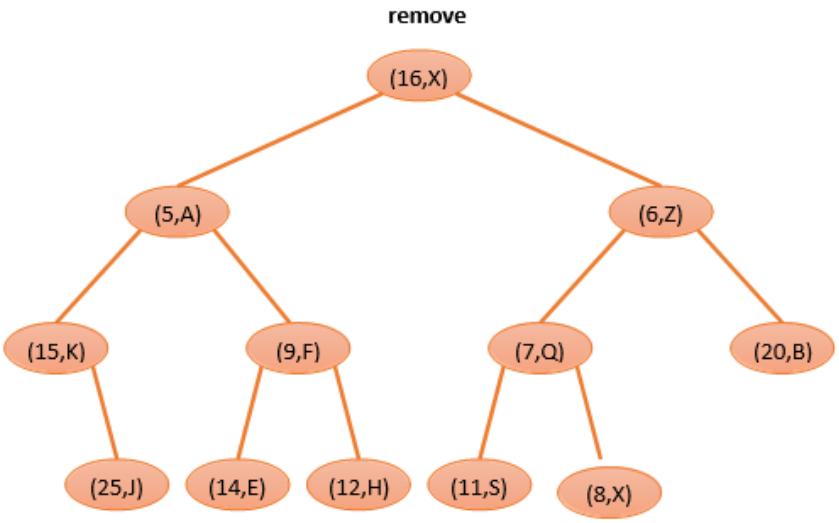


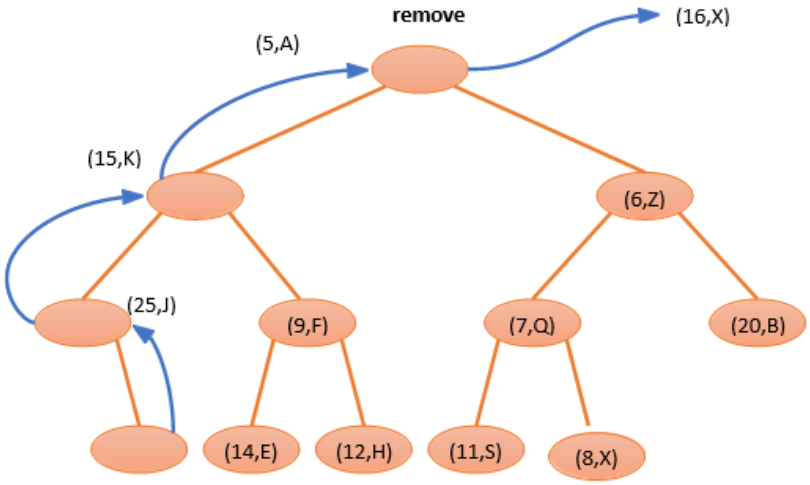
11. Apresente todos os passos do algoritmo para remover a chave 16 do heap abaixo:

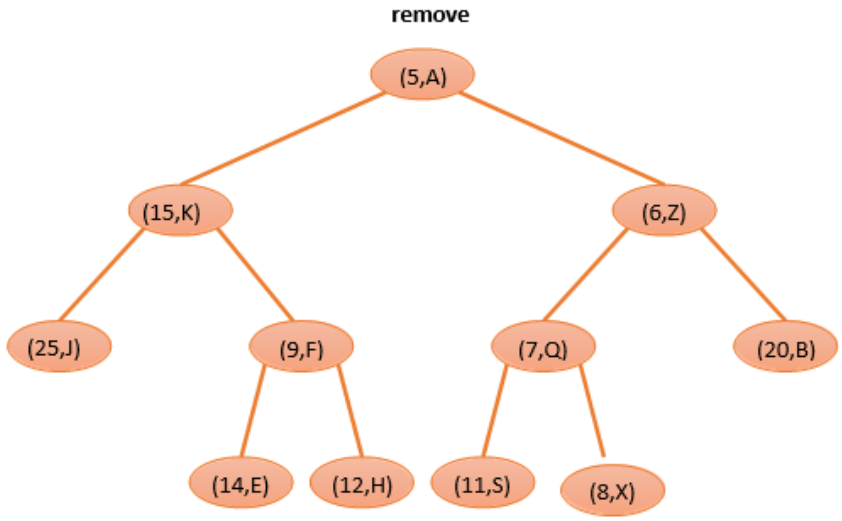





remove
(4,C)
(16,X)
(5,A)
(6,Z)
(15,K)
(9,F)
(7,Q)
(20,B)
(25,J)
(14,E)
(12,H)
(11,S)
(8,X)


remove
(16,X)
(5,A)
(6,Z)
(15,K)
(9,F)
(7,Q)
(20,B)
(25,J)
(14,E)
(12,H)
(11,S)
(8,X)


remove
(16,X)
(5,A)
(15,K)
(6,Z)
(25,J)
(9,F)
(7,Q)
(20,B)
(14,E)
(12,H)
(11,S)
(8,X)



12. Mostre como implementar o TAD pilha usando apenas uma fila de prioridade e uma variável inteira adicional.
    R: Utilizando um max-heap, criamos uma variável cont para contagem, que inicia com o valor 0. O cont será a chave utilizada para o entry, logo, deverá haver uma sobrecarga do construtor de entrada: o usuário irá passar como parâmetro apenas o valor a ser armazenado, o que seria equivalente ao push da pilha; este construtor deverá pegar o valor que o usuário passar e repassá-lo em uma chamada de função para a sobrecarga de construtor com os parâmetros chave e valor, atribuindo de forma autônoma como chave o valor de cont, que será incrementado após sua utilização, para a próxima entrada a ser feita. Feito isso, para fazer um pop, basta chamar a função equivalente a removeMax do max-heap, que consequentemente removerá o último elemento adicionado ao arranjo, equivalente ao topo da pilha.

13. Mostre como implementar o TAD fila (padrão) usando apenas uma fila de prioridade e uma variável inteira adicional.
    R: Utilizando um min-heap, criamos uma variável cont para contagem, que inicia com o valor 0. Novamente, o cont será a chave utilizada para o entry, logo, deverá haver uma sobrecarga do construtor de entrada: o usuário irá passar como parâmetro apenas o valor a ser armazenado, o que seria equivalente ao enqueue da fila; este construtor deverá pegar o valor que o usuário passar e repassá-lo em uma chamada de função para a sobrecarga de construtor com os parâmetros chave e valor, atribuindo de forma autônoma como chave o valor de cont, que será incrementado após sua utilização, para a próxima entrada a ser feita. Feito isso, para fazer um dequeue, basta chamar a função equivalente a removeMin do min-heap, que consequentemente removerá o primeiro elemento adicionado ao arranjo, equivalente ao começo da fila.

TAD Mapa

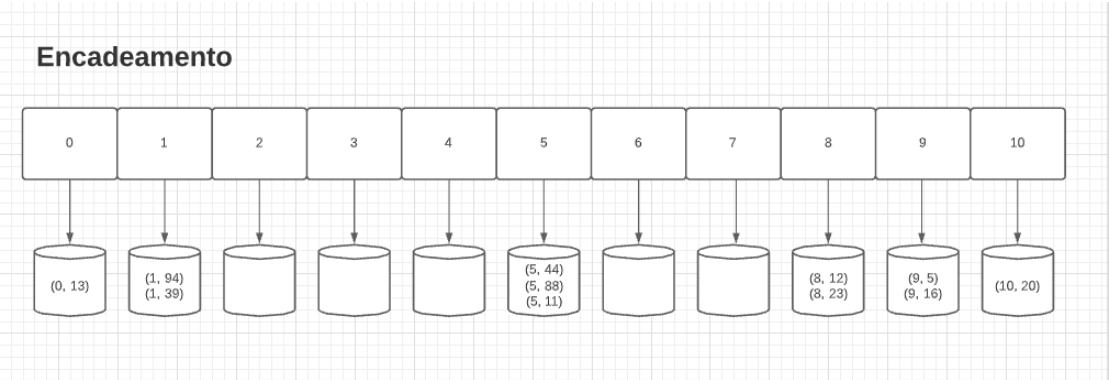
1. Qual dos esquemas de tratamento de colisão de tabela hash consegue tolerar um fator de carga superior a 1 e qual não consegue?

   R:
   Tolera (CARGA > 1): Encadeamento Separado;
   Não tolera (CARGA > 1): Endereçamento aberto (Teste linear, teste quadrático e hashing Duplo).

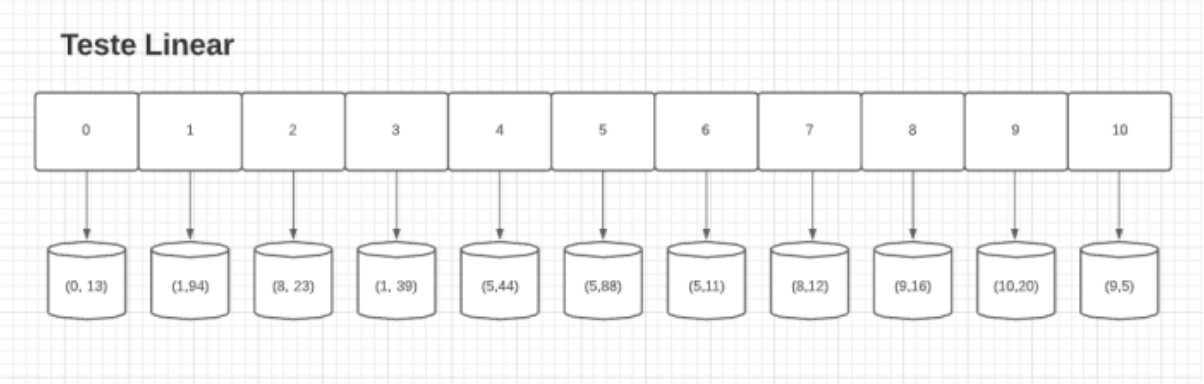
2. Qual seria um bom código hash para um número de identificação de veículo que é uma cadeia de caracteres representando números e letras no formato “9X9XX99X9XX999999,” onde um “9” representa um dígito e um “X” representa uma letra?

   R: Hash com shift - Resposta: 324564517

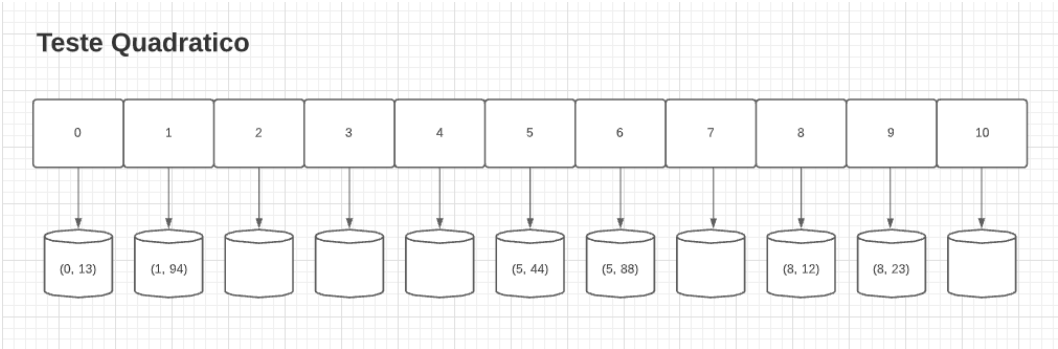
3. Desenhe a tabela hash com 11 elementos, que resulta a partir do uso da função de hash, h(i) = (3i + 5) mod 11, para colocar as chaves 12, 44, 13, 88, 23, 94, 11, 39, 20, 16 e 5, assumindo que as colisões serão tratadas por encadeamento.



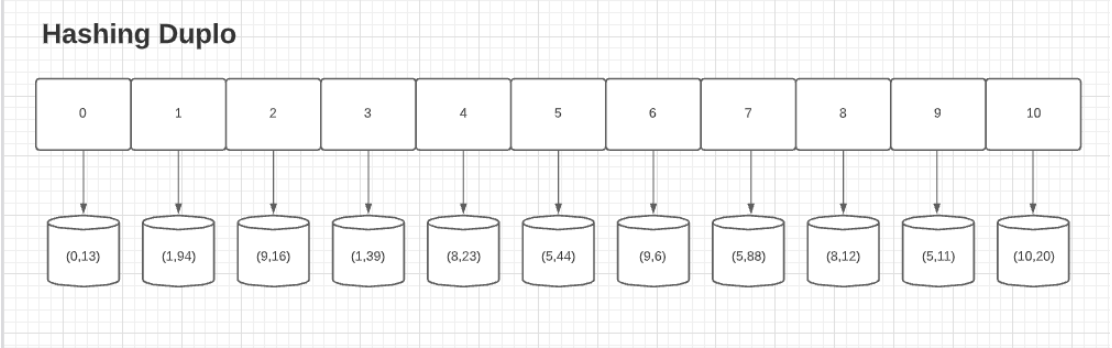
4. Qual será o resultado do exercício 3 se assumirmos que as colisões serão tratadas por teste linear?



5. Mostre o resultado do exercício 3 assumindo que as colisões são tratadas por teste quadrático, até o ponto em que o método falha.



6. Qual é o resultado do exercício 3 assumindo que as colisões são tratadas por hashing duplo usando uma função hash secundária h'(k)=7 (k mod 7)?



7. Forneça uma descrição em pseudocódigo da inserção em uma tabela hash que usa teste quadrático para resolver colisões, assumindo que se usa o truque de substituir elementos deletados com um objeto indicando "item desativado".

Algoritmo InsercaoHashQuadraticoSubstitutivo(k, v):

faz cálculo de hash

verifica a posição em **M** está disponível

se não está disponível, faz teste quadrático

quando encontra disponível, faz o **put**(k, v)

senão

falha do Teste Quadrático

8. Pesquise sobre o TAD Dicionário e descreva a principal diferença desse TAD com o TAD Mapa.

   R: A principal diferença é que: mapas devem ter chaves únicas, já dicionários podem ter múltiplas entradas com a mesma chave. Um HashMap é uma estrutura de dados mapeada que usa um algoritmo de hash para vincular a chave ao valor, e é uma das implementações possíveis do dicionário, logo, mapas são um subconjunto do dicionário. Um dicionário pode ter mais de um mapa de chaves para um valor, ou um mapa de chaves para vários valores.

9. Implemente e teste o HashTableMap considerando os slides de 66 a 72.
   R: // ---------- Implementação Java

10. Desafio:
    - Escreva uma classe de verificação ortográfica que armazena conjuntos de palavras, W, em uma tabela hash e implementa o método spellCheck(s) que executa uma verificação ortográfica sobre a string s relativa ao conjunto de palavras, W.
    - Se s está em W, então a chamada para spellCheck(s) retorna uma coleção iterável que contém apenas s, assumindo-se que tenha sido grafada corretamente neste caso.
    - Por outro lado, se s não está em W, então a chamada para spellCheck retona a coleção iterável de todas as palavras de W que podem corresponder à grafia correta de s.
    - O programa pode ser capaz de tratar todas as formas normais que s pode omitir uma palavra em W, incluindo trocar caracteres adjacentes de uma palavra, inserção de um único caractere entre outros dois, remoção de um único caractere de uma palavra e substituição de um caractere em uma palavra por outro.
      R: Não foi feito

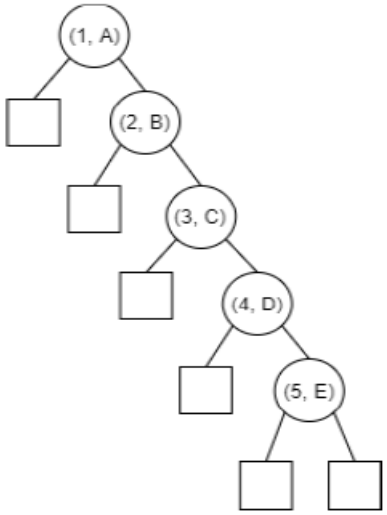
TAD Dicionário

1. Implemente e teste o TAD conforme os slides 9 a 15.
   R: // ---------- Implementação Java

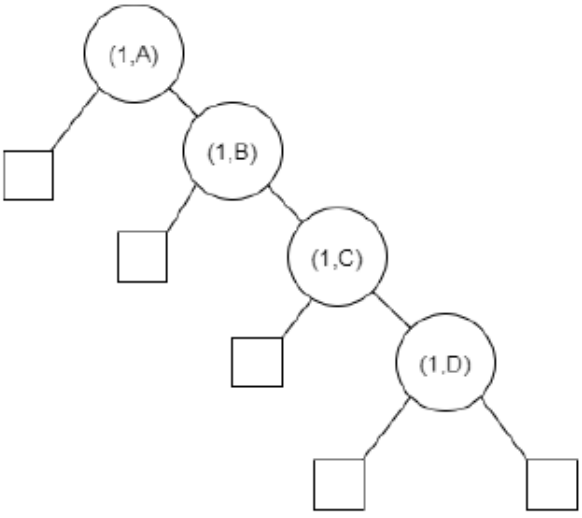
TAD Mapa Ordenado – Árvore Binária de Busca

1. Implemente e teste o TAD-Mapa Ordenado - Árvore Binária de Busca (slides de 21 a 27)
   R: // ---------- Implementação Java

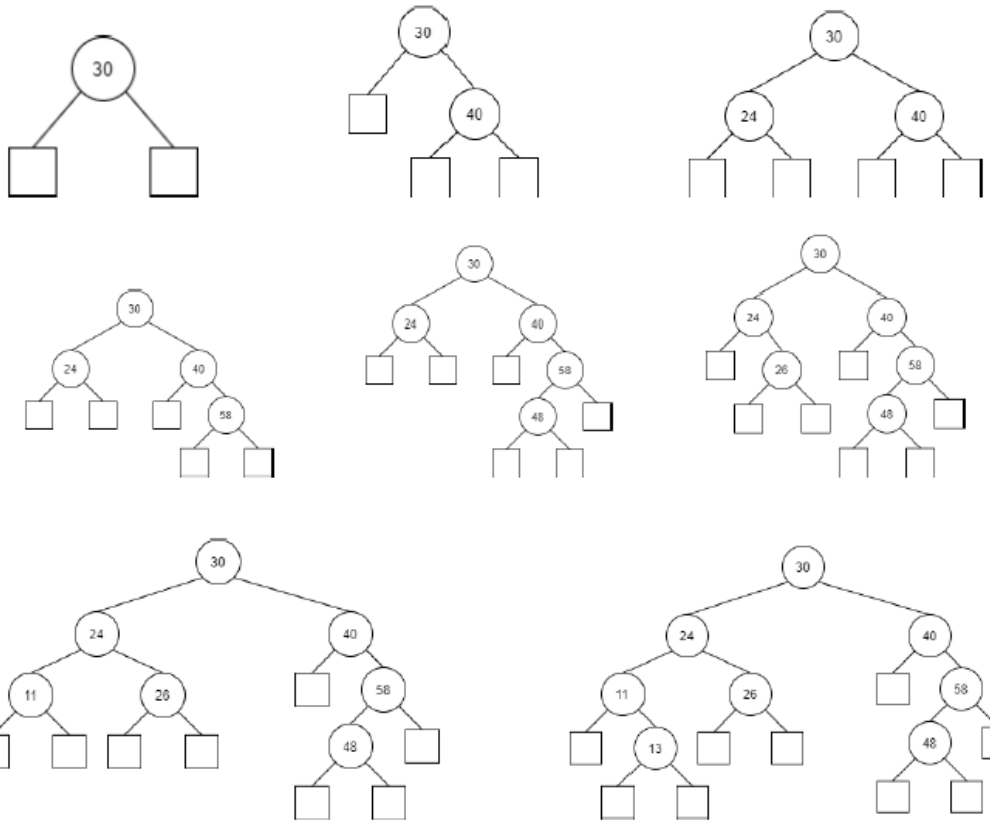
2. Inserindo-se as entradas (1,A), (2,B), (3,C), (4,D) e (5,E), nessa ordem, em uma árvore de pesquisa binária inicialmente vazia, qual será sua aparência?



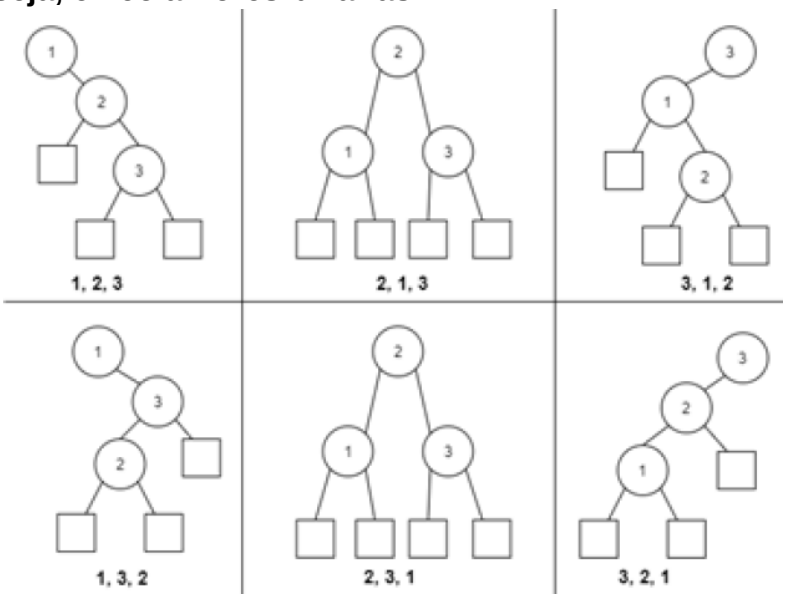
3. Define-se uma árvore binária de pesquisa em que as chaves iguais à chave do nodo podem estar ou à esquerda ou à direita da subárvore deste nodo. Suponha que se altere a definição na qual restringimos chaves iguais na subárvore à direita. Qual seria a subárvore de uma árvore binária de pesquisa que contenha somente chaves iguais, como visto neste caso?



4. Insira, em uma árvore binária de pesquisa vazia, itens com as chaves 30, 40, 24, 58, 48, 26, 11, 13 (nesta ordem). Desenhe a árvore após cada inserção.

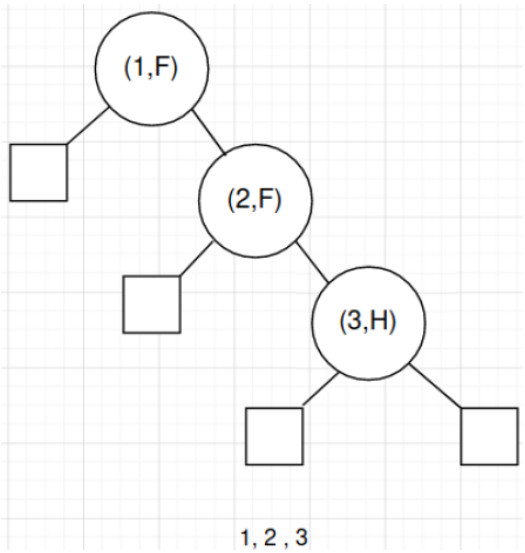


5. Quantas árvores binárias de pesquisas diferentes podem armazenar as chaves {1,2,3}?
   R: São possíveis seis combinações, porém duas delas são idênticas, ou seja, cinco árvores binárias.

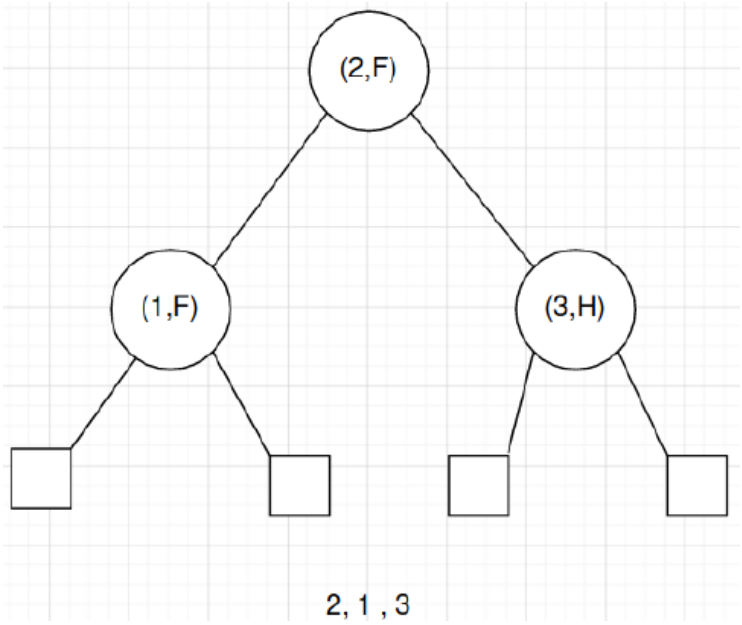


6. O professor Amongus afirma que a ordem na qual um conjunto fixo de itens é inserido em uma árvore binária de pesquisa não interessa — sempre resulta na mesma árvore. Apresente um pequeno exemplo que prove que ele está errado.

Dependendo do valor da chave do root, a árvore terá distribuição dos galhos de formas diferentes



Neste outro caso, o primeiro item a ser inserido é a chave valor 2, que gera a árvore do exemplo 2 a seguir.



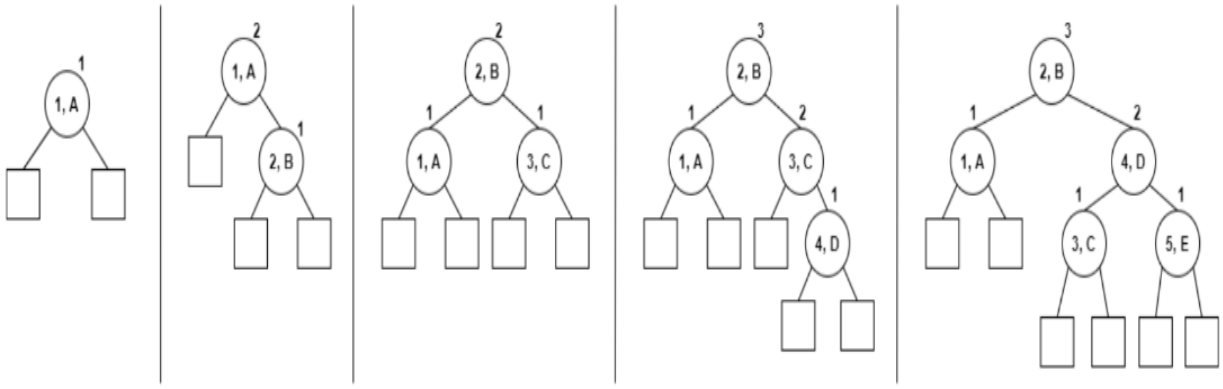
Conforme os exemplos apresentados acima a ordem dos itens a serem inseridos na árvore binária, altera sua estrutura.

TAD Mapa Ordenado – Árvore AVL

1. Implemente e teste o TAD-Mapa Ordenado - AVL (slides de 30 a 35)

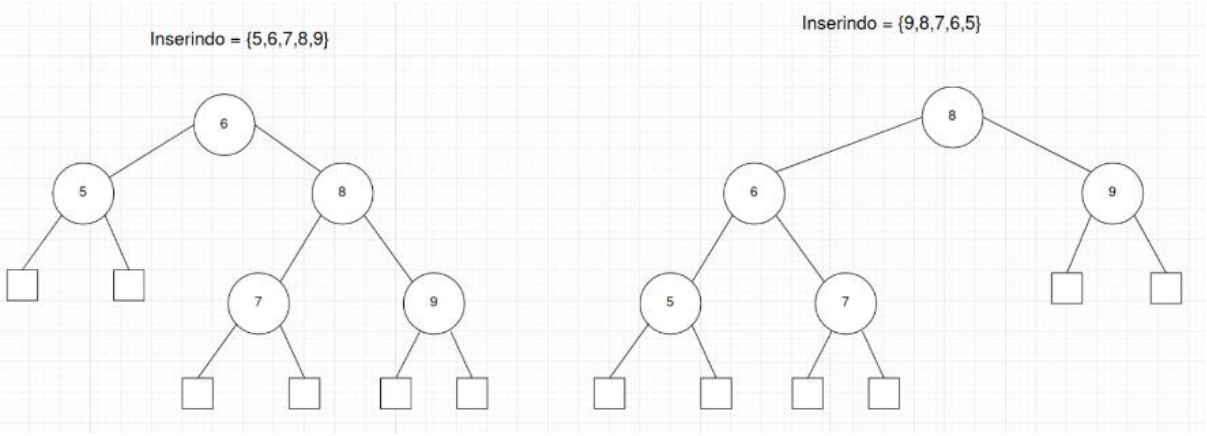
   R: // ---------- Implementação Java

2. Inserindo-se as entradas (1,A), (2,B), (3,C), (4,D) e (5,E), nessa ordem, em uma árvore AVL inicialmente vazia, qual será sua aparência?

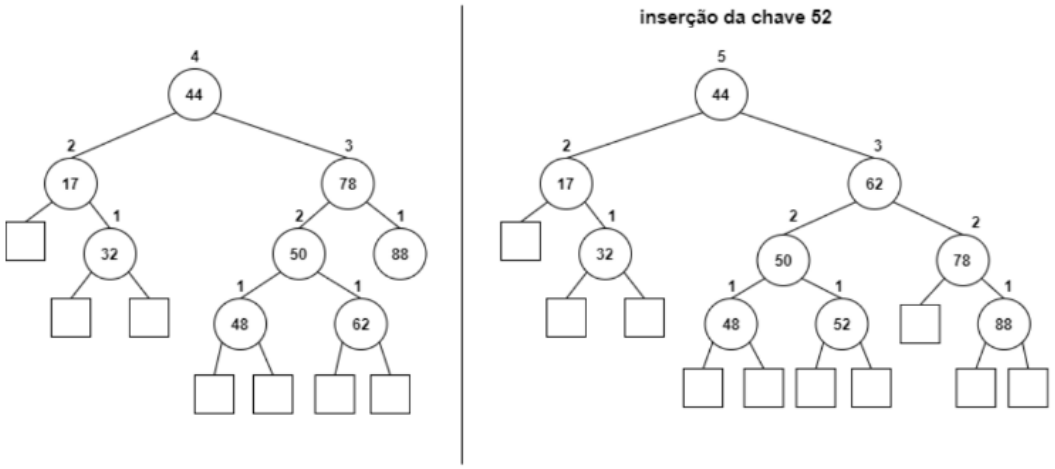


3. O professor Amongus afirma que a ordem na qual um conjunto fixo de itens é inserido em uma árvore AVL não interessa — sempre resulta na mesma árvore. Apresente um pequeno exemplo que prove que ele está errado.

   R: Realizando a inserção das chaves 5,6,7,8,9 e 9,8,7,6,5 na AVL resultarão em estruturas diferentes devido ao balanceamento.



4. Desenhe a árvore AVL resultante da inserção de um elemento com chave 52 na árvore AVL abaixo.



5. Desenhe a árvore AVL resultante da remoção de um elemento com chave 62 na árvore AVL após a inserção da chave 52 na árvore AVL abaixo.