Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II
Profs.: Felipe Domingos da Cunha, Max do Val Machado e Rodrigo Richard Gomes

Trabalho Prático IV

Regras Básicas

- `extends Trabalho Prático 03`
- Fique atento ao Charset dos arquivos de entrada e saída.

  Observação:

  Nas questões de árvore, utilizamos o mostrar pré.

  Não será necessário implementar a opção de remoção nas TADs abaixo.

A National Basketball Association (em português: Associação Nacional de Basquetebol; abreviação oficial: NBA) é a principal liga de basquetebol profissional da América do Norte. Com 30 franquias sendo membros da mesma (29 nos Estados Unidos e 1 no Canadá), a NBA também é considerada a principal liga de basquete do mundo. É um membro ativo da USA Basketball (USAB), que é reconhecida pela FIBA (a Federação Internacional de Basquetebol) como a entidade máxima e organizadora do basquetebol nos Estados Unidos. A NBA é uma das 4 'major leagues' de esporte profissional na América do Norte. Os jogadores da NBA são os mais bem pagos esportistas do mundo, por salário médio anual.

A liga foi fundada na cidade de Nova Iorque em 6 de Junho de 1946, como a Basketball Association of America (BAA). A liga adotou o nome de National Basketball Association em 1949 quando se fundiu com a rival National Basketball League (NBL). A liga tem diversos escritórios ao redor do mundo, além de vários dos próprios clubes fora da sede principal na Olympic Tower localizada na Quinta Avenida 645. Os estúdios da NBA Entertainment e da NBA TV são localizados em Secaucus, New Jersey.

O arquivo **players.csv** contém um conjunto de dados de jogadores da liga de basquete norte americana - NBA extraídos do site `https://www.kaggle.com/drgilermo/nba-players-stats`. Essa base contém registros de jogadores desde 1950. Um total de 67 temporadas da NBA. Este arquivo **sofreu algumas adaptações** para ser utilizado neste e nos próximos trabalhos práticos. Tal arquivo deve ser copiado para a pasta /tmp/. Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

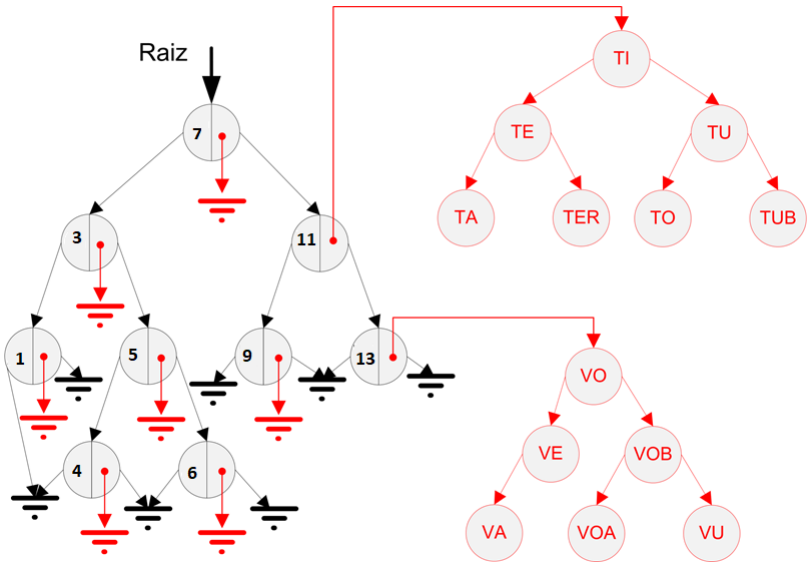
Árvores

Observação: ATENÇÃO para os algoritmos de árvore que já estão implementados no Github!

1. **Árvore Binária em Java**: Crie uma Árvore Binária, fazendo inserções de registros conforme a entrada padrão. A chave de pesquisa é o atributo **Nome do Jogador**. Não insira um elemento se sua chave estiver na árvore. Em seguida, pesquise se alguns registros estão cadastrados na Árvore, mostrando seus respectivos caminhos de pesquisa. A entrada padrão é igual a da questão de "Pesquisa Sequencial". A saída padrão é composta por várias linhas, uma para cada pesquisa. Cada linha é composta pelo caminho ou sequência de ponteiros (`raiz, esq ou dir`) utilizados na pesquisa e, no final, pelas palavras SIM ou NAO. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_arvoreBinaria.txt com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

2. **Árvore Binária de Árvore Binárias em Java**: Refaça a questão anterior, contudo, considerando a estrutura de árvore de árvore. Nessa estrutura, temos uma árvore binária tradicional na qual cada nó tem um ponteiro para outra árvore binária. Graficamente, a primeira árvore está no plano xy e a árvore de seus nós pode ser imaginada no espaço tridimensional. Temos dois tipos de nós. O primeiro tem um número inteiro como chave, os ponteiros esq e dir (ambos para nós do primeiro tipo) e um ponteiro para nós do segundo tipo. O outro nó tem uma String como chave e os ponteiros esq e dir (ambos para nós do segundo tipo). A chave de pesquisa da primeira árvore é o atributo **altura mod 15** e, da outra, é o atributo **Nome do Jogador**. Conforme a figura abaixo.

   Destaca-se que nossa pesquisa faz um "mostrar" na primeira árvore e um "mostrar" na segunda. Faremos um "mostrar" na primeira árvore porque ela é organizada pelo **altura mod 15**, permitindo que o valor desejado esteja na segunda árvore de qualquer um de seus nós. Faremos o "mostrar" na segunda porque ela é organizada pelo atributo **Nome do Jogador**. Antes de inserir qualquer elemento, crie a primeira árvore, inserindo todos seus nós e respeitando a ordem **7, 3, 11, 1, 5, 9, 13, 0, 2, 4, 6, 8, 10, 12 e 14**. O arquivo de log será matrícula_arvoreArvore.txt.



3. **Árvore AVL em C**: Refaça a primeira questão deste trabalho com Árvore AVL em C. O nome do arquivo de log será matrícula_avl.txt.

4. **Árvore Alvinegra em Java**: Refaça a primeira questão deste trabalho com Árvore Alvinegra. O nome do arquivo de log será matrícula_avinegra.txt.

5. **Tree sort em Java**: Um tipo de árvore que é um algoritmo de ordenação. Este constrói uma árvore de busca binária dos elementos a serem classificados, em seguida, percorre a árvore (em ordem) para que os elementos sair em ordem de classificação. Efetue a ordenação dos jogadores pelo nome usando o algoritmo `Tree sort`. Ao seu arquivo de log, contabilize a quantidade de comparações efetuadas para inserção de todos os jogadores na árvore. O nome do arquivo de log será matrícula_treesort.txt.

6. **Árvore *Trie* em Java**: Crie duas árvores do tipo *trie* com os nomes dos Jogadores, fazendo inserções de registros conforme a entrada padrão. Não insira um elemento se sua chave estiver na árvore. Em seguida, faça o merge das duas árvores. Na árvore resultante (que não tem nomes repetidos), pesquise alguns nomes. A entrada padrão é igual a da questão de "Pesquisa Sequencial". Entretanto, após a primeira ocorrência da palavra FIM, temos outros Jogadores que devem ser inseridos na segunda árvore. A saída padrão é composta por várias linhas, uma para cada pesquisa resultando nas palavras SIM ou NÃO. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_arvoreTrie.txt com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

7. **Tabela *Hash* Direta com Reserva**: Refaça a primeira questão deste trabalho com Tabela *Hash* Direta com Reserva. A função de transformação será **altura mod tamTab** onde tamTab (tamanho da tabela) é 21. A área de reserva tem tamanho 9, fazendo com que o tamanho total da tabela seja igual a 30. A saída padrão será a posição de cada elemento procurado na tabela

(na *hash* ou na área de reserva). Se o elemento procurado não estiver na tabela, escreva a palavra NÃO. Além disso, o nome do arquivo de log será matrícula_hashReserva.txt.

8. **Tabela *Hash* Direta com Rehash**: Refaça a questão anterior com Tabela *Hash* Direta com *Rehash*. A primeira função de transformação será **altura mod tamTab** onde tamTab (tamanho da tabela) é 25 e a outra, **(altura + 1) mod tamTab**. O nome do arquivo de log será matrícula_hashRehash.txt.

9. **Tabela *Hash* Indireta com Lista Simples em C**: Refaça a questão anterior com Tabela *Hash* Indireta com Lista Simples. A função de transformação será **altura mod tamTab** onde tamTab (tamanho da tabela) é 25. O nome do arquivo de log será matrícula_hashIndireta.txt.