UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
Departamento de Ciências de Computação

SCC0502 – Algoritmos e Estruturas de Dados I

Trabalho 2 – especificação

Em grupos de até dois alunos, resolva os problemas abaixo. Ao final, um dos membros da dupla deve submeter um arquivo compactado na atividade aberta no Tidia, incluindo os TADs necessários para compilação do programa.

Parte 1 (4,0)

Avalie e compare os 3 métodos de ordenação padrões de um conjunto de dados: por Inserção (insertion sort), por bolha (bubble sort) ou por união (merge sort).

Para efeito deste projeto, assumiremos todos os métodos vão ordenar em ordem crescente e o conjunto antes e depois da ordenação deverá estar em uma **LISTA dinâmica e encadeada**.

Para tal avaliação, um mesmo conjunto deverá ser ordenado usando os três métodos e o tempo contado. Como vários fatores fora do seu controle podem interferir na contagem de tempo (interação com outros programas, desvio de recursos pelo SO, ação do anti-virus, etc...), o programa deve realizar múltiplas ordenações do mesmo conjunto usando cada método e retornar a média dos resultados.

Como entrada, o programa (na main.c) deve pedir ao usuário um número inteiro (N), que será o número de itens no conjunto a ser ordenado. Em seguida um segundo número inteiro será pedido, que será o número de repetições (R) que o programa fará, antes de apresentar a média.

Uma vez que o usuário escolha N e R, o programa deve avaliar os três métodos em diferentes casos, criando 3 conjuntos de inteiros a serem ordenados: Um inicialmente em ordem crescente, um inicialmente em ordem decrescente e um inicialmente aleatório. Os números podem ser repetidos no conjunto aleatório.

O resultado deve ser apresentado em 9 valores claramente explicitados. O tempo médio gasto para cada método ordenar o caso crescente, em seguida o tempo médio para cada método ordenar o caso decrescente e por último o tempo médio para cada método ordenar o caso aleatório.

Por fim, realize um conjunto de testes e justifique os resultados obtidos usando um pequeno arquivo de texto que deve ser enviado junto com o trabalho. A justificativa não precisa ser complexa, apenas avalie se o resultado condiz com o esperado e porquê.

Para calcular o tempo, utilize as funções da biblioteca **time.h**. Para criar o conjunto aleatório utiliza a função **rand()** com uma semente determinada por **srand()**.

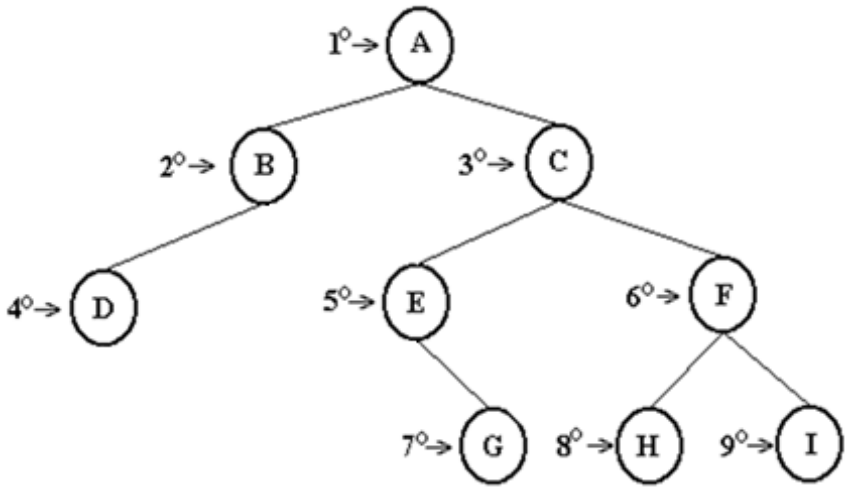
OBS: É necessário usar uma semente e a semente utilizada deve ser sempre a mesma, tornando os resultados reprodutíveis. Use o número USP de um dos integrantes da dupla como semente.

Parte 2 (6,0)

Os alunos deverão implementar, em C, um código que peça ao usuário uma string de até 100 caráteres diferentes (incluindo números e símbolos contidos na tabela ASCII, até o valor 127), finalizada quando o usuário pressionar enter, que devem ser armazenados em largura em uma árvore.

A string deve ser armazenada, **elemento a elemento**, em uma **árvore binária dinâmica** de char, em largura. O caractere espaço (número 32 em ASCII), deve representar ausência de um nó. Conforme o exemplo abaixo, onde '·' representa um caractere espaço:

Entrada: ABCD·EF·····GHI



Armazenamento em largura

Por fim, o programa deve caracterizar a árvore, informando os seguintes dados:

- Impressão da árvore completa (utilize a função imprimir mostrada em sala)
- Altura?
- Número de nós folha?
- É uma árvore cheia?
- É uma árvore binária de busca?
- É uma árvore AVL?
- Os elementos ficam ordenados se apresentados em Pré-ordem?
- Os elementos ficam ordenados se apresentados em ordem?
- Os elementos ficam ordenados se apresentados em Pós-ordem?

Estas informações devem ser obtidas usando a árvore construída, **NÃO** a string original de entrada.

As strings inseridas sempre seguirão o formato discutido em sala para uma árvore estática (por vetor), não precisa manejar **este** erro especifico no programa. Desconsiderem o caractere '\0' no fim da string.

Por fim, o programa deve reestruturar a árvore binária como uma **árvore AVL** e caracterizar a árvore resultante, com as mesmas informações acima. Isso deve ser feito inserindo os elementos da string novamente (na mesma ordem), agora usando a função de inserção em árvore AVL discutida em sala.

OBS. Sugiro utilizar a função fgets() de c para obter a string do usuário, evitando problemas com os espaços.