UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação - ICMC
SCC 202/502 – Algoritmos e Estrutura de Dados I - 2º Sem /2018
Prof. Robson Cordeiro
PAE: Guilherme Queiroz, Leonardo Moraes

Trabalho 2: Manipulação de Dados em Memória Primária

Entregar no Tidia até: 08/12/2018

1. Motivação e Objetivo

Para **agilizar** a recuperação de dados, cientistas da computação têm proposto diversos algoritmos, heurísticas e estruturas de dados para adequar-se às diferentes aplicações. Dentre as estruturas de dados clássicas, existem: a **lista encadeada**, **árvore binária**, **árvore AVL** (*Adelson-Veslky & Landis*), dentre outras. Cada uma dessas estruturas possui diferentes particularidades das quais uma aplicação pode se beneficiar.

Este trabalho tem como objetivo familiarizar o estudante com as diversas estruturas de dados, propondo uma **análise crítica** através da implementação e utilização destas estruturas em grandes bases de dados. Para isso, o estudante deverá:

1. Implementar estruturas para manipulação eficiente de dados.
2. Avaliar a escalabilidade dos algoritmos medindo o tempo de **inserção**, **busca** e **remoção** de dados.
3. Ilustrar os resultados do item 2 em formato **gráfico** e apresentar uma justificativa detalhada de bons e maus **resultados**, incluindo as **vantagens** e **desvantagens** de cada estrutura implementada no Item 1.

2. Indicações Gerais

- O trabalho deve ser feito em dupla;
- Serão aceitos apenas trabalhos em linguagem de programação C.

3. Especificações

3.1 Das estruturas

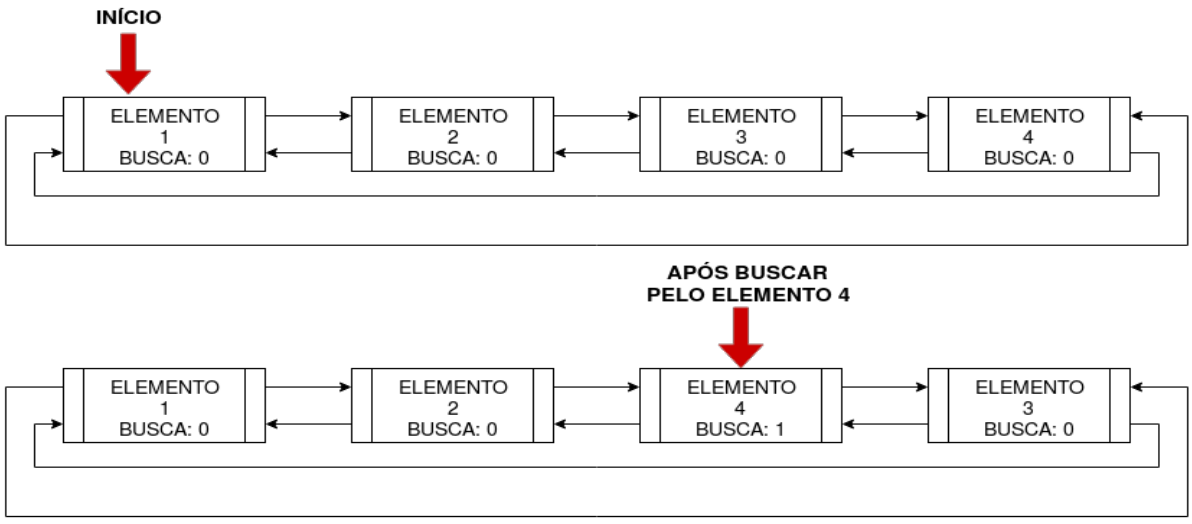
Seu código deve contemplar as funções de inserção, remoção e busca para as seguintes estruturas clássicas:

- (BB) Busca binária (estática/sequencial);
- (LO) Lista ordenada (dinâmica/encadeada);

- (LOS) Lista ordenada com sentinela para agilizar buscas (dinâmica/ encadeada);
- (ABB) Árvore binária de busca (dinâmica/encadeada);
- (AVL) Árvore AVL (dinâmica/encadeada);
- (LFREQ) Lista circular dinâmica duplamente encadeada, organizada pela frequência de busca. (ver Item 3.1.1)

3.1.1 Lista circular dinâmica duplamente encadeada, organizada pela frequência de busca - LFREQ

Uma busca sequencial é uma operação que localiza um valor específico entre os valores armazenados em uma lista. Para isso, o valor procurado é comparado com o valor de cada nó da lista na ordem em que os nós estão armazenados. Para melhorar o desempenho dessa operação pode-se manter os nós da lista ordenados por frequência de consulta, ou seja, os nós com valores mais frequentemente pesquisados devem ficar mais próximos do início da lista. Isso pode ser feito modificando-se a operação de busca. Quando o valor procurado é encontrado em um nó `p!=inicio`, `p` deve ser trocado de posição com o nó imediatamente anterior a ele; pode-se também trocar apenas a informação dos nós. Após várias consultas, a lista tenderá a manter os nós com valores mais frequentemente pesquisados nas primeiras posições.



3.2 Da implementação

Deve-se criar uma única função `int main(void){...}` que execute as seguintes operações em cada estrutura para diversos valores de `n`:

1. **Inserção** de `n` números inteiros em ordem crescente (de `0` a `n-1`), decrescente (de `n-1` a `0`) e aleatória (entre `0` e `n-1`);

2. **Remoção** de `n` números inteiros em ordem crescente, decrescente e aleatória:
    a. Em todos os casos, invoque a função de remoção para `n/2` elementos existentes na estrutura e `n/2` não existentes;
3. **Busca** por `n` números inteiros aleatórios:
    a. Busque por `n/2` elementos existentes e `n/2` não existentes;

Saída e formato: imprima `9` tabelas como a seguinte com **tempo médio** relativo a 10 execuções das operações anteriormente descritas, para cada estrutura de dados e valor de `n`. Faça a impressão com a função `printf()`.

```
Tabela 1: Tempo de insercao crescente
     n=100      n=1.000    n=10.000   n=100.000
BB
LO
LOS
ABB
AVL
LFREQ
```

Cada linha da tabela contém números reais separados por tabulação. As outras tabelas seguem o mesmo formato, com referência a:

- `Tabela 2: Tempo de insercao decrescente`
- `Tabela 3: Tempo de insercao aleatoria`
- `Tabela 4: Tempo de remocao crescente (apos insercao crescente)`
- `Tabela 5: Tempo de remocao decrescente (apos insercao crescente)`
- `Tabela 6: Tempo de remocao aleatoria (apos insercao aleatoria)`
- `Tabela 7: Tempo de busca (apos insercao crescente)`
- `Tabela 8: Tempo de busca (apos insercao decrescente)`
- `Tabela 9: Tempo de busca (apos insercao aleatoria)`

A saída deve respeitar **com exatidão** o formato definido. Para tanto, é disponibilidado o programa de teste de formato `testaFormato.c` e o arquivo `exemplo.txt` com um exemplo de saída válida. Para verificar se a saída de seu arquivo executável `exec` está no formato correto, rode os comandos a seguir:

```
./exec > saida.txt
gcc -o testaFormato testaFormato.c
./testaFormato saida.txt
```

Deve aparecer na tela a string `"Formato correto"`.

ATENÇÃO: Códigos que não seguirem o formato correto de saída terão nota ZERO, sem exceção!

3.3 Da análise crítica

Use os dados das tabelas obtidas no item anterior para montar e apresentar, em um relatório, gráficos com o valor de **`n`** no eixo `X` e a **média de tempo** (em escala logarítmica) no eixo `Y`. Assim, cada gráfico terá **`4` pontos no eixo `X`** representando o número crescente de elementos e **`6` curvas** representando as estruturas de dados utilizadas. As tabelas originais também devem ser apresentadas no relatório, junto aos gráficos correspondentes.

IMPORTANTE: Para cada gráfico/tabela, deverá ser apresentada uma **discussão detalhada** sobre os resultados obtidos, elencando as vantagens e desvantagens de cada estrutura para aquele caso em particular, isto é, os motivos para bons ou maus resultados. Adicionalmente, indique qual é a melhor estrutura para um **caso geral**, em que executam-se operações de busca, remoção e inserção indiscriminadamente, de forma que não seja possível prever as operações mais frequentemente executadas e nem mesmo os valores dados como entrada para as mesmas. Apresente **JUSTIFICATIVA DETALHADA** para sua indicação. Por fim, deve-se apresentar uma conclusão sucinta sobre o desenvolvimento deste trabalho.

4. Critérios de Avaliação

O comportamento das estruturas clássicas é **conhecido**, portanto, os gráficos e tabelas **devem** apresentar resultados **condizentes** com a literatura, obviamente, considerando diferenças de hardware. Observem que os resultados reportados no relatório refletirão a qualidade de sua implementação. Ainda sobre a implementação será avaliado:

- **Padronização**: o programa deve respeitar estritamente o formato de saída previamente definido;
- **Corretude**: o programa deve fazer o que foi especificado;
- **Estruturas de dados utilizadas**: adequação e eficiência;
- **Observação de "boas práticas" de programação**: TAD, modularidade do código, documentação interna, indentação, etc.

Sobre a discussão e análise crítica: ambas devem ser **bem fundamentadas**, explicadas com **clareza** e complementar os gráficos.

OBSERVAÇÃO

• Plágio de programas ou de material adicional não será tolerado; será utilizada ferramenta automática de detecção de plágio, além de análise manual.

*Qualquer material similar ao de outra dupla terá **NOTA ZERO** independentemente de qual for a cópia.*

5. Entrega

ATENÇÃO: Confira sua submissão no Tidia. Trabalhos não submetidos terão nota ZERO, sem exceção!

A entrega do trabalho será via **Atividade do Tidia**. São requeridos:

1. **Códigos:** um arquivo comprimido (ZIP) que deverá ser postado por apenas um dos membros do grupo. Este deve conter (a) arquivo `readme.txt` com nomes e números USP dos integrantes do grupo; (b) arquivos de código-fonte do programa; e (c) arquivo `howtodo.txt` com passo a passo de como compilar o programa **em gcc** (deixar claro que compilador usou, versão, se utilizou alguma biblioteca diferente, qual sistema operacional utilizou, etc).
2. **Relatório:** o relatório solicitado no Item 3.3, com no mínimo `4` páginas e no máximo `10` páginas.

Entregas em atraso: a nota máxima do trabalho será decrescida de `1.0` ponto ponto por dia de atraso na entrega.