Trabalho 3 – análise experimental da eficiência de algoritmos de busca

Considere um sistema em que são realizadas buscas e inserções em arquivos binários que contém chaves inteiras de quatro bytes.

Há um arquivo principal já ordenado de maneira decrescente com chaves inteiras já existente no sistema.

Considere que um usuário:

1. indica quantas K inserções ele fará via teclado,
2. em seguida, digita essas K chaves. Considere que todas essas inserções serão feitas de uma única vez em um arquivo temporário, ou seja, não serão feitas no arquivo principal.
3. indica quantas X buscas que fará.
4. em seguida, digita as X chaves inteiras a serem buscadas.

Dados os seguintes valores:

- o nome do arquivo principal,
- o número K de inserções,
- os valores das K chaves digitadas pelo usuário via teclado,
- o número X de buscas
- as X chaves a serem consultadas,

Calcule o **número de comparações executadas** para:

1) **Método 1**: Primeiramente busque de maneira sequencial no arquivo principal e, em seguida, no temporário. Tão logo a chave seja encontrada, ela será retornada, ou seja, se a chave for encontrada no arquivo principal, não há a necessidade de procurá-la no temporário

2) **Método 2**: Utilize a busca binária pela chave no arquivo principal e, caso essa não seja encontrada, deve-se procurá-la no temporário de maneira sequencial. Tão logo a chave seja encontrada, ela será retornada;

Exemplo

Considere que o programa receberá como entrada:

```
arquivo.dat
3
1 2 15
4
3 13 15 99
```

Em que arquivo.dat é o nome do arquivo principal já ordenado de maneira decrescente. O valor 3 indica o número K de chaves que serão inseridas no arquivo temporário. Os valores das chaves a serem inseridas no arquivo temporário são: 1, 2 e 15. O valor 4 indica as X buscas que fará no sistema, em que

3, 13, 15 e 99 são as chaves a serem procuradas.

As saídas produzidas pelo programa devem respeitar o formato:

<núm-comparações-chave1-metodo1>_<núm-comparações-chave1-metodo2>**\n**
<núm-comparações-chave2-metodo1>_<núm-comparações-chave2-metodo2>**\n**
…
<núm-comparações-chaveX-metodo1>_<núm-comparações-chaveX-metodo2>**\n**

Exemplo de saída:

3_3**\n**
11_4**\n**
14_3**\n**
14_3**\n**

Em que _ são espaços em branco e **\n** são quebras de linha. Será fornecido um par de arquivos de entrada e saída, e um arquivo binário com dados para testes. Você poderá produzir seus próprios arquivos para testes adicionais.

Informações importantes

- Sobre a avaliação

    1. Um dos objetivos da disciplina de ICC2 é o **aprendizado individual** dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos, nem codificar em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código.
        - O plágio vai contra o código de ética da USP.
        - Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação.
        - O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e idéiar. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você **não** deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado do seu colega:
            - depois de o seu colega ter visto o seu código, será muito mais difícil para ele imaginar uma solução original e própria;
            - o seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.
        - Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela.
        - O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2. Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e **os trabalhos com alta similaridade detectada terão suas notas zeradas**, tanto aqueles relativos ao código de origem quanto do código copiado.

3. A avaliação incluirá a porcentagem de acertos verificada pelo SQTPM e também a análise do seu código, incluindo endentação, comentários, bom uso da memória e práticas de programação. Portanto faça seu código com cuidado, da melhor forma possível.

- Sobre o sistema de submissão (SQTPM):

1. Seu código deverá ser submetido num arquivo fonte .c. Esse arquivo deverá **obrigatoriamente** conter no início um comentário com seu nome, número USP, turma e data da entrega do trabalho.

2. A compilação do código é feita pelo comando:
```
gcc -o prog proc.c -lm -Wall
```

3. A saída do seu programa deve ser exatamente igual à saída esperada, incluindo: espaços em branco, quebras de linha e precisão decimal.

4. Há um limite de 30 segundos para a execução dos 10 casos de teste e um limite de 18.5MB de memória total para ser utilizada. Você deverá gerenciar bem o tempo de execução e o espaço ocupado para que seu programa fique dentro desses limites. O erro de "Violação de Memória" significa que seu programa está usando mais do que 18.5MB. Esse erro também pode significar vazamento de memória, acesso indevido a arranjo ou arquivo.

5. Como se trata de leitura de arquivos, o servidor poderá demorar um pouco para mostrar os casos de teste, após o envio. Aguarde um pouco mais de 30 segundos, sem recarregar a página, para obter a resposta.