Algoritmos e Estruturas de Dados II
2º semestre de 2013 - Turma 02

Segundo Exercício Programa/Trabalho

Busca em Arquivos

1 Introdução

A proposta deste exercício programa é exercitar conceitos vistos em aula sobre busca em arquivos. Os arquivos sobre os quais as buscas serão realizadas serão compostos por registros de tamanho fixo, cada um com 128 bytes, sendo que cada registro possui: um campo **id**, um identificador numérico composto por 4 bytes cujo valor é sempre positivo (maior ou igual a zero); e um campo **conteudo**, um vetor de 124 caracteres capaz de representar *strings* de até 123 caracteres. Esse registro corresponde à seguinte estrutura na linguagem C:

```
typedef struct {

    int32_t id;
    char conteudo[124];

} Registro;
```

Sua tarefa neste EP consiste em implementar 3 algoritmos distintos de busca: busca sequencial, busca binária e busca por índice. A busca por índice, em particular, deve ser implementada através de um índice com 2 níveis: nível "topo" e o nível "base". O nível "base" corresponde ao índice denso para arquivo de registros, enquanto o nível "topo" corresponde ao índice esparso (e portanto menor) para o índice "base". Como a realização da busca binária, assim como a geração de índices, exigem a ordenação prévia de registros e entradas de índice, vocês também deverão implementar o algoritmo de ordenação externa *k-way merge*.

As buscas sobre os arquivos serão sempre executadas tendo identificadores como chaves de pesquisa, e as ordenações devem ser feitas em ordem crescente de valor do identificador. Além disso, pode-se assumir que cada identificador de registro contido em um arquivo é único. Para a implementação tanto das buscas quanto da ordenação externa, as leituras e escritas em disco devem ser feitas sempre em blocos de 4096 bytes, e deve-se assumir que seu EP só é capaz de manter um máximo de 2000 blocos em memória RAM em um dado instante.

Além das implementações, também deverão ser realizados alguns experimentos, a fim de comparar o desempenho de cada algoritmo de busca implementado, e deverá ser produzido um relatório com os resultados obtidos. As análises dos resultados deverão levar em conta dois critérios distintos, embora relacionados entre si: número de blocos lidos durante uma busca e tempo de execução de uma busca.

2 Detalhes do funcionamento do EP

A execução do EP deve ser feita da seguinte forma:

```
>./ep2 config.txt
```

onde `config.txt` é um arquivo de configuração, em formato texto, que possui a seguinte estrutura:

```
<operação>
<arquivos de entrada>
<arquivos de saída>
<numero de buscas>
<id 0>
<id 1>
<id 2>
...
<id k-1>
```

O parâmetro `operação` define o qual operação seu programa irá executar e deve ter um dos seguintes valores: **ordena**, para realizar ordenação externa; **gera_indice**, para gerar os arquivos de índice a serem usados posteriormente em buscas por índice; **busca_seq**, para executar uma busca sequencial; **busca_bin**, para executar uma busca binária; e, finalmente, **busca_indice** para executar uma busca através de consulta a arquivos de índice.

O parâmetro `arquivos de entrada` definem ou mais nomes de arquivos de entrada, que possuem significados ligeiramente distintos dependendo da operação a ser executada. Para a ordenação externa, geração do índice e busca sequencial, deve-se especificar apenas um arquivo de entrada, que nada mais é do que do que um arquivo composto por uma sequencia de registros (cuja estrutura já foi especificada). Para a busca binária, deve-se especificar também um único arquivo de entrada, que também é um arquivo composto por uma sequência de registros, com a diferença que os registros devem estar ordenados em ordem crescente de identificador. Já para a busca por índice, devem ser especificados 3 arquivos de entrada: indice "topo", indice "base" e arquivo de registros propriamente dito.

O parâmetro `arquivos de saída` definem um ou mais nomes de arquivos de saída, que e também possuem significados ligeiramente distintos dependendo da operação especificada. Para a ordenação externa, deve-se definir um único arquivo de saída que deverá conter todos os registros do arquivo de entrada, ordenados por ordem crescente de identificador. Para a geração de índices, devem ser especificados 2 arquivos de saída: indice "topo" e indice "base" que deverão conter, respectivamente, os índices esparso e denso do arquivo de entrada. Para as buscas (sequencial, binária ou por índice) deve-se especificar apenas um arquivo de saída que irá conter os resultados das buscas realizadas.

As linhas seguintes no arquivo de configuração deverão estar especificadas apenas quando alguma das buscas for escolhida através do parâmetro `operação`. O parâmetro `número de buscas` define o

número k de buscas a serem realizadas. Seguindo esta linha deve haver uma sequência de k linhas, cada uma contendo um identificador a ser pesquisado. O arquivo de saída para as buscas devem apresentar a seguinte formatação:

```
<id 0> <resultado> <conteudo> <número de blocos lidos> <tempo de busca>
<id 1> <resultado> <conteudo> <número de blocos lidos> <tempo de busca>
...
<id k-1> <resultado> <conteudo> <número de blocos lidos> <tempo de busca>
```

Cada uma das k linhas presentes no arquivo de saída apresentam informações referentes às buscas de cada identificador definido no arquivo de configuração. Nestas linhas: `resultado` deve ser a string **SIM** (para uma busca bem sucedida) ou a string **NAO** (quando um identificador pesquisado não é encontrado — note que a string **NAO** não deve estar acentuada a fim de evitar eventuais problemas devido a *encodings* diferentes em plataformas distintas); `conteudo` deve conter, entre aspas simples, a string referente ao campo **conteudo** do registro encontrado, em caso de busca bem sucedida, ou a string vazia `''` caso contrário; `número de blocos lidos` é um valor inteiro que indica a quantidade de blocos de disco que foram lidos durante a busca pelo identificador; e `tempo de busca` é um valor inteiro que indica quantos milissegundos a busca demorou para ser realizada.

2.1 Sobre a implementação da busca por índice

Como já mencionado, a busca por índice deve ser feita a partir de um índice em 2 níveis, onde o nível "base" corresponde a um índice denso para o arquivo de registros (isto é, existe uma entrada neste índice para cada registro do arquivo), e o índice "topo" corresponde a um índice esparso para o índice "base" (isto é, apenas algumas entradas do índice "base" são indexadas pelo índice "topo").

Cada entrada dos índices também devem ser tratadas como registros de tamanho fixo, cada um com 8 bytes, sendo os primeiros 4 bytes referentes ao **id** do registro, e os 4 bytes seguintes referentes ao *offset* (deslocamento) do registro dentro do arquivo de registros. Dessa forma, cada entrada de índice corresponde à seguinte estrutura na linguagem C:

```
typedef struct {

    int32_t id;
    int32_t offset;

} EntradaIndice;
```

Como foi definido que a unidade mínima de leitura/escrita são blocos de 4096 bytes (capazes, portanto, de armazenar até 512 entradas de índice), o índice "topo" (esparso) deverá conter apenas entradas para a primeira entrada de cada bloco do índice "base" (denso). Dessa forma, o número de entradas no indice "topo" será (aproximadamente) 512 vezes menor do que o número de entradas no índice "base".

Note que que criar um índice "topo" com um número maior de entradas não irá ajudar a reduzir o número de leituras de blocos durante uma busca, além resultar em um índice que ocupa mais espaço. Note também que reduzir o número de entradas no índice "topo", não irá produzir benefício nenhum pois, embora resulte em um índice que ocupe menos espaço, irá aumentar o número de leituras de blocos durante uma busca.

Embora os arquivo de registros e o índices "base" considerados para este EP possam ocupar mais do que 2000 blocos de 4096 bytes, inviabilizando a carga total deles em memória RAM, vocês podem assumir que o índice "topo" sempre caberá integralmente em memória RAM.

3 Experimentos e relatório

Além da implementação dos algoritmos de busca e ordenação externa, vocês também deverão realizar alguns experimentos práticos e redigir um relatório a partir dos resultados obtidos. Um roteiro mais detalhado sobre como os experimentos deverão ser conduzidos será disponibilizado em breve.

4 Prazos e entrega

A entrega do EP deve ser feita até o dia 18/01/2014 às 23:55 pelo TIDIA-Ae. Entregue um arquivo zip contendo: os arquivos fontes do EP, um arquivo README explicando como compilar e rodar seu EP, e o relatório em formato PDF. Este EP pode ser feito em grupos de até 3 pessoas.

Boa diversão!