Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2013

Trabalho Prático 3 - Ordenação Externa

1 Introdução

Durante uma semana o assunto mais falado entre os amigos do Zambis foi sua festa. De tão bem falada, muitos de seus amigos enviaram para ele vários e-mails, cartas e telegramas. Zambis não quer deixar de responder nenhuma mensagem que recebeu, porém por serem muitas, ele decide que a forma mais justa é distribuir um número para cada um de seus amigos e ir respondendo as mensagens em ordem crescente desses números. Ou seja, o amigo que recebeu o número 1 vai ser respondido primeiro, o que recebeu o número 2 vai ser o segundo e assim sucessivamente.

O Zambis, quer fazer essa ordenação sem precisar perder muito tempo. Para isso ele decide que vai pedir sua ajuda para desenvolver um algoritmo que faça esse trabalho para ele. Porém por ele ter recebido muitas mensagens é impossível de se ordenar utilizando somente a memória principal de seu computador. Sendo assim, a única forma de se realizar esse trabalho é a através da ordenação externa. Além disso, o Zambis pediu para que essa ordenação seja feita através da intercalação balanceada de vários caminhos utilizando a técnica de seleção por substituição na etapa de geraçáo dos blocos ordenados e que seja utilizado o método 2f fitas.

2 Entrada

A entrada será um arquivo contendo uma grande quantidade de números. O tamanho do arquivo pode chegar ao tamanho de 1GB. Cada número estará em uma linha e você deve ordenar todos os números. O número 0\n pode ser utilizado como marcador do fim de bloco na etapa de interpolação. O último bloco terminará com o fim de arquivo. Lembre-se de que pode haver números repetidos, já que um amigo pode ter enviado mais de uma mensagem para Zambis. Considere que os números são inteiros positivos e o seu valor máximo é 4,294,967,296 - representação de um int de 32 bits.

A seguir um pequeno exemplo de entrada:

```
3343
13
241
54
2324
3
```

3 Saída

A saída deve ser um arquivo contendo os mesmos valores do arquivo de entrada porém com os valores ordenados.

Veja a saída para o exemplo anterior:

3
13
54
241
2324
3343

4 Limite de memória

Um dos desafios desse trabalho consiste em utilizar um tamanho de memória limitado para realizar a ordenação de um arquivo que não pode ser carregado por completo na mesma. Um dos parâmetros para execução do seu programa será o limite de memória que ele poderá utilizar. Esses limites **devem ser respeitados**. Existem ferramentas para medição do uso de memória controlar a quantidade de memória alocada para determinado programa durante sua execução. Vide (https://github.com/pshved/timeout) e (http://www.ss64.com /bash/ulimit.html), por exemplo. Utilize essas ferramentas durante seus testes.

Cada fita deverá ser simulada como um arquivo com nome f[número].

5 Parâmetros de entrada

O seu executável deverá **obrigatoriamente** ser capaz de receber o nome de um arquivo de entrada - de onde serão lidas as instâncias do problema - e o nome de um arquivo de saída - onde serão gravadas as soluções. Além disso, o terceiro parâmetro refere-se ao número de fitas utilizado para ordenação externa, seguido pelo tamanho da memória disponível para seu programa (dado em bytes). O comando para executá-lo deverá seguir o exemplo abaixo:

/.tp3 input.txt output.txt 21 256

Não teremos casos em que o número de fitas ultrapasse o tamanho da memória.

6 Entrega

- A data de entrega desse trabalho é **30/10/13**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <número matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.

- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixo devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua **nenhuma** pasta. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

7 Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução** proposta para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem signicavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

8 Código

O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.

- O utilitário **make** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido. O utilitário deverá gerar um executável com o nome **tp3**.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)

- **Variáveis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanto siga rigorosamente **os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- **Legibilidade e boas práticas** de programação serão avaliadas