MC202GH - ESTRUTURAS DE DADOS

2º Semestre de 2017

Professor: Rafael C. S. Schouery

Monitores: Yulle Glebbyo Felipe Borges (PED)

Caio Vinícius Piologo Véras Fernandes (PAD)

Lab. 06 - Intercalação de Múltiplos Vetores Ordenados

Peso 3

1. O Problema

Dados **n** vetores ordenados, cada um de tamanho $\mathbf{m_i}$ onde **i** representa o i-ésimo vetor, o objetivo é escrever um algoritmo que retorne um vetor ordenado com todos os elementos de cada um dos vetores, ou seja, um algoritmo que faça a intercalação dos vetores de forma a manter a propriedade de ordenação.

Considerando **M**, a quantidade total de elementos (em todos os vetores), essa tarefa poderia ser resolvida trivialmente através da concatenação dos vetores (que pode ser feita em O(**M**)) seguida da execução de um algoritmo rápido de ordenação, como o Mergesort (O(**M** log **M**)). Entretanto, podemos aproveitar a propriedade de que cada vetor já está ordenado para escrever um algoritmo que tenha tempo de execução assintótico melhor que a solução trivial.

Para isso, podemos utilizar a estrutura de heap de mínimo. Construímos um heap, no qual cada nó possui um dos vetores de entrada, e sua chave é o primeiro elemento deste vetor. Assim, após inserir todos os vetores no heap, a raiz do heap terá o vetor com o menor primeiro elemento. Podemos, repetidamente, remover o primeiro elemento do vetor na raiz do heap, imprimindo-o, e então corrigir a posição deste vetor no heap considerando seu novo primeiro elemento (que anteriormente era o segundo elemento). Quando o heap estiver vazio, todos os elementos estarão na saída e a solução estará completa. Se implementado cuidadosamente, este algoritmo executará em O(**M** log **n**).

2. Entrada

A primeira parte da entrada é o número de vetores **n**. Em seguida, cada vetor é fornecido um por um na forma tamanho do vetor seguido dos elementos do vetor na linha seguinte. Desta forma temos uma entrada genérica:

n
m_1
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ … $x_{1,m1}$
m_2
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ … $x_{2,m2}$
…
m_n
$x_{n,1}$ $x_{n,2}$ $x_{n,3}$ … $x_{n,mn}$

onde **n** é a quantidade de vetores; **m_i** é o tamanho do vetor **i**; **$x_{i,j}$** é o **j**-ésimo elemento do **i**-ésimo vetor.

Exemplo de entrada:

```
5
4
1 4 10 11
3
2 3 4
5
-8 -4 5 7 11
2
8 10
5
-7 6 8 9 12
```

3. Saída

Na saída, temos a sequência ordenada, contendo todos os elementos de todos os vetores. Os elementos devem ser separados por um espaço em branco e ao fim da sequência deve-se incluir um **\n**. Obs: **Deve** existir um espaço em branco **após o último elemento**, antes do **\n**.

```
-8 -7 -4 1 2 3 4 5 6 7 8 8 9 10 10 11 11 12
```

4. Informações

- Este laboratório possui **peso 3**.
- **Não** há um número máximo de submissões.

- No início de cada arquivo a ser submetido, insira um comentário com seu nome, RA e uma breve descrição do conteúdo do arquivo.
- Apenas comentários no formato `/* comentário */` serão aceitos. Comentários com `//` serão acusados como erros pelo SuSy.
- Os cabeçalhos permitidos são:
    - **`stdio.h`**
    - **`stdlib.h`**
- A submissão da sua solução deverá conter múltiplos arquivos:
    - **heap.h**: interface do heap
    - **heap.c**: implementação do heap
    - **lab06.c**: cliente que utiliza a estrutura
- Será disponibilizado um **Makefile** para este trabalho na página do laboratório:
    - Para compilar seu projeto, basta navegar até o diretório do projeto e utilizar o comando ‘make’ no terminal do Linux.
    - Veja mais instruções na página da disciplina.
- Toda a memória alocada deve ser liberada adequadamente ao fim do programa. Isso pode te ajudar a evitar a ocorrência de falhas relacionadas a memória.
- Indente corretamente todo o seu código. Escolha entre espaços ou tabs para indentação e seja coerente em todos os arquivos. Indentação é fundamental para a legibilidade do seu código, e ajuda no entendimento do fluxo de controle do seu programa.

5. Critérios de Avaliação

- Seu código deverá passar por todos os casos de teste do SuSy definidos para este laboratório. Caso positivo, utilizaremos os seguintes critérios adicionais:
    - A **não utilização da estrutura heap** para a solução do problema proposto resultará em **nota zero**
    - Falha na implementação do comportamento esperado para o heap acarretará em uma penalização

- Implementações que executem em tempo pior que **O(M log n)** serão penalizadas
- Utilização de bibliotecas ou funções não permitidas pelo enunciado resultará em uma **penalização severa** na nota