SCC0220 - Laboratório de Introdução à Ciência da Computação II

Prof. Jean R. Ponciano
Estagiário PAE: João Victor C. N. de Sousa
Monitores: Matheus Vieira Fernandes e Fernando Valentim Torres

Departamento de Ciências de Computação (SCC)
Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo

Entregável 03 – Volta USP (Parte 2)

Com as inscrições concluídas, o CEFER precisa organizar a ordem de largada da Volta USP São Carlos. Para isso, os inscritos de cada comunidade devem ser ordenados pelo tamanho dos seus nomes (número de caracteres, desconsiderando espaços em branco).

O problema é que a equipe de organização não sabe nada sobre algoritmos de ordenação — eles apenas entendem a ideia de que as “bolhas maiores sobem em um copo de refrigerante”. É aí que entra você: sua missão é implementar o algoritmo *Bubble Sort*, que imita esse comportamento, para ordenar os inscritos de cada comunidade.

Entrada

- Cada linha da entrada contém o nome de um inscrito seguido por " - usp" ou " - externa".
- O nome não precisa ser real: podem ser usadas palavras aleatórias ou strings artificiais, como por exemplo *yyjfvm*.
- A entrada termina em EOF.

Saída

- O programa deve imprimir duas seções, uma para a comunidade USP e outra para a comunidade Externa, no seguinte formato:

```
USP - [t1, t2, t3, ...]
Comparações: X1, Trocas: Y1

Externa - [t1, t2, t3, ...]
Comparações: X2, Trocas: Y2
```

- Onde:
    - t_i é o tamanho do nome (número de caracteres, sem contar espaços).
    - **X** é o número total de comparações realizadas pelo algoritmo.
    - **Y** é o número total de trocas realizadas pelo algoritmo.

Exemplo de entrada

```
aline - externa
thiago - usp
joao - usp
enrique - externa
```

Exemplo de saída

```
USP - [4, 6]
Comparações: 1, Trocas: 0

Externa - [5, 7]
Comparações: 1, Trocas: 0
```

Submissões:

- Run.codes: versão do programa que realiza a ordenação por Bubble Sort.
- E-disciplinas: relatório em PDF (até 3 páginas), contendo:
    - Descrição sucinta do funcionamento do algoritmo Bubble Sort.
    - Código da implementação da sua solução.
    - Análise crítica de desempenho do algoritmo, discutindo:
        - Tempo de execução medido no Run.codes (CPU time).
        - Número de comparações e trocas.
        - Três cenários distintos:
            - Melhor caso: lista já ordenada.
            - Pior caso: lista ordenada de forma inversa.
            - Caso médio: lista em ordem aleatória.
        - Discussão sobre simplicidade do código e uso de funções auxiliares.

Casos de análise de desempenho (explicação detalhada):

- Melhor caso (já ordenado):
  Quando a lista de nomes já está em ordem crescente de tamanho.
  Exemplo: `[2, 3, 4, 5]`.
  Nesse cenário, o Bubble Sort apenas verifica que não há trocas a serem feitas, realizando o menor número possível de operações.

- Pior caso (ordenado inverso):
  Quando a lista de nomes está em ordem totalmente contrária ao esperado.
  Exemplo: `[5, 4, 3, 2]`.
  Aqui o Bubble Sort precisará percorrer a lista várias vezes, realizando o máximo de trocas possíveis, o que representa o maior custo de execução.

- Caso médio (ordem aleatória):
  Quando a lista de nomes está em uma ordem mista, sem padrão específico.
  Exemplo: `[3, 1, 4, 2]`.
  Nesse caso, o desempenho esperado fica entre o melhor e o pior caso, já que haverá algumas trocas e comparações, mas não o máximo possível.

Prazo: até dia 25/09.

Atenção: Caso haja suspeita de uso de IA na implementação ou na escrita do relatório, o professor poderá requisitar apresentação e arguição sobre o que foi submetido. Neste caso, a nota (individual) será atribuída a partir do desempenho obtido na apresentação/arguição.

Dicas:

- Ao implementar o Bubble Sort, contabilize corretamente o número de comparações e trocas realizadas.
- Teste seu programa em diferentes cenários:
  - Melhor caso (já ordenado).
  - Pior caso (ordenado de forma inversa).
  - Caso médio (nomes em ordem aleatória).