Universidade do Estado de Santa Catarina
Departamento de Ciência da Computação
Ciência da Computação - Complexidade de Algoritmos

Trabalho I – Algoritmos de Ordenação

(1) Escreva os algoritmos: **Bubble Sort**, **Insert Sort**, **Merge Sort, Quick Sort 1** (primeiro elemento fixo como pivô), **Quick Sort 2** (pivô escolhido aleatoriamente), **Heap Sort, Counting Sort** e **Bucket Sort**, para ordenar um vetor de *n* elementos numéricos (OBS: utilize inteiros de 64 bits, e.g. long long).

(2) Gere arquivos de entrada para testar o tempo de execução dos algoritmos implementados. Deverão ser criados arquivos de entrada com 25.000, 50.000, 75.000, 100.000 e 1.000.000 de elementos, com valores numéricos entre 0 e o tamanho do vetor. Todavia, deverão ser criadas quatro versões para cada tamanho de entrada: (1) em ordem crescente, (2) em ordem decrescente, (3) em ordem aleatória e (4) em ordem aleatória, mas com um elemento 100.000.000 (em uma posição aleatória do vetor).

(3) Teste os 20 tipos de entradas diferentes para cada algoritmo de ordenação implementado e escreva um relatório comparativo apresentando os tempos de execução. Caso um algoritmo demore mais de 5 minutos para executar, o teste pode/deve ser interrompido, registrando-se no relatório como: "> 5 minutos". O relatório deve comparar o comportamento de cada algoritmo individualmente para cada tipo de entrada e comparar o comportamento entre os diferentes algoritmos (por exemplo, quais foram melhores/piores algoritmos para um vetor já ordenado? E para um vetor aleatório? E se existir um elemento muito grande?)

DICAS:

+ Crie os vetores como variáveis globais (devido ao tamanho da entrada);
+ A pilha de chamadas recursivas pode estourar dependendo do S.O. e das configurações de compilação. Pode ser necessário aumentar o tamanho da pilha (*stack*);
+ Cuidado ao implementar o **bucket sort**! Muitas versões online implementam o **counting sort** e dizem que é o **bucket sort**;
+ Seu relatório deve fazer uma relação entre as complexidades dos algoritmos (tempo e espaço) e seus tempos de execução mensurados na prática;
+ Para o ***bucket sort***, tente realizar variações com diferentes tamanhos de *buckets* e não esqueça de explicitar qual o algoritmo de ordenação utilizado para cada balde.

Linguagens permitidas: C, C++, Java, Python.

Material a ser entregue: relatório + códigos das implementações

Restrições:

- O relatório deverá ter no **máximo** 8 páginas (sem um modelo pré-definido)
- O relatório deverá ser entregue em formato pdf
- Este trabalho poderá ser realizado no máximo em duplas
- Este trabalho poderá ser entregue no máximo até o dia: **21/09/2018**