Universidade Federal de Campina Grande
Centro de Engenharia Eletrica e Informática
Departamento de Sistemas e Computação
Graduação em Ciência da Computação

Exercícios sobre Algoritmos de Ordenação por Comparação (Dividir para Conquistar)

Objetivo: Praticar a implementacao de algoritmos de ordenação por comparação que utilizam a técnica de dividir para conquistar.

O endereço do sistema de submissão é o **https://les.dsc.ufcg.edu.br:8443/EasyLabCorrection**.

Atividades necessárias antes de iniciar o exercicio:

1. Crie um projeto no Eplipse chamado LEDA, por exemplo (pode ser qualquer outro nome que lhe convier);
2. Descompacte o arquivo baixado (exceto o PDF) na pasta dos fontes (normalmente **src**) do seu projeto LEDA criado no seu workspace. O arquivo baixado tem a seguinte estrutura:
   -sorting
   --Sorting.java (INTERFACE CONTENDO A ASSINATURA DO METODO DE ORDENAÇÃO)
   --SortingImpl.java (IMPLEMENTAÇÃO BASE A SER HERDADA POR TODAS AS IMPLEMENTAÇÕES)
   -- Util.java (CLASSE AUXILIAR CONTENDO O METODO DE SWAP A SER USADO NAS IMPLEMENTACOES)
   --divideAndConquer
   --- Mergesort.java (IMPLEMENTACAO A SER PREENCHIDA PELO ALUNO)
   --- Quicksort.java (IMPLEMENTACAO A SER PREENCHIDA PELO ALUNO)
   --- hybridMergesort
   ---- HybridMergesort.java (IMPLEMENTACAO A SER PREENCHIDA PELO ALUNO)
3. No Eclipse, selecione a pasta dos fontes no projeto LEDA e faça um refresh (apertar F5). Note que deve aparecer a estrutura de pacotes e os arquivos mencionados acima.

Obs: NÃO modifique a assinatura dos métodos. Caso contrário os testes não funcionarão.

Agora voce está pronto para começar a trabalhar nas seguintes atividades:

1. Implemente o método sort(T[] array, int leftIndex, int rightIndex) nas classes Mergesort.java e Quicksort.java.

Observe a classe HybridMergesort.java. Ela representa a implementação de uma variação do mergesort que pode fazer uso do insertion sort (um algoritmo híbrido) da sequinte forma: o mergesort é aplicado a entradas maiores a um determinado limite (SIZE_LIMITE). Caso a entrada tenha tamanho menor ou igual ao limite o algoritmo usa o insertion sort. A implementação híbrida deve considerar os seguintes detalhes:

- Ter contadores das quantidades de mergesorts e insertion sorts aplicados, de forma que essa informação possa ser capturada pelo teste.
- A cada chamado do método de sort(T[] array), esses contadores são resetados.
- O algoritmo híbrido deve ser in-place.

Instruções para o envio

Ao terminar o exercício, você precisa enviar apenas os arquivos que voce implementou, compactados, seguindo a mesma estruturas de pacotes do original. Ou seja, seu arquivo compactado deve ter a seguinte estrutura:

-sorting
--divideAndConquer
--- Mergesort.java
--- Quicksort.java
--- hybridMergesort
---- HybridMergesort.java

Obs: a compactação DEVE ser feita a partir do diretório raiz de seus fontes de forma a preservar a estrutura de pastas que refletem a estrutura dos pacotes (package) Java. Uma boa dica é compactar a pasta “sorting” e depois remover os arquivos que nao devem ser enviados. Certifique-se de que seu arquivo compactado tem a estrutura de pacotes requerida antes de envia-lo. Se suas classes estiverem empacotadas (com informação de package) e você compactá-las diretamente sem a estrutura de pastas correta, seu código não vai compilar. Modifique o nome do arquivo compactado para NOME_COMPLETO_DO_ALUNO.ZIP.

Observações finais:

- **A interpretação do exercício faz parte do roteiro.**
- **O roteiro é individual. É como se fosse uma prova prática e a conversa entre alunos é proibida.**
- **É proibido coletar códigos prontos e adaptar. Implemente as questões. Isso é para seu aprendizado.**
- **Caso você observe qualquer problema no sistema de submissão, contacte o professor imediatamente.**