Ifes Campus Serra
BSI – Bacharelado de Sistemas de Informação
Programação II

Avaliação Pontos Extras: TBO e Recursão

Total de pontos: 15 pontos

Regras para a prova: utilize apenas os comandos python vistos em sala de aula // para manipulação de arquivos, utilizar apenas open, close, readline, write // não utilizar expressões de lista // não utilizar dados com tipos // na construção da prova, utilizar apenas os arquivos de dados fornecidos pelo Professor, SEM modificações // para construir as respostas, utilize os nomes de arquivos .py e nomes de funções conforme pedidos nos enunciados // todas as figuras exibem apenas *exemplos* para auxiliar na interpretação do enunciado // celulares, pendrives, não são permitidos durante a prova.

O Professor pode solicitar que o aluno explique a solução da prova.

A prova é indivual .

Questão 1: Implementação do merge sort (8 pts)

Pesquise por 2 versões do algoritmo **merge sort**, uma versão recursiva e uma versão iterativa (não recursiva).

Escreva a versão **não recursiva** em um arquivo chamado **mergesort-nr.py**. Escreva a versão **recursiva** em um arquivo chamado **mergesort-r.py**. **Não altere esses nomes.**

a) Altere as duas versões da função merge sort para incluir a função callback de comparação nos moldes do que foi discutido e implementado em aula para as funções de sort: bolha, selection e insertion. Inclua a função callback como último parâmetro nas funções de merge sort.

Analise o código das 2 versões do merge sort e descubra qual trecho do código original deve ser substituído pela chamada da função callback de comparação.

b) Construa uma aplicação em python chamada **apmerge.py**. A aplicação deve implementar o seguinte algoritmo: **i)** importar os arquivos mergesort-r.py e mergesort-nr.py; **ii)** definir uma função compara que viabiliza a classificação em ordem decrescente de estado, crescente de municipio e descrescente de cep; **iii)** ler o arquivo **bdcepsruas.txt** para uma lista de listas onde as listas internas armazenam cada componente da linha do arquivo (entre vírgulas) como um elemento separado da lista (**ver figura 1**); **iv)** classificar a lista do item **(iii)** invocando a versão não recursiva da função merge sort (invocada com a função compara, item **(ii)**); **v)** salvar a lista ordenada em **(iv)** em um arquivo chamado **bdmergesort-nr.txt** (não altere esse nome); **vi)** repetir os passos **(iv)** e **(v)** para a versão recursiva da função merge sort; **vii)** salvar a lista ordenada em um arquivo chamado **bdmergesort-r.txt** (não altere esse nome); **viii)** fim.

IMPORTANTE

Se o tamanho da lista produzir erro na execução da versão recursiva do merge sort (stack overflow), então crie uma versão reduzida do arquivo para uso apenas no merge sort recursivo.

```
[[Praça Anníbal Anthero Martins, no. 90, 13419-564, Tabuazeiro, Manguinhos, PA],
 [Praça Anita Francisca Crystello, no. 21, 26517-482, do Moscoso, Colatina, AL],
 [Rua Anníbal Vieira Rabayolli, no. 14, 21389-401, Caratoíra, Muniz Freire, PB],
 [Escadaria Antenor Passos, no. 6, 15895-036, Caratoíra, Viana, DF],
 [Escadaria Antônia Pereira Paiva, no. 82, 21389-401, Jardim da Penha, Cachoeiro, AC]]
```

Figura 1: apenas um exemplo de um trecho da lista

ATENÇÃO (NOTA DA QUESTÃO)

Esta questão somente estará correta se:

- as versões do merge sort forem alteradas corretamente e fizerem a classificação do arquivo bdcepsruas.txt conforme a ordem solicitada;
- a função compara for corretamente construída.

Questão 2: Implementação do quick sort (8 pts)

Pesquise por 2 versões do algoritmo **quick sort,** uma versão recursiva e uma versão iterativa (não recursiva).

Escreva a versão não recursiva em um arquivo chamado **quicksort-nr.py**. Escreva a versão recursiva em um arquivo chamado **quicksort-r.py**. **Não altere esses nomes.**

a) Altere as duas versões da função quick sort para incluir a função callback de comparação nos moldes do que foi discutido e implementado em aula para as funções de sort: bolha, selection e insertion. Inclua a função callback como último parâmetro nas funções de quick sort.

Analise o código das 2 versões do quick sort e descubra qual trecho do código original deve ser substituído pela chamada da função callback de comparação.

b) Construa uma aplicação em python chamada **apquick.py**. A aplicação deve implementar o seguinte algoritmo: **i)** importar os arquivos quicksort-r.py e quicksort-nr.py; **ii)** definir uma função compara que viabiliza a classificação em ordem decrescente de estado, crescente de municipio e descrescente de cep; **iii)** ler o arquivo **bdcepsruas.txt** para uma lista de listas onde as listas internas armazenam cada componente da linha do arquivo (entre vírgulas) como um elemento separado da lista (**ver figura 1**); **iv)** classificar a lista do item **(iii)** invocando a versão não recursiva da função quick sort (invocada com a função compara, item **(ii)**); **v)** salvar a lista ordenada em **(iv)** em um arquivo chamado **bdquicksort-nr.txt** (não altere esse nome); **vi)** repetir os passos **(iv)** e **(v)** para a versão recursiva da função quick sort; **vii)** salvar a lista ordenada em um arquivo chamado **bdquicksort-r.txt** (não altere esse nome); **viii)** fim.

IMPORTANTE

Se o tamanho da lista produzir erro na execução da versão recursiva do quick sort (stack overflow), então crie uma versão reduzida do arquivo para uso apenas no quick sort recursivo.

ATENÇÃO (NOTA DA QUESTÃO)

Esta questão somente estará correta se:

- as versões do quick sort forem alteradas corretamente e fizerem a classificação do arquivo bdcepsruas.txt conforme a ordem solicitada;
- a função compara for corretamente construída.

Material para a Atividade

Como recurso para a prova, está sendo entregue o arquivo **para-aval-pontos-extra.zip** contendo os seguintes arquivos: prog2-pontos-extra.pdf (enunciado), bdcepsruas.txt (base de dados para ordenação).

Entrega

Compacte os arquivos **mergesort-nr.py**, **mergesort-r.py, apmerge.py, quicksort-nr.py, quicksort-r.py, apquick.py, bdcepsruas.txt, curvassort.py** em arquivo chamado **prog2-p3.zip**.

Compactações em formato rar não serão corrigidas.

Entregue o arquivo compactado na tarefa do ava aberta para a avaliação.

Aviso importante

Somente serão corrigidas as soluções entregues via tarefa do ava associada à avaliação. Soluções não entregues, ou entregues via outros mecanismos, receberão a nota zero.

Caso não consiga fazer a entrega da tarefa devido a questões técnicas como falta de conexão com internet ou servidor ava fora do ar, registre a causa do impedimento e **procure o Professor no dia seguinte a entrega.** Compareça levando o arquivo **prog2-pontos-extra.zip** e providencie o devido envio para a tarefa do ava.

Não seguir o procedimento do parágrafo anterior implicará em nota zero para a avaliação.

Nomes da funções e arquivos

Os códigos devem respeitar TODOS os nomes de funções e arquivos mencionados no enunciado.

Bons estudos!