Fundação CECIERJ
Tecnologia em Sistemas de Computação
Programação com Interfaces Gráficas
AD1 2^o semestre de 2018
Gabarito

Professores: Mario Benevides e Paulo Roma

Resumo

Todos nós já vimos classificações (rankings) de algum tipo, sejam de faculdades, telefones inteligentes ou comerciais de Futebol. Os motores de busca produzem rotineiramente, sob demanda, classificações e recomendações de hotéis, restaurantes ou eletrodomésticos de acordo com uma variedade de critérios. O trabalho consiste em implementar algoritmo para calcular a distância entre classificações seguindo as diretrizes estabelecidas no enunciado da AD1.

1 Calculando Distâncias entre Classificações

Os principais métodos serão listados aqui, na AD2 deverá ser entrege o que não foi implementado na AD1 além da interface gráfica para o programa.

```
def sortAndCount(self, lista):
    """
    sortAndCount ordena e conta as inversões da lista.
    @param lista: um ranking.
    @return: Retorna a soma das inversões e a lista.
    """
    if len(lista) == 1:
        return 0, lista
    else:
        m = len(lista) // 2
        LEsquerda = lista[:m]
        LDireita = lista[m:]
        invEsq, LEsquerdaOrd = self.sortAndCount(LEsquerda)
        invDir, LDireitaOrd = self.sortAndCount(LDireita)
        invLista, LCompleta = self.mergeAndCount(LEsquerdaOrd, LDireitaOrd)
    return (invEsq + invDir + invLista), LCompleta

def mergeAndCount(self, Lalpha, Lbeta):
    """
    Combinando e contando inversões entre duas sequências, recebe duas
    listas ordenadas.
    Retorna a quantidade de inversões.
    @param Lalpha lista.
    @param Lbeta lista.
    @return: quantidade de inversões e uma lista concatenada
    """
    i = j = count = 0
    Lgama = []
    while i < len(Lalpha) and j < len(Lbeta):
        if Lalpha[i] < Lbeta[j]:
            Lgama.append(Lalpha[i])
            i = i + 1
        else:
```

```
            Lgama.append(Lbeta[j])
            count = count + (len(Lalpha) - i)
            j = j + 1
    if i >= len(Lalpha):
        Lgama = Lgama + Lbeta[j:]
    if j >= len(Lbeta):
        Lgama = Lgama + Lalpha[i:]
    return count, Lgama

def getNumItems(self):
    """
    Retorna o número de itens do ranking.
    @return: numero de itens do ranking
    """
    return len(self.ranking)

def getRank(self, i):
    """
    Retorna o rank do item i.
    Levanta exceção ValueError se i não estiver no ranking.

    @param i: um elemento do ranking
    @return: o rank do elemento i
    """
    if i in ranking:
        return self.ranking[i-1]
    else:
        raise ValueError("Item não está no ranking.")

def footrule(self, r1, r2):
    """
    Retorna a distância footrule entre r1 e r2.
    Levanta uma exceção TypeError se r1 ou r2 forem None.
    Uma exceção ValueError se r1 e r2 tiverem comprimentos diferentes.
    @param r1: um ranking do tipo lista
    @param r2: um ranking do tipo lista
    @return: distância footrule entre os dois ranking
    """
    if r1 == None or r2 == None:
        raise TypeError("Ranking não pode ser igual a None.")
    elif len(r1) != len(r2):
        raise ValueError("r1 e r2 tem tamanhos diferentes.")
    else:
        count = 0
        for i in range(len(r1)):
            count = count + abs(r1[i] - r2[i])
        return count
```

2 Testes

Na AD1, os testes podem ser relativamente simples. Já na AD2, será cobrado um conjunto completo de testes, mais a confecção de uma interface gráfica para o programa.