University of Augsburg, Institute of Computer Science
Prof. Dr. P. Fischer
J. Kastner, L. Rudenko

SS 2019
06. May. 2019
Solution 1

Analyzing Massive Data Sets

Exercise 1: Bonferroni's Principle (live)

The solution was discussed in the exercise.

Exercise 2: Python - 1 (homework)

Only the standard Python is used in this exercise, no special libraries.

a) **Merge Sort**:

```
#merge function
def merge(arr1, arr2):
    result = []
    while len(arr1) != 0 and len(arr2) != 0:
        if arr1[0] < arr2[0]:
            result.append(arr1[0])
            arr1.remove(arr1[0])
        else:
            result.append(arr2[0])
            arr2.remove(arr2[0])
    if len(arr1) == 0:
        result += arr2
    else:
        result += arr1
    return result

#Code for merge sort
def mergesort(arr):
 if len(arr) <= 1:
   return arr

 middle = len(arr) // 2
 left = mergesort(arr[:middle])
 right = mergesort(arr[middle:])

 return merge(left, right)

#test
print(mergesort([3,6,8,10,1,2,1]))
```

b) **Depth-first Search**:

```
def deepSearch(graph, start, destination, visitedNodes = []):
    if start == destination:
        return True
    if graph[start]:
        for node in filter(lambda x: x not in visitedNodes,
        graph[start]):
            visitedNodes.append(node)
            if deepSearch(graph, node, destination, visitedNodes):
                return True
            else:
                visitedNodes.append('Node ' + str(node) +
                ' is finished, no path from ' + str(node) +
                ' to ' + str(destination))
    return False

adjList = {0:[1,2], 1:[2,3], 2:[4], 3:[4,5], 4:[], 5:[]}
visitedNodes = []
print(deepSearch(adjList, 0, 5, visitedNodes))
print(visitedNodes)
```

c) **Breadth-first Search**:

```
import queue as q
def breitensuche(adj, start, destination, visitedNodes = []):
    queue = q.Queue()
    queue.put(start)
    while queue.qsize() > 0:
        currentNode = queue.get()
        visitedNodes.append(currentNode)
        for successor in adj[currentNode]:
            if successor in visitedNodes:
                visitedNodes.append('Node ' + str(successor) +
                'was already visited earlier.')
                continue
            elif successor == destination:
                visitedNodes.append(successor)
                return True
            else:
                queue.put(successor)
    return False

adjList = {0:[1,2], 1:[2,3], 2:[4], 3:[4,5], 4:[], 5:[]}
visitedNodes = []
print(breitensuche(adjList, 0, 5, visitedNodes))
print(visitedNodes)
```

Exercise 3: Python - Pandas (live)

a) Find 25 suppliers with the lowest account balance.

```
import pandas as pd
supplier = pd.read_csv('correct_path/supplier.tbl', sep='|',
names=['s_Id','s_Name','Address','Nationkey','Phone','Acctbal','s_Comment','Dummy'])

# supplier.drop("Dummy", axis=1, inplace=True)
# supplier.drop(supplier.columns[len(supplier.columns)-1], axis=1, inplace=True)
supplier.drop(supplier.columns[-1], axis=1, inplace=True)

# top_25 = (supplier.sort_values(by = 'Acctbal')[['s_Name', 'Acctbal']]).head(25)
top_25 = supplier.nsmallest(25, 'Acctbal')[['s_Name', 'Acctbal']]

print(top_25)
```

b) How many suppliers have a positive account balance?

```
# supplier_pos = supplier[supplier['Acctbal'] > 0.0][['s_Name']].aggregate('count')
supplier_pos_bal = supplier[supplier['Acctbal'] > 0.0][['s_Name']].count()

print(supplier_pos)
```

or

```
supplier_pos_bal = supplier[supplier['Acctbal'] > 0.0]

print(supplier_pos_bal.shape[0])
```

c) Find out all brands produced by the same manufacturer and calculate the items number and the total sales price for each brand of each manufacturer.

```
part = pd.read_csv('correct_path/part.tbl', sep='|', names=['p_Id','p_Name',
'Mfgr','Brand','Type','Size','Container','Retailprice','p_Comment','Dummy'])
part.drop(part.columns[-1], axis=1, inplace=True)

brand_manufacturer = part.groupby(['Mfgr', 'Brand']).agg({'p_Name': 'count',
'Retailprice': 'sum'})

print(brand_manufacturer)
```

d) How many items have 3 words in their name?

```
partNameSeries = part['p_Name']

print(partNameSeries[(partNameSeries.str.split()).map(len) == 3].aggregate('count'))
```

or

```
partNameSeries = part['p_Name']

print(np.sum(partNameSeries.str.split().map(len) == 3))
```

e) How many different items does each supplier have?

```
partsupp = pd.read_csv('correct_path/partsupp.tbl', sep='|',
names=['p_FK','s_FK','Availqty','Supplycost','Comment_partsupp','Dummy'])
partsupp.drop(partsupp.columns[-1], axis=1, inplace=True)

supplierPS = pd.merge(supplier, partsupp, left_on = 's_Id', right_on = 's_FK')
supplierPart = pd.merge(supplierPS, part, left_on = 'p_FK', right_on = 'p_Id')

print(supplierPart.groupby('s_Name')['p_Name'].count())
```