DATA1050 Final Exam

Released 10 December 2020, 13:00 ET.
Due 12 December, 13:00 ET

Name: _____Enmin Zhou_____

Banner ID: _____B01694721_____

Brown Short Name: _____ezhou24_____

Date: _____Dec 12 2020_____

Setup

- The Course Staff will be on Zoom and in Prismia from 1-2 pm ET to provide clarifications.
- All clarifications will be added to the exam handout in green and red. After that please use private posts to Piazza.
- No question clarifications will be added after 1 pm on Friday.

Examination Rules

- Use the provided boxes for your answers to make clear which part should be graded, otherwise we won't know what to grade.
- All work must be entirely your own.
- **You must work incrementally in a copy of this google document. No credit if we can not see your solo progress.**
- The resources listed on the following page are the only resources allowed for this exam. No open googling. No discussion with 3rd parties.

Brown Short Name:_____

Allowed Resources

This is an open book exam, not an open google exam. You may use the resources previously allowed for the midterm, PG Exercises, the suggested SQL books, and the learning materials suggested for MongoDB and Databricks, and other materials on those sites.

1) Python Basics, Searching, TDD

In this section, we give you code to web scrape Shakespeare's Othello and ask you to perform a series of tasks on the data. Before you begin, ensure you have both Requests and Beautiful Soup installed. You will also need to figure out the correct import statements. **(Remember this is an exam, so no posting to Piazza or asking your friends on how to do this.)**

For the curious: Requests is a simple library that allows you to pull html documents from the internet (and make other API requests), and Beautiful Soup is a lightweight library that allows the user to manipulate HTML pages. To get a gist of what these libraries can do see this example

Please use the create a working version of the code below using the development environment of your choice that.

Requirements:

1) Create good docstrings for all functions you implement
2) Create clear and efficient solutions for those functions
3) Include the time and space complexity for those function
4) Include appropriate test cases for those functions

```
# Your code answer here:

#TODO add appropriate import statements

import requests
from bs4 import BeautifulSoup

# TODO add appropriate import statements

page = requests.get('http://shakespeare.mit.edu/othello/full.html')
html = page.text

soup = BeautifulSoup(html, 'html.parser')


def clean_page(html):
    # convert the page.text into a BeautifulSoup object
    soup = BeautifulSoup(html, 'html.parser')
    # navigate to the HTML body node (look at calls to .body)
```

```
    body = soup.body
    # remove the header (note that it is wrapped in a table)
    table_tag = body.table
    table_tag.decompose()
    # now return the text from the body
    return body.get_text()


### Problem 1.1 All the Words
# 1. Make this function make a string of text into a list of words.
# 2. Have it take out the punctuation.
# 3. Have it make the words lowercase.
# 4. It must return the list
def lowercase_and_split(s):
    # time complexity: O(n), space : O(n)
    # lower all the string and use space to replace non-alpha characters and rejoin into a string
    lower_s = "".join([i.lower() if i.isalpha() else " " for i in s])
    # split the sentence into words based on the space because space separate words in english
    words = lower_s.split(" ")
    # remove the empty string because of the two space next to each other in the first process
    real_words = [w for w in words if w != '']
    return real_words


### Problem 1.2 Largest Words
# 1. This function should take a list of words.
# 2. It needs to return the n largest words.
# 3. When words are the same length they need to be returned in alphabetical order.
def largest_words(text, n=3):
    # space O(1), time O(n^2)
    # sort the text based on string length
    for i in range(len(text)):
        for j in range(i+1,len(text)):
            if len(text[i]) < len(text[j]):
                text[i], text[j] = text[j], text[i]
    # sort the text based on alphabetical order
```

```
    for i in range(len(text)):
        for j in range(i+1,len(text)):
            if len(text[i]) == len(text[j]) and text[i] > text[j]:
                text[i], text[j] = text[j], text[i]
    return text[:n]



### Problem 1.3 Most Common Words
# 1. Create a list of the n words that occur the most.
# 2. When words are the same length they need to be returned in alphabetical order.
def most_common_words(text, n=3):
    # time O(n^2), space O(n)
    count = {}
    # count the occurrence of words
    for i in text:
        if i in count.keys():
            count[i] = count[i] + 1
        else:
            count[i] = 1
    # sort the dict and return n most frequent words, this is O(nlogn)
    sorted_count = [(k,v) for k, v in sorted(count.items(), key=lambda item: item[1])]
    # append the top words into top list
    top = []
    i = 0
    while i < n:
        top.append(sorted_count[i])
        i += 1
    remaining = []
    # append more than n words if there are words have same occurrence in parallel
    if i < len(sorted_count) and sorted_count[i][1] == top[-1][1]:
        remaining = [k for k,v in sorted_count[i:] if v == top[-1][1]] + [k for k,v in top if
v == top[-1][1]]
    # take those last words in parallel out of top list
    top = [k for k,v in top if v > top[-1][1]]
    # use previous function to return the largest top words
    top = largest_words(top, len(top)) + largest_words(remaining, n - len(top))
    return largest_words(top, n)
```

```
### Problem 1.4 Find Words
# Your job to quickly find all occurrences of a set of words in a text stored as another
# list of words.  This function needs to do that, and should return a list of lists
# with the locations corresponding to each word that is to be found.
def find_words(words, text):
    # space: O(n), time: O(n)
    # build a dictionary to store the seen word locations if word is in words
    locations = {}
    # loop through the text and add word location into locations
    for i in range(len(text)):
        # check if the word is in words
        if text[i] in words:
            # check if we see the words before
            if text[i] in locations.keys():
                locations[text[i]] = locations[text[i]] + [i]
            else:
                locations[text[i]] = [i]
    # return the lists of list in order
    results = [locations[word] for word in words]
    return results


### Problem 1.5 Tokenization
# This function takes a list of words and a dictionary, and uses the dictionary to
# replace words with numbers using a token dictionary.  If a word is not found, add a
# new entry in the dictionary for it, and a unique number is assigned to it.
# When it is all done, this function returns both the tokenized list and the token
dictionary.
def tokenize(text, tokens=None):
    # space: O(n), time: O(n)
    # create tokens dict if none
    if tokens is None:
        tokens = {}
    # use an incre to set token for new words
    incre = 0
    # loop through the text
    for i in range(len(text)):
        # use token if we see the word before
```

```
        if text[i] in tokens.keys():
            text[i] = tokens[text[i]]
        # use incre if we haven't seen
        else:
            # find a new unique incre the the unseen word
            while incre in tokens.values():
                incre += 1
            tokens[text[i]] = incre
            text[i] = incre
            incre += 1
    return text

def test_lowercase_and_split():
    s = 'my name!! is Enm,in Zhou[]'
    assert lowercase_and_split(s) == ['my', 'name', 'is', 'enm', 'in', 'zhou']


def test_largest_words():
    s = ['my', 'name', 'is', 'enm', 'in', 'zhou']
    assert largest_words(s) == ['name', 'zhou', 'enm']

def test_most_common_words():
    s = ['my', 'name', 'is', 'enm', 'in', 'zhou']
    assert most_common_words(s) == ['name', 'zhou', 'enm']

def test_find_words():
    s = ['my', 'name', 'my', 'is', 'enm', 'zhou', 'in', 'zhou']
    assert find_words(['my', 'is', 'zhou'], s) == [[0, 2], [3], [5,7]]

def test_tokenize():
    s = ['my', 'name', 'my', 'is', 'enm', 'zhou', 'in', 'zhou']
    assert tokenize(s) == [0,1,0,2,3,4,5,4]

if __name__ == "__main__":
    text = clean_page(html)
    words = lowercase_and_split(text)
    test_lowercase_and_split()
    test_largest_words()
    test_most_common_words()
```

```
    test_find_words()
    test_tokenize()
```

Brown Short Name:_____

Problem 1.6 (Extra Credit) Tokenization Trade Offs

Explain how you could adjust the other functions above to utilize tokenization. One design consideration is how you would pass lists of tokens and the token dictionary around. Does it make sense to keep the two distinct or would a data structure that holds them both be helpful? Explain. Discuss the drawbacks and wins available when using a Tokenized text representation for each function -- include discussion of Time and Space Complexity.

Be specific and use complete sentences.

Your answer here:
In the previous functions, we will sort and compare based on the value of tokens if we use the tokenization. However, the part that counts the occurrences will not change since tokenization does not have an impact on that. When we build the token dictionary, the key will be the tokens and the values will be the words. If we design the token based on the length and alphabetical order the words, we will have a smaller time complexity in sorting and comparing part in the above, However we will have to pass a dictionary every time so that our space complexity will be larger.

2) Code Understanding

Problem 2.1 Code Reuse

Refactor the MaxStack implementation given in Lecture 5 into a MaxMeanMinStack class that provides immediate access to the current maximum, mean, and minimum of the stack contents.

Requirements:

1. Design it in such a way that it is a “drop-in” replacement for MaxStack, i.e. any functions that depend on MaxStack will continue to operate correctly with MinMeanMax stack.
2. Include a comprehensive set of tests, including a test that requirement 1 is satisfied.
3. Update or add the time and complexity for each new method.

```
# Your code solution here
class MaxMeanMinStack():
    # O(1) time | O(1) space
    def __init__(self):
        "Initalizes an instance of the class"
        self.maxStack = []
        self.minStack = []
        self.meanStack = []
        self.stack = []

    # O(1) time | O(1) space
    def peek(self):
        "Returns what is on the top of stack but does not remove it"
        return self.stack[-1]

    # O(1) time | O(1) space
    def pop(self):
        "Removes top element from stack and returns it"
        self.maxStack.pop()
        self.minStack.pop()
        self.meanStack.pop()
        return self.stack.pop()

    # O(1) time | O(1) space
    def push(self, number):
        "Adds an element to the top of the stack"
        newMax = number
        newMin = number
        newMean = number
        if len(self.maxStack):
            newMax = max(self.maxStack[-1], number)
        if len(self.minStack):
            newMin = min(self.minStack[-1], number)
        if len(self.meanStack):
            newMean = (self.meanStack[-1] * len(self.meanStack) + number) / (1 +
len(self.meanStack))
        self.maxStack.append(newMax)
```

```
        self.minStack.append(newMin)
        self.meanStack.append(newMean)
        self.stack.append(number)

    # O(1) time | O(1) space
    def getMax(self):
        "Return the largest item currently in the stack"
        return self.maxStack[-1]

    # O(1) time | O(1) space
    def getMin(self):
        "Return the smallest item currently in the stack"
        return self.minStack[-1]

    # O(1) time | O(1) space
    def getMean(self):
        "Return the mean value currently of the stack"
        return self.meanStack[-1]


def test_MaxMeanMinStack():
    # Q: How do we test a Class?
    ms = MaxMeanMinStack()  # Q: Make an instance, and test that.
    assert ms.maxStack == [], 'init'  # Verify init is working (Whitebox)
    assert ms.minStack == [], 'init'  # Verify init is working (Whitebox)
    assert ms.meanStack == [], 'init'  # Verify init is working (Whitebox)
    ms.push(2)
    assert ms.peek() == 2, 'n=1'
    assert ms.getMax() == 2, 'n=1'
    assert ms.getMin() == 2, 'n=1'
    assert ms.getMean() == 2, 'n=1'
    ms.push(3)
    ms.push(4)
    ms.push(1)
    assert ms.peek() == 1, 'n=1'
    assert ms.getMax() == 4, 'n=1'
    assert ms.getMin() == 1, 'n=1'
    assert ms.getMean() == 2.5, 'n=1'
    ms.pop()
    ms.pop()
    assert ms.peek() == 3, 'n=1'
    assert ms.getMax() == 3, 'n=1'
    assert ms.getMin() == 2, 'n=1'
    assert ms.getMean() == 2.5, 'n=1'


if __name__ == '__main__':
    test_MaxMeanMinStack()
```

Problem 2.2 Code Extensions

Brown Short Name:_____

Your boss at Box-Plots-R-Us.com loves box plots (go figure) and has asked you to modify your class to also allow immediate access to the lower quartile, median, and upper quartile of the stack contents. Describe how you would do this and the performance of the ~~MinMeanMaxQ1Q2Q2Stack~~ MinMeanMaxQ1Q2Q3Stack stack class. Include answers to the following:

1. Discuss how you would implement the modifications.
2. What if any performance impacts these changes would have on your existing implementation of MinMaxMean stack
3. What the performance of any new method(s) will be.

Be specific and use complete sentences.

Your answer here:
I will add three more lists in the MinMeanMaxQ1Q2Q3 class to record current Q1, Q2 and Q3. These 3 lists perform in the same way as the previous 3 lists and can have O(1) time and space complexity in all of its in-class functions.

Problem 2.3 Generalizing Code (Extra Credit)

Describe how you might create a StatStack that has the same performance of MaxStack which would allow users to specify a statistic of interest that they want immediate access to. Be specific on what the user would need to supply to describe their statistic of interest, and how it would be applied. Implement an example if you know how.

Be specific and use complete sentences.

Your answer here:
The StatStack will have the same structure as the MaxStack. There is a list in the class called ‘statStack’ and a list called ‘stack’. The getStat function will return the current statistic specified by the user as an input parameter. To update the statStack, the user needs to specify a mathematical way to update the current statistic if a new element is pushed into the stack. Therefore, in the init function, the user has to provide a method to calculate the mathematical update of the ‘statStack’.

Brown Short Name:_____

3) Sorting

Sorts that change their behavior based on the values of the data they are sorting, like Bubble Sort, are said to be **adaptive sorts**. Other sorts, like Heap Sort, do not change their behavior based on the values of the data said to be **non-adaptive sorts**.

Problem 3.1 Worst, Best and Partial Cases

Describe worst case and best case inputs for Bubble Sort and Heap Sort. When a list is almost fully sorted, which will perform better? Why?

Your answer here:
The worst case for Bubble sort is O(n^2) and the best case is O(n). Because if keep track of the number of swaps in each pass, in the first pass, we can terminate the program if no exchanges are made, which takes n-1 steps and the time complexity is O(n). The worst case and best case for Heap sort are both O(nlogn). The best case is still O(nlogn) because when all elements are equal (O(n), since you don't have to reheapify after every removal, which takes log(n) time since the max height of the heap is log(n)). When a list is fully sorted, Bubble sort will perform better.

Problem 3.2 Adaptive and Non-adaptive sort families

Create a table of the information that contains this information for 4 additional sorts. Include the sorts used by Python's sort function (Timsort). Be specific and use complete sentences.

Your answer here:
Timsort is adaptive, which has O(n) in best case and O(nlogn) in the worst case.
Insertion sort is adaptive, which has O(n) in best case and O(n^2) in the worst case.
Selection sort is non-adaptive, which has O(n^2) in both best and worst case.
Bucket sort is adaptive, which has O(n+k) in best case and O(n^2) in the worst case.

Problem 3.3 Adaptive Merge Sort

You can make in-place Merge Sort adaptive by avoiding merge steps where possible. Explain the criteria for this. What is the time and space complexity of this approach for a fully sorted list? Explain your answer. Be specific and use complete sentences.

Your answer here:
Since in the merge step of two sub sorted lists, we need to compare each element of two sublists to get a sorted merged list of the two. However, if we compare the last element x of sublist 1 and the first element y of the sublist 2, we can directly append 2 to the tail of 1 if x is larger than y. In this case, we can shorten the time we need in merging and get a time complexity of O(logn) for a fully sorted list.

Problem 3.4 Implement Adaptive Inplace Merge Sort

Requirements:

1) Provide appropriate test cases

2) Provide appropriate docstrings

```
### Your code answer here
def mergeSort(arr):
    if len(arr) > 1:
        # Finding the mid of the array
        mid = len(arr) // 2
        # Dividing the array elements
        L = arr[:mid]
        # into 2 halves
        R = arr[mid:]
        # Sorting the first half
        mergeSort(L)
        # Sorting the second half
        mergeSort(R)
        i = j = k = 0
        # check the criteria for best case
        # here i use <= and < to preserve the original order if two elements are the same
in arr
        if L[-1] <= R[0]:
            for t in range(0, len(L)):
                arr[t] = L[t]
            for t in range(len(L), len(L) + len(R)):
                arr[t] = R[t-len(L)]
        elif R[-1] < L[0]:
            for t in range(0, len(R)):
                arr[t] = R[t]
            for t in range(len(R),len(R) + len(L)):
                arr[t] = L[t-len(R)]
        else:
            while i < len(L) and j < len(R):
                if L[i] < R[j]:
                    arr[k] = L[i]
                    i += 1
                else:
                    arr[k] = R[j]
                    j += 1
                k += 1
            # Checking if any element was left
            while i < len(L):
                arr[k] = L[i]
                i += 1
                k += 1
            while j < len(R):
                arr[k] = R[j]
                j += 1
                k += 1


def test_merge_sort():
    arr = [12, 11, 13, 5, 6, 7]
    mergeSort(arr)
```

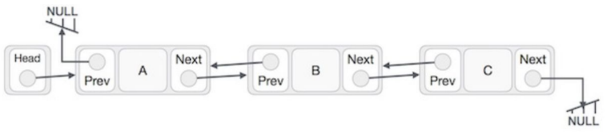
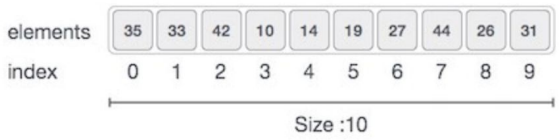
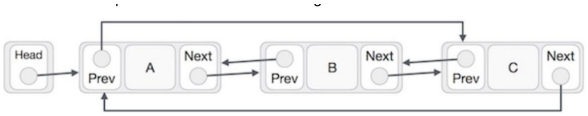
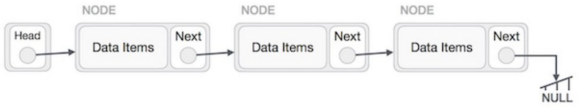
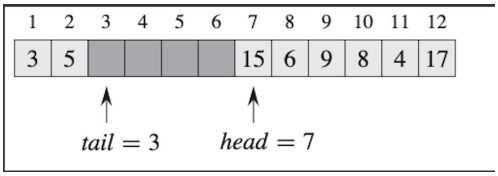
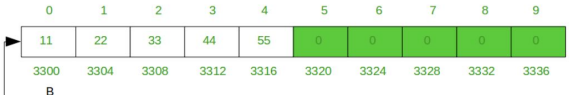
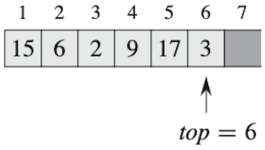
```
    assert arr == [5,6,7,11,12,13]


if __name__ == '__main__':
    test_merge_sort()
```

Brown Short Name:_____

4) Data Structure Definitions

Match the sentences to the corresponding data structure diagram

	Answer	Sentence		Data Structure Diagram
4.1	B	This is an array	A	
4.2	D	This is a singly linked list	B	
4.3	E	This is a queue	C	
4.4	C	This is a circular linked list	D	
4.5	G	This is is a stack	E	
4.6	A	This is a doubly linked list	F	
4.7	F	This is a dynamic array	G	

Brown Short Name:_____

5) Data Structure Representation

Problem 5.1 Trees as Arrays
The array approach for representing heaps presented here is typically used to store complete binary trees. Can this approach other types of binary trees? Explain and give examples. Be specific and use complete sentences.

Your answer here:
Yes, since the relative position between parents and childs are defined in a mathematical way by the index, we can easily get the node by calculating the index. Even if the tree is not complete, we can still use the array to store it and there will be no conflicts since empty nodes will be left as 'None' in the array. For example, a binary tree with root -> left child -> right child will be in the form [root, left child 1, None, None, right child].

Problem 5.2 Trees as Arrays
If this array representation is traversed from left to right, in what order are the nodes of the tree visited? Be specific and use complete sentences.

Your answer here:
The traversal will be preorder because in the array, the child is always behind the parent. There we will always meet all the parents before we meet their children from left to right in the array.

Problem 5.3 Trees as Arrays
The book's implementation prefixes a zero to the array representation used. Explain why this is the case. Be specific and use complete sentences.

Your answer here:
Since in mathematics, we use 1 to mark the start of a sequence, while in computer science, the start index of an array is usually 0. The 0 is inserted at the first to make the array representation match the mathematical representation and calculation.

Problem 5.4 Trees as Arrays
In order to eliminate this leading zero, different calculations are needed to find the index to the parent node, left child node, and right child node from a given node index i. Implement Python functions to calculate those indices below.

```
### Your code answer here, assume n is the length of the array.
def parent(i, n):
    # time complexity O(1): space: O(2)
    # return None if it is the root
    # else track the children by index in the mathematical formula
    if i == 0:
        return None
    elif i // 2 == 0:
```

```
        return (i-2) / 2
    else:
        return (i-1) / 2

def left_child(i, n):
    # time complexity O(1): space: O(2)
    # return None if out of list length
    if i * 2 + 1 < n:
        return i*2+1
    return None

def right_child(i, n):
    # time complexity O(1): space: O(2)
    # return None if out of list length
    if i * 2 + 2 < n:
        return i * 2 + 2
    return None
```

Brown Short Name:_____

6) Algorithms - Heaps

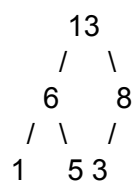
Problem 6.1 Heap or Sleep?

Draw the max-heap that results from using a max-heap version of buildHeap on the following numbers:

[5, 6, 3, 1, 8, 13]

Hint: You can check your understanding as follows: The code cell at the bottom of the build_heap page contains the complete min-heap implementation. It can easily be modified into a max-heap implementation.

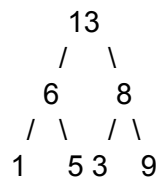
Your answer here:



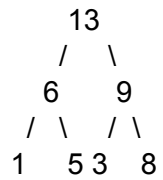
Draw the state of this ~~min-heap~~ max-heap after each of the following operations are performed in sequence.

Problem 6.2 insert(9)

Your answer here:
1:



2:

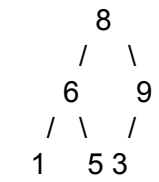


Brown Short Name:_____

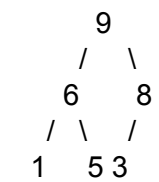
Problem 6.3 delMax()

Your answer here:

1:



2:



Brown Short Name:_____

7) Complexity Definitions and Properties

State whether the following Claims are true or false, and then prove your statement.

7.1 Claim $f(n) = log(2n) \in O(log(n))$

Your answer here:
True. log(2n) = log(2) + log(n). Since when n -> infinity, (log(2n))/log(n) = 1, so log(2n) $\in$ O(log(n)).

7.2 Claim $f(n) = n\ log\ n \in O(n\ log(2n))$

Your answer here:
True. Since nlog(2n) = nlog(n) + nlog(2) and log(2) is a constant, O(nlog(2n)) = O(nlogn) and f(n) $\in$ O(nlog(2n)).

7.3 Claim $O(n!)$ *is a proper subset of* $O(2^n)$

Your answer here:
False. When n goes to infinity, limit n!/ 2^n = limit n/2 = infinity. Hence, O(n!) is not a proper subset of O(2^n) .

7.4 Claim $O(2^n)$ *is a proper subset of* $O(n!)$

Your answer here:
True. When n goes to infinity, limit 2^n /n! = limit 2/n = 0, Hence, O(n!) is a proper subset of O(2^n).

7.5 Claim $O(2^n)$ *is a proper subset of* $O(10^n)$

Your answer here:
True. When n goes to infinity, limit $2^n / 10^n$ = limit 1 / 5^n = 0. Hence, O(2^n) is a proper subset of O(10^n).

8) SQL

Please use this notebook for the SQL topic:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6365821042734797/936865453821170/6131264883587655/latest.html
Instructor Note: The notebook for the topic uses the Databricks platform. It also depends on Databricks problems in 9.2, as such we recommend completing part 9.2 before attempting this part. There are a total of 5 questions and 5 expected outputs for Part 8. Read through the notebook carefully and paste your output to the questions in your exam submission, as well as the contents of the cell that produced those outputs.
~~**Instructor Note: The notebook for the topic uses the Databricks platform. It also depends on Databrick problem 9.2.1**~~
~~**A link to this notebook will be provided soon.**~~

My completed SQL notebook is here:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8961796018408077/224134217062465/8694839941008773/latest.html

8.1

```
Input Cell
%sql
select director, count(*) as count from movies  where year >= 2011 and director is not null group by director order by count DESC limit 5;
```

Output Cell

(2) Spark Jobs

	director	count
1	Andrew Jones	24
2	Sergey A.	24
3	Rene Perez	18
4	David DeCoteau	17
5	Michael Feifer	16

Showing all 5 rows.

Command took 1.45 seconds -- by enmin_zhou@brown.edu at 12/11/2020, 9:49:05 PM on data1050

8.2

```
Input Cell
%sql
select country,
  sum(CASE WHEN budget IS NOT NULL THEN 1 ELSE 0 END)/count(*) AS percentage
from movies where country not like '%,%' group by country order by percentage DESC, country ASC limit 10;
```

Output Cell

▸ (2) Spark Jobs

	country	percentage
1	Ghana	1
2	Greenland	1
3	Honduras	1
4	Mauritius	1
5	Mozambique	1
6	Palestine	1
7	Serbia and Montenegro	1
8	Kosovo	0.6666666666666666
9	Mongolia	0.6666666666666666
10	Puerto Rico	0.6428571428571429

Showing all 10 rows.

Command took 0.87 seconds -- by enmin_zhou@brown.edu at 12/11/2020, 10:05:41 PM on data1050

8.3

```
Input Cell
%sql
select m.imdb_title_id as id, m.title, count(*) as degree from movies as m, names as n, title_principals as t
where m.imdb_title_id = t.imdb_title_id and n.imdb_name_id = t.imdb_name_id and m.title = 'A Tale of Two
Cities' group by m.imdb_title_id, m.title;
```

Output Cell

8.4

```
Input Cell
%sql
with f as (select m.imdb_title_id as id, m.title, n.name from movies as m, names as n, title_principals as t
where m.imdb_title_id = t.imdb_title_id and n.imdb_name_id = t.imdb_name_id and n.name='Kevin Bacon')
select count(distinct n.name)-1 as number_of_people from names as n, f, title_principals as t where f.id =
t.imdb_title_id and n.imdb_name_id = t.imdb_name_id;
```

```
Output Cell
```

8.5

```
Input Cell
%sql
SELECT COUNT(DISTINCT n.name) - 1 - 406 as number_of_people
FROM names AS n
INNER JOIN title_principals AS t0
   ON n.imdb_name_id = t0.imdb_name_id
   AND t0.imdb_title_id IN(
      SELECT
         imdb_title_id
      FROM title_principals AS t1
      WHERE imdb_name_id IN (
         SELECT
            kevin_bacon_name_1.imdb_name_id
         FROM names AS kevin_bacon_name_1
         INNER JOIN title_principals AS t2
         ON kevin_bacon_name_1.imdb_name_id = t2.imdb_name_id
```

```
        AND t2.imdb_title_id IN(
            SELECT
                imdb_title_id
            FROM title_principals AS t3
            WHERE t3.imdb_name_id = (
                SELECT
                    imdb_name_id
                FROM names AS kevin_bacon_name_0
                WHERE kevin_bacon_name_0.name='Kevin Bacon' limit 1
            )
        )
    )
)
INNER JOIN movies AS m
    ON t0.imdb_title_id = m.imdb_title_id
INNER JOIN title_principals AS tp
    ON t0.imdb_title_id = tp.imdb_title_id;
```

```
Output Cell
```

▸ (2) Spark Jobs

	number_of_people
1	21888

Showing all 1 rows.

9) Data Systems

Problem 9.1 MongoDB skill check
You are a data engineer working on a MongoDB system. Your database has a collection called “items” to represent different items for sale at an online store. Your job is to add a field named “tax” with value equal to 0.15 for all orders where the “country” field is equal to Canada “CA.” What query would you write in the MongoDB shell to add this tax to the items?

Your answer here:

```
db.getCollection(‘items’).aggregate([
  {“$match”: {“country”: {“eq”: “CA”}}},
  {“$addFields”: {“tax”: 0.15}}
])
```

Problem 9.2 DataBricks.com skill check
Please import this notebook for the Databricks problems:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6365821042734797/952699336051620/6131264883587655/latest.html

Update 12/10 17:15, a student pointed out some typos due to markdown syntax for problem 9.2.2. We've updated the notebook as well as clarified some of our terms. If you follow the expected output on the old version, you will still be fine. But if you're confused by some point, do refer to the updated notebook.

Additional hint for 9.2.3: if you're having trouble running GraphFrames, recall that for it to be pre-installed on Databricks, you must use a ML Beta cluster.

~~https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6365821042734797/952699336051620/6131264883587655/latest.html~~

My completed Databricks notebook is here:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8961796018408077/2579410407470021/8694839941008773/latest.html

There are a total of 4 questions and 4 expected outputs for Problem 9.2. Read through the notebook carefully and paste your output to the questions in your exam submission, as well as the contents of the cell that produced those output.

9.2.1

```
Input Cell
movies_df = spark.read.format("csv").option("multiLine", "true").option("header",
"true").option("inferSchema", "true").load("s3a://data1050-f20-final/IMDb_movies.csv")
names_df = spark.read.format("csv").option("header", "true").option("inferSchema", "true").option("multiLine",
"true").load("s3a://data1050-f20-final/IMDb_names.csv")
ratings_df = spark.read.format("csv").option("header", "true").option("inferSchema", "true").option("multiLine",
"true").load("s3a://data1050-f20-final/IMDb_ratings.csv")
title_principals_df = spark.read.format("csv").option("header", "true").option("inferSchema",
"true").option("multiLine", "true").load("s3a://data1050-f20-final/IMDb_title_principals.csv")
print(movies_df.count(), names_df.count(), ratings_df.count(), title_principals_df.count())
```

```
Output Cell
(16) Spark Jobs
movies_df:pyspark.sql.dataframe.DataFrame = [imdb_title_id: string, title: string ... 20 more fields]
names_df:pyspark.sql.dataframe.DataFrame = [imdb_name_id: string, name: string ... 15 more fields]
ratings_df:pyspark.sql.dataframe.DataFrame = [imdb_title_id: string, weighted_average_vote: double ... 47 more
fields]
title_principals_df:pyspark.sql.dataframe.DataFrame = [imdb_title_id: string, ordering: integer ... 4 more fields]
85853 297778 85855 835505
```

```
Command took 22.84 seconds -- by enmin_zhou@brown.edu at 12/11/2020, 5:22:35 PM on data1050
```

9.2.2

```
Input Cell
name_df = names_df[['imdb_name_id', 'name']]
name_df = name_df.withColumnRenamed("imdb_name_id", "id")
title_df = movies_df[['imdb_title_id', 'title']]
title_df = title_df.withColumnRenamed("imdb_title_id", "id").withColumnRenamed("title", "name")
verticesDF = name_df.union(title_df)
principal_df = title_principals_df[['imdb_title_id', 'imdb_name_id']]
edgesDF = principal_df.withColumnRenamed('imdb_title_id', 'src').withColumnRenamed('imdb_name_id',
'dst')
print("verticesDF:", verticesDF.count(), "check:", name_df.count()+title_df.count())
print(verticesDF.columns)
print(edgesDF.columns)
```

```
Output Cell
(6) Spark Jobs
  name_df:pyspark.sql.dataframe.DataFrame = [id: string, name: string]
  title_df:pyspark.sql.dataframe.DataFrame = [id: string, name: string]
  verticesDF:pyspark.sql.dataframe.DataFrame = [id: string, name: string]
  principal_df:pyspark.sql.dataframe.DataFrame = [imdb_title_id: string, imdb_name_id: string]
  edgesDF:pyspark.sql.dataframe.DataFrame = [src: string, dst: string]
verticesDF: 383631 check: 383631
['id', 'name']
['src', 'dst']


Command took 11.91 seconds -- by enmin_zhou@brown.edu at 12/11/2020, 5:38:54 PM on data1050
```

9.2.3

```
Input Cell

from graphframes import *
g = GraphFrame(verticesDF, edgesDF)
id = verticesDF.filter(verticesDF.name=='A Tale of Two Cities').collect()[0].asDict()['id']
g.outDegrees.filter(g.outDegrees.id==id).show()
```

```
Output Cell

(3) Spark Jobs
+---------+---------+
|       id|outDegree|
+---------+---------+
|tt0008652|        8|
+---------+---------+


Command took 6.95 seconds -- by enmin_zhou@brown.edu at 12/11/2020, 8:57:35 PM on data1050
```

9.2.4

```
Input Cell
id = verticesDF.filter(verticesDF.name=='Kevin Bacon').collect()[0].asDict()['id']
src = edgesDF.filter(edgesDF.dst==id).select('src').collect()
entries = []
for i in src:
  tid = i[0]
  dst = edgesDF.filter(edgesDF.src==tid).select('dst').collect()
  for j in dst:
    entries.append(verticesDF.filter(verticesDF.id==j[0]).collect()[0].asDict()['name'])
res = [[i] for i in sorted(set(entries))]
print(len(res) - 1)
res_df = spark.createDataFrame(res, ['name'])
display(res_df.head(10))
```

~~(note that the output for this cell might be very long, so please ensure that you've printed out the number of entries in your first line as per the instructions, followed by the output table. Databricks notebooks have an option for you to copy the output of a cell)~~

Clarification 10/12 15:30, We've updated the question requirements to make it easier for you to copy the results. Please see the updated notebook. The task remains the same, but the new output requirements are to
1. Output the number of entries in your result.
2. Sort your result by ascending order of actor name, then output the first 10 rows.

```
Output Cell
```

▸ (4) Spark Jobs

▸ entries_df: pyspark.sql.dataframe.DataFrame = [name: string]

406

	name
1	Aaron Ryder
2	Aaron Sorkin
3	Al Reinert
4	Alan Bowne
5	Alan Heim
6	Alan Marshall
7	Alan Mruvka
8	Alec Baldwin
9	Alexander Gruszynski
10	Alexander Rodnyansky

Showing all 10 rows.

Command took 0.43 seconds -- by enmin_zhou@brown.edu at 12/12/2020,