

The mock-exam

Introduction to Data Science and Programming

2019-12-04, 14:00–16:00

Question 1. (100 points)

The city of Copenhagen needs to assess the needs of the kindergartens (børnehaver) in Copenhagen. By law, each kindergarten teacher may take care of at most 6 children, and every child must be taken care of by at least one teacher.

The politicians have done some homework. They know that children are often getting sick and staying at home. So, they have decided to distribute funding for 1/10 less children than there actually are to each kindergarten.

As a result of this decision, the kindergarten leaders say they are struggling to cover their costs. These leaders have a chairperson to speak on behalf of them. A meeting is to be held between the ministry and your boss (the chairperson) in two hours. The chairperson calls you (a contract data scientist) to get some numbers.

The chairperson provides you with the following data files.

1. **[Data 1:]** A list of kindergartens in the city in csv format, with columns:
 - kindergarten id,
 - kindergarten name, and
 - kindergarten address.
2. **[Data 2:]** A spreadsheet for each kindergarten in csv format, where rows are hours of the day, and columns are, for each hour:
 - time period of the day (hour)
 - date
 - number of children in attendance,
 - number of teachers in attendance (whether they can take care of children or not),
 - number of teachers not able to take care of children (due to breaks, administration, or other tasks)

Answer the following questions:

1. (Identifying the problem. 10 points) What is the key problem that the chairperson would like to address here?
2. (Identifying key information. 20 points) What numbers could you compute from the data quickly to help the chairperson of the kindergarten leaders make their case with respect to this problem?
3. (Explaining the key information. 20 points) The chairperson has no understanding of Data Science. Explain what the numbers are and what they show.
4. (Data structures. 20 points) Before starting to implement any solution, you should think the solution through. State how you would implement a programme to extract the key information. How do you get started seeing what is in the file? How will you read the data in? Which data structure will you use and why?
5. (Missing data. 20 points) Some kindergartens have not been very diligent in noting the exact information per hour and there are a lot of empty fields in Data 2 (except for time period of the day and date). For the other columns:
 - (a) (10 points) What problems do these missing values pose to your analysis?
 - (b) (10 points) What strategies could you implement so that these missing values do not influence your analysis too much?
6. (Useless information. 10 points) What information is unnecessary for your analysis and why?

1. Prove that more money is needed (to prevent cutting of funding)

2. Number of children per teacher (in attendance) at each hour of day - >6 means probably less funding is not going to cut it. Also make sure that not too many breaks/administration are the cause of this.

3. #children/teacher is the number of children that a teacher has to take care of on average in that timeframe. If this number is greater than 6, more teachers are needed (and thus not less).

4a. head(file)

4b. Numpy - np.read_csv?

4c. Numpy matrices - column operations (less for-loops and faster)

Easy to see parts of data

Easy to plot with some library

5a. Missing values are always bad because they don't contribute to the data.

- Might not be able to cover our backs because of the missing data might be (eg. where there are no children in attendance)

5b. Drop them, ignore them, forget them. (np.drop_na(axis=0)?)

- Some kind of average out (not recommended unless you know more than given about the data)

6. Data 1 is useless for the given (simple) analysis.

- Number of teachers not able to take care of children is also not important if they need to do that work.

- If doing a more complicated analysis address is very important (Maybe some parts of CPH are having many problems and other parts of CPH have no problems or not many problems with not having enough teachers!)

Question 2. (60 points)

Consider the block of Python code below. The two data structures both contain some English word counts obtained from text data.

```
the_list = [  
    ("game", 11732),  
    ("play", 9987),  
    ..., # the list continues here  
    ("jauntily", 2)]  
  
the_dict = {  
    "game": 11732,  
    "play": 9987,  
    ..., # the dictionary continues here  
    "jauntily": 2}
```

the_dict is better because it has constant lookup time, compared to the_list, which has linear lookup time.

Answer the following questions.

1. (30 points) Say that we need to efficiently retrieve the counts for the individual words many times. Out of the two proposed data structures above, which one would be better fitted for that task and why?
2. (30 points) Let both data structures contain n words. Say that we need to increase the count for some word by 1. This word is only identified by the word string, say `word="stuff"`: we do not know its position in the list. In the worst case scenario, explain how long would it take, in relation to n , to perform this operation for `the_list` and how long for `the_dict`? What about the best case scenario?

Worst case:

- list: n checks for equality (iterating through entire list, when element is last in list)
- dict: 1 hashing/lookup/check

Best case:

- list: 1 check for equality (if element is first in list)
- dict: 1 hashing/lookup/check

Question 3. (20 points)

Consider the following block of Python code.

```
for i in range(0, 101):  
    if i % 3 == 0:  
        print(i)
```

Answer the following questions.

1. (5 points) How many times does the above `for`-loop execute?
2. (5 points) What does the code print out?
3. (10 points) How do we implement the same functionality by replacing the `for`-loop above with a `while`-loop?

101

All numbers divisible by 3 between 0 and 101, including 0.

```
i = 0  
while i < 101:  
    if i % 3 == 0:  
        print(i)  
    i += 1
```

Question 4. (20 points)

Consider the following block of Python code.

```
def secret_operation(k):  
    for i in range(1, len(k)):  
        j = i  
        temp = k[j]  
        while j > 0 and temp < k[j-1]:  
            k[j] = k[j-1]  
            j = j - 1  
        k[j] = temp  
    return k
```

(20 points) Let the object `k` that is passed to the function be a Python list of integers. What is the secret operation that the above function performs with the input list? Also demonstrate that you are correct by showing what happens to two short but informative lists of integers.

Sorting the list in ascending order (Insertion sort)

[3,2,1] => [2,3,1] => [2,1,3] => [1,2,3]
[1,2,3] => [1,2,3]

Question 5. (50 points)

Prove, using induction, that $n^2 - 7n + 12$ is non-negative for all integers $n \geq 3$.

Question 6. 20 points

Consider the following block of Python code.

```
def add_pairs(n):  
    return n*(n-1) + add_pairs(n-1)  
print(add_pairs(10))
```

The recursive algorithm is meant to calculate the following sum:

$$\sum_{i=1}^n i(i+1)$$

The code does not terminate.

We are missing a base case

1. (10 points) Why does the code not terminate?
2. (10 points) Explain how to fix the code so that it implements the above sum.

```
def add_pairs(n):  
    if n == 1: #Add base case to make code terminate  
        return n*(n+1)  
    return n*(n+1) + add_pairs(n-1) #Fix formula from n*(n-1) to n*(n+1)
```

Question 7. (60 points)

Consider the following data set of 15 individuals.

ID	City	Insured	Gender	Age	Income	Illness
1	1	1	0	41	40000	0
2	1	1	0	54	40000	0
3	1	1	0	42	50000	0
4	1	1	0	40	40000	0
5	1	0	0	46	50000	0
6	1	0	1	36	50000	0
7	1	0	0	32	30000	0
8	1	1	0	39	30000	1
9	1	1	0	51	70000	0
10	1	1	1	30	50000	0
11	1	1	0	48	40000	0
12	1	1	0	47	55000	0
13	2	1	0	46	60000	1
14	1	0	0	42	50000	0
15	1	1	0	61	0	0
	N	N	N	N	R (I)	N

<https://microbenotes.com/wp-content/uploads/2019/03/Nominal-Ordinal-Interval-and-Ratio-Data.jpg>

70000 - Max
55000
50000 - 3rd Q
50000
50000
- 2nd quantile (45000)
40000
40000
40000 - 1st Q
40000
0 - Min

5 point summary:
[0, 40k, 45k, 50, 70k]

Do the following.

- (10 points) For each variable, identify its type: nominal, ordinal, interval, or ratio.
- (30 points) Calculate by hand the five number summary for the income of all healthy individuals who live in city 1, have gender 0, and are at least 35 years old. Also report the sorted incomes for these individuals.
- (20 points) Suppose you load the data into Python like this:

```
data = numpy.loadtxt('data.csv', skiprows=1, dtype='int')
```

Create a mask that selects the individuals from question (b), like this:

```
mask = ( ) & ( ) & ..
```

```
mask =  
(data[:,6] == 0) & # healthy  
(data[:,1] == 1) & # city 1  
(data[:,3] == 0) & # gender 0  
(data[:,4] >= 35) # age >=35
```

I (Canot be R because it has an entry of 0)

Question 8. (50 points)

Consider the data set `data2 = data[mask, 2:6]`, where `data` and `mask` are taken from the previous question.

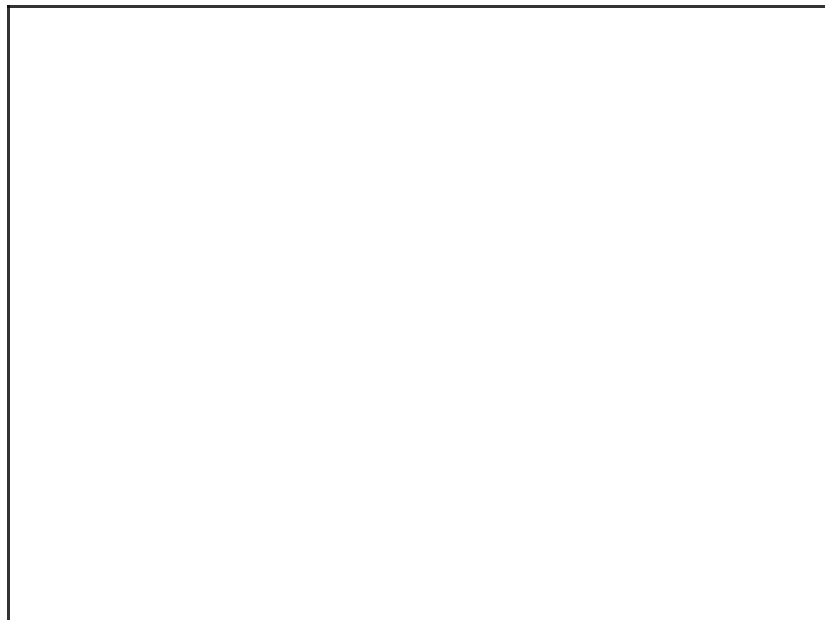
Draw by hand as exactly as possible what the following code will output. Use the box below the code for `axes1`. Draw all elements, including ticks, labels, bars, scatterplot markers. You do not need to get the colours right.

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

categories, counts = np.unique(data2[:, 0], return_counts=True)

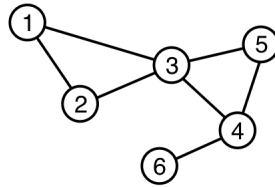
fig = plt.figure(figsize=(4, 3))
axes1 = fig.add_axes([0, 0, 1, 1])
axes2 = fig.add_axes([0.1, 0.4, 0.3, 0.55])

axes1.bar(categories, counts)
axes1.set_ylabel('Count')
axes1.set_xticks(categories)
axes1.set_xticklabels(("Uninsured", "Insured"))
axes2.scatter(data2[:, -1], data2[:, -2]);
```



Question 9. (50 points)

Given the following network.



Calculate:

1. (5 points) The degrees k_1, k_2, \dots, k_6 , and the average degree $\langle k \rangle$ of the network.
2. (15 points) The clustering coefficients c_1, c_2, \dots, c_6 , and the network average clustering coefficient C
3. (15 points) The average path length ℓ and the diameter D .
4. (15 points) Explain 3 organising principles of networks (1 sentence per principle is enough). For each principle, name an example of a real-world network.

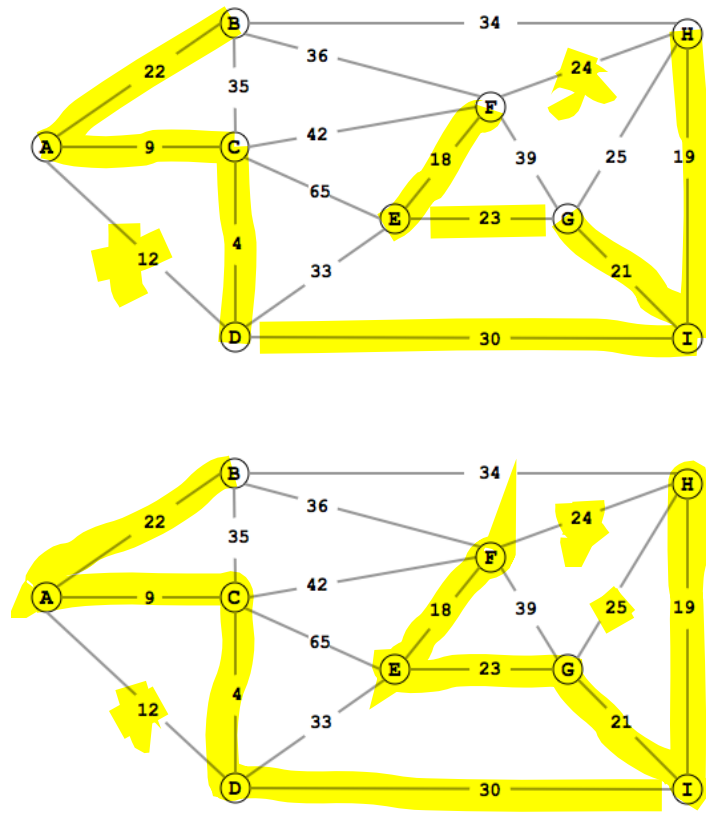
Question 10. (45 points)

For the network from the previous question:

1. (5 points) Write down the adjacency matrix.
2. (5 points) Write down the edge list.
3. (5 points) Write down the adjacency list.
4. (5 points) Give an example of a walk of length 9.
5. (5 points) Calculate all degree centralities $C_D(1), C_D(2), \dots, C_D(6)$
6. (10 points) Calculate all betweenness centralities $C_B(1), C_B(2), \dots, C_B(6)$
7. (10 points) Calculate all closeness centralities $C_C(1), C_C(2), \dots, C_C(6)$

Question 11. (30 points)

Consider the following undirected weighted graph. The figure is repeated two times for your convenience in relation to the questions below.



Complete the following.

- (15 points) Find the minimum spanning tree of the above graph by using Kruskal's algorithm. Highlight the tree edges in the first figure. Also list the edges in the order of selection by the algorithm.
- (15 points) Find the minimum spanning tree of the above graph, this time by using Prim's algorithm. Start with node F . Highlight the tree edges in the second figure. Also list the edges in the order of selection by the algorithm.

[C,D],[A,C],[E,F],[H,I],
[G,I],[A,B],[E,G],[D,I]
edge list of kruskals
algorithm

[F,E],[E,G],[G,I],[I,H],[
D,I],[D,C],[A,C],[A,B]
edge list of prims al-
gorithm

Question 12. (30 points)

Consider the following code:

```
class Client:
    bonus = 1000

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance + self.bonus

        if self.balance < 5000:
            self.level = "Basic"
        else:
            self.level = "Advanced"

    def deposit(self, amount):
        self.balance += amount
        return self.balance

class Savings(Client):
    interest_rate = 0.01

    def update_balance(self):
        self.balance += self.balance*self.interest_rate
        return self.balance
```

Savings inherits from Client =>
Savings is the child of Client &
Client is the parent of Savings

Inheritance

1. (10 points) What is the relation between Client and Savings? Which key concept of object-oriented programming enables this relation?
2. (10 points) What is then the output from the following code?

```
Lina_Tran = Savings("Lina Tran", 4500)
Lina_Tran.update_balance()
```

4545 (4500+4500*0.01)

3. (10 points) What is then the output from the following code?

```
bonus = 4000
John_Doe = Client("John Doe", 3000)
Client.deposit(John_Doe, 3000)
print(John_Doe.level)
```

"Basic" (since it's not updated upon deposit)