Heidelberg University — Group Parallel and Distributed Systems (PVS)
Winter semester 2019/2020 — Artur Andrzejak, Tuyen Le

# Problem Set 10 for lecture Mining Massive Datasets

Due January 20, 2020, 11:59 pm

---

## Exercise 1 **(2 points)**

Consider a marriage agency in charge of pairing persons based on their similarities and common interests. Each customer has to fill-in $b$ questionnaires $Q_1, \dots, Q_b$ containing $r$ questions each. Two customers are matched if they give exactly the same answers *to all $r$ questions* for *at least one* of these $b$ questionnaires. Assume that two specific customers $C_1$ and $C_2$ give with probability $p$ the same answer to any of the questions in any of the questionnaires. State formulas (and explain how you derived them) for the following cases:

**a)** The probability that $C_1$ and $C_2$ are matched.

**b)** The probability that exactly two (no matter which) questionnaires match, i.e. have the same answers for both $C_1$ and $C_2$.

## Exercise 2 **(3 points)**

Define the graph $G_n$ to have the $2n$ nodes

$$a_o, a_1, \dots, a_{n-1}, b_0, b_1, \dots, b_{n-1}$$

and the following edges. Each $a_i$ (for $i = 0, 1, \dots, n-1$) is connected to the nodes $b_j$ and $b_k$, where

$$j = 2i \mod n \text{ and } k = (2i+1) \mod n$$

For instance, the Graph $G_4$ has the following edges:

$(a_0, b_0), (a_0, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_0), (a_2, b_1), (a_3, b_2), (a_3, b_3)$.

**a)** Find a perfect matching for $G_4$ and one for $G_5$. **(1 point)**

**b)** How many different perfect matchings do $G_4$ and $G_5$ have? **(2 points)**

## Exercise 3 **(1 point)**

Consider the bipartite graph with nodes $1, 2, 3, 4, a, b, c, d$ and edges

$(1, a), (1, c), (2, b), (3, b), (3, d), (4, a)$. Draw this graph.

This bipartite graph has a perfect matching. However, whether or not the greedy algorithm yields a perfect matching depends on the ordering of incoming edges.

Find one ordering of the edges for which the greedy algorithm gives us a perfect matching and one ordering for which it does not.

## Exercise 4 **(1.5 points)**

Suppose we apply the BALANCE algorithm (with bids of 0 or 1 only) to a situation where advertiser A bids on query words x and y, while advertiser B bids on query words x and z. Both have a budget of \$2. Identify in the following list a sequence of four queries that will certainly be handled optimally by the algorithm, and explain why this is the case (and why not in the other cases).

**1)** xyyy **2)** xyyx **3)** yyxx **4)** xzyz

## Exercise 5 **(1.5 points)**

Consider an execution of the BALANCE algorithm with four advertisers, $A_1, A_2, A_3, A_4$, and four kinds of queries, $Q_1, Q_2, Q_3, Q_4$. Advertiser $A_1$ bids on queries $Q_1$ and $Q_2$; $A_2$ bids on queries $Q_2$ and $Q_3$; $A_3$ on queries $Q_3$ and $Q_4$; and $A_4$ on queries $Q_1$ and $Q_4$. All bids are equal to 1, and all clickthrough rates are equal. All advertisers have a budget of 3, and ties are broken in favor of the advertiser with the lower index (e.g., $A_1$ beats $A_2$). Queries appear in the following order:

$Q_1, Q_2, Q_3, Q_3, Q_1, Q_2, Q_3, Q_1, Q_4, Q_1, Q_4$

What is the sequence of advertisers that the BALANCE algorithm will yield? What is the competitive ratio for this instance?

## Exercise 6 **(7 points)**

This exercise is the second in the series of tasks related data processing with Spark. You should ideally reuse the implementation developed in the previous problem set (i.e. reading in an *EC2* dataset[1] and creating a Spark dataframe). For each of the following subtasks submit your code as a part of the solution.

a) Implement a subroutine in Spark which takes a dataframe (created in the last exercise) as a parameter and generates for each found unique combination *(<InstanceType>, <ProductDescription>, <AvailabilityZone>)* a new dataframe with associated pairs *<Timestamp>* and *<Price>* (we call such a dataframe a *price timeseries*). Apply Spark's transformations for higher efficiency where possible. The routine should return a dictionary which has as keys combinations *(<InstanceType>, <ProductDescription>, <AvailabilityZone>)* with each combination encoded as a single string, and as values the corresponding price timeseries.

b) Implement a subroutine *saveTimeseries(name, D)* that saves to disk a price timeseries D. The filename should be derived from the string parameter name by respecting the filename restrictions (name is typically a string encoding the combination *(<InstanceType>, <ProductDescription>, <AvailabilityZone>)* as in part a)). This routine should account for file collisions: if there exist another dataframe D' with the same filename (hence corresponding to the same combination), read this dataframe into memory and merge D with D', removing any duplicates in the process. Save the resulting dataframe into disk.

c) To test your implementation, read every file from the provided Amazon dataset and submit the amount of elements and average price obtained for the dataframe with the combination: *(<m4.16xlarge>,<Linux/UNIX>,<ca-central-1a>)*.

[1]Available on Moodle, Weekly problem sets: Datasets/dataset-EC2-series/