Exercise Sheet 6

Due date: Monday, June 19 until 13:00

- Please upload your solutions to RWTH Moodle.
- The due date is at Monday, June 19 until 13:00.
- Hand in your solutions in groups of **two to three students**. If you need to change your group, contact algds@lics.rwth-aachen.de.
- Hand in the solutions of your group as a single PDF file.
- A discussion regarding this exercise sheet will take place on **Friday, June 23 14:30** in room AH II.
- Note that Exercise 2 gives 0 points and will not be corrected.

Exercise 1 (Metropolis-Hastings Sampling of Independent Sets) 2+2+2=6 points

A set $I \subseteq V$ of nodes in a graph $G = (V, E)$ is called *independent*, if $(v, w) \notin E$ for all $v, w \in I$. Denote the independent sets of G by $\mathcal{I}_G \subseteq 2^{V(G)}$.

Fix some graph G. In this exercise, we want to uniformly sample from $\mathbb{U} := \mathcal{I}_G$ using Metropolis-Hastings sampling. Recall that the Metropolis-Hastings algorithm performs a random walk on an undirected, connected graph. To avoid confusion with G, we call this graph $\mathcal{H} = \big(V(\mathcal{H}), E(\mathcal{H})\big)$. (It is called $\mathcal{G}$ in the lecture.)

a) Define $V(\mathcal{H})$ and $E(\mathcal{H})$ for sampling from $\mathcal{I}_G$.

Hint: The edge relation $E(\mathcal{H})$ describes which elements of $\mathcal{I}_G$ are "neighbours". Think of a useful notion of "being a neighbour" for sets of nodes of G.

b) Find the maximum degree Δ of $\mathcal{H}$ and show that $\mathcal{H}$ is connected.

c) Define transition probabilities $q_{I,J}$ for $I, J \in V(\mathcal{H})$. Show that the stationary distribution of the resulting Markov chain is the uniform distribution over $\mathcal{I}_G$.

Hint: Use slide 6.48 of the lecture and recall that $\mathcal{D} = Z\mathcal{P}$ where $\mathcal{P}$ is the target distribution and Z is some constant. For our exercise, we know that $\mathcal{P}$ is the uniform distribution on $\mathcal{I}_G$.

Exercise 2 (Shuffling and Coupling) 0 points

Consider a Markov chain for shuffling n cards, where at each step a card is chosen uniformly at random and moved to the top. We run the chain until every card has been moved to the top at least once. Show that, at this stopping time, the state of the chain is uniformly distributed on all possible permutations of the cards.

Exercise 3 (Map-Reduce Algorithms for an Online Retailer) 2+2+1=5 points

We consider data records about the orders made at an online retailer. The data is given in key-value pairs of the shape $(o, (c, p, q, d))$ where

- o is an order ID,
- c is a customer ID,
- p is a product ID,
- q is a price in \$, and
- d is a timestamp.

A key value pair $(o, (c, p, q, d))$ represents an order made by customer c on date (and time) d, having ordered product p for a price of q dollars. If an order consists of multiple product purchases, then it is captured by multiple pairs with the same order ID (and same customer ID and time stamp). We assume that the price q of any product p stays the same over all orders.
Solve the following tasks.

a) *Product Revenue.* Specify a *single round* Map-Reduce algorithm that outputs all key-value pairs (p, s) where p is a product, and s is the total income generated for the retailer by selling p (that is, the sum of p's price over all orders of p).

Hint: Keep in mind that a product may be ordered multiple times by the same customer, possibly even in the same order.

b) *Bargain Shoppers.* Specify a *single round* Map-Reduce algorithm that outputs all key-value pairs $(c, (p, q))$, where c is a customer who *only* bought products with price < 20\$, and p is a product they bought with price q. You do not need to handle duplicates in this task.

c) *Also Bought.* Let p_0 be a fixed product ID. The Map-Reduce algorithm shown below outputs all key-value pairs (c, p), where c is a customer *who ordered* p_0 and p is another product they bought.

MAP: On input $(o, (c, p, q, d))$ emit $(p_0, (c, p))$.
REDUCE: On input (p_0, values) do
for all c appearing in a pair in *values*:
if $(c, p_0) \in \mathit{values}$, emit (c, p) for all $(c, p) \in \mathit{values}$ with $p \neq p_0$.

This algorithm technically does the job, but it is a terrible Map-Reduce algorithm. Explain why.

Remark: Use the „on input ..., [do some computation,] emit ...“ format for specifying your pseudocode. Make sure that your algorithms are not unnecessarily inefficient.

Exercise 4 (Analysis of Matrix Multiplication) **6+3=9 points**

In this exercise, we consider another measure for analysing Map-Reduce algorithms, and apply it to the algorithms for matrix multplication that were shown in the lecture : the single-round algorithm $\mathcal{A}_1$ (slide 7.34), the two-round algorithm $\mathcal{A}_2$ (slide 7.33), and the generalized single-round algorithm $\mathcal{A}_1^s$ (slide 7.46, where s is the „number of stripes“ parameter).
As input, we consider two matrices $A, B \in \mathbb{R}^{n \times n}$. For the three algorithms presented in the lecture solve the following tasks, ignoring the possiblity of node failures.

a) For $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_1^s$, and for each phase of the respective algorithm determine

- the total number of tasks, and
- the maximum running time of a single task

Assume the worst case input and give your result depending on n (and for $\mathcal{A}_1^s$ additionally depending on s).

Remark: For the maximum running time part, use $\mathcal{O}$-notation and just count the number of operations.

		Total number of tasks	Running time of a single task
$\mathcal{A}_1$	MAP: REDUCE:		
$\mathcal{A}_2$	First MAP: First REDUCE: Second MAP: Second REDUCE:		
$\mathcal{A}_1^s$	MAP: REDUCE:		

b) Now, for all three algorithms, compute the worst case running time (in $\mathcal{O}$-notation) using the measure above, **taking parallelization over N nodes into acccount**, for

	$N \in \Theta(1)$ nodes	$N \in \Theta(n)$ nodes	$N \in \Theta(n^2)$ nodes:
$\mathcal{A}_1$			
$\mathcal{A}_2$			
$\mathcal{A}_1^s$			