Exercise Session 12

Exercise 1.
(CLRS 25.1–4) Show that matrix multiplication defined by EXTEND-SHORTEST-PATH is associative. *Hint:* Let us write $A \odot B$ for EXTEND-SHORTEST-PATH(A, B). You have to prove that for arbitrary $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{p\times q}$ we have that $(A \odot B) \odot C = A \odot (B \odot C)$. Proof of matrix multiplication shit

Exercise 2.
(CLRS 25.1–9) Modify FASTER-ALL-PAIRS-SHORTEST-PATHS so that it can determine whether the graph contains a negative-weight cycle. Justify the correctness of your solution.

```
FASTER-ALL-PAIRS-SHORTEST-PATHS($W$)
1  n = W.rows
2  $L^{(1)}$ = W
3  m = 1
4  **while** m < n - 1
5      let $L^{(2m)}$ be a new n X n matrix
6      $L^{(2m)}$ = EXTEND-SHORTEST-PATH($L^{(m)}, L^{(m)}$)
7      m = 2m
8      **for** i = 1 to n
9          **if** $L^m_{i,i} < 0$
10             **return** "Negative weight cycle"
11 **return** $L^{(m)}$
```

We know that for $m \leq |V|$, this condition is upheld by the outer while loop on line 4. As this algorithm still checks for the minimum weight of the various paths, hence if we have any that $L^m_{i,i} < 0$ implies that we have a negative weight cycle, therefore we know that if we have a negative shortest path from $l_{i,i}$ we must have hit a negative weight cycle.

Exercise 3.
Let $G = (V, E)$ be a weighted directed graph represented using the weight matrix $W = (w_{ij})$ where

$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ w(v_i, v_j) & \text{if } i \neq j \text{ and } (v_i, v_j) \in E \\ \infty & \text{if } i \neq j \text{ and } (v_i, v_j) \notin E \end{cases}$$

How would we delete an arbitrary vertex v from this graph, without changing the shortest-path distance between any other pair of vertices? Describe an algorithm that constructs a weighted directed graph $G' = (V \setminus \{v\}, E')$ such that shortest-path distance between any two vertices in G' is equal to the shortest-path distance between the same two vertices in G in $O(|V|^2)$ time.

★ **Exercise 4.**
Assume that $G = (V, E)$ is a directed acyclic graph represented using adjacency-lists.

(a) Describe an algorithm that computes the transitive closure of G, i.e. $G^* = (V, E^*)$, and analyse its running time.

(b) Can you generalise your solution to directed graphs that may contain cycles?

Exercise 5.

Rick has given Morty a detailed map of the Clackspire Labyrinth, which consists of a directed graph $G = (V, E)$ with non-negative edge weights W (indicating distance from one location in the map to the other), along with a list of dangerous locations $D \subset V$ that Morty has to avoid.

(a) **Morty has to determine for each pair of locations $i, j \in V \setminus D$ the length of the shortest walk $i \rightsquigarrow j$ that does not have intermediate vertices in D. Describe the algorithm that Morty can use to solve the problem.**

We modify The Floyd-Warshall Algorithm with a single additional condition that has to be checked every time we compute $d^k_{i,j}$. We need to make sure that the path were computing does not go through a dangerous vertex $d^k_{i,j} \in D$, meaning this vertex has to be skipped.

(b) **Assume now that Morty also needs to pass by some location in the set $S \subseteq (V \setminus D)$. How does Morty compute the length of the shortest walk from $i \rightsquigarrow j$ that avoids dangerous locations D while ensuring to pass by some location in S?**

We do the same Floyd-Warshall algorithm as in a). The algorithm outputs a V^2 matrix such that for each $i, j \in V \setminus D$ we have computed the shortest path for all $i \rightsquigarrow j$. To determine the shortest path $i \rightsquigarrow S \rightsquigarrow j$ by definition of the shortest path, such that $i \rightsquigarrow S \rightsquigarrow j$ must be the same as $\delta(i, S) + \delta(S, j)$

We do the same Floyd-Warshall algorithm. This algorithm makes sure that we do not pass any dangerous vertices $d^k_{i,j} \in D$, the algorithm will give us a V^2 matrix such that we have computed the all paths shortest paths avoiding all dangerous vertices. Ie this means that we know the shortest path from any given point to all other points. Therefore we can by definition show that we can travel from a starting point $i \rightsquigarrow S \in V$ midpoint and $S \rightsquigarrow j$ which is the final location. By definition the shortest path from $i \rightsquigarrow S \rightsquigarrow j$ therefore must be the combination of the shortest paths from the start to the mid point and the midpoint to the ending location. This means it can compute the shortest path of any