Please read the instructions carefully:

1. There are two stages to each assignment:

   (a) For the first stage, you need to submit your homework solution as a single pdf file to NYU Classes. It can be either a scanned copy of your written solution or photos joined into a pdf. You are also allowed to typeset the solutions in MS Word/LaTeX. In case of scanned copies it is your responsibility that the solution is readable. There are **no late submissions accepted**, as sample solution outlines (to be used for the second stage of the homework) will be given out on the day of the submission of the first stage of the homework. You should leave around 5 to 6 cm of space between each part of the answers for the second stage of the homework.

   (b) For the second stage of the assignment, you are required to critique your own solutions and analyze the mistakes that you made. Based on the outlines provided, you need to write the self criticism in the empty space that you left in your homework. Do not rewrite your solution, you need to understand and point out the mistakes that you made when writing the original solutions.

   A good analysis of your solution should summarize the key steps, ideas or techniques that you previously missed. It should prompt you to reorganize your thoughts and think about what you can do to improve the solutions. Even in the case where you completely missed the solution it is insightful to point out why the approach adopted by you does not work and identify the step where it fails. This will allow you to get full credit on the self evaluation part of the homeworks and of course help you in understanding the question.

   Mark correct solutions in **green**, incorrect solutions in **red** and if you are unsure about any solutions write it in **blue**.

2. For each question where you are required to design an algorithm, you need to clearly explain the algorithm and prove the running time complexity as asked in the question.

3. You can assume $O(1)$ complexity of arithmetic of numbers (e.g., multiplication/division of numbers and calculations of $n!$) unless otherwise stated in the question.
Question 1. (10 points) DFS mechanics
For each graph in fig. 1, show the “DFS forest” resulting from an execution of DFS. Whenever there is a choice of vertices, choose the one that is alphabetically first. Identify the cross, forward, and back edges, and label each vertex with its discovery and finishing time.

Question 2. (10 points) SCC mechanics
For each graph in fig. 1, run the strongly-connected components algorithm. Whenever there is a choice of vertices, choose the one that is alphabetically first. Show the “DFS Forest”, including discovery and finishing times, for both runs of the algorithm. Draw the resulting component graph.

Question 3. (10 points) Dijkstra mechanics
Run Dijkstra’s algorithm for the graph in fig. 2, starting from 0. Whenever there is a choice of two or more vertices, choose the one with the smaller value. Show the state of the algorithm at the end of each loop iteration.
**Question 4. (20 points) MST mechanics**
For the graph in fig. 3, do the following operations:

![Figure 3: Query graph for MST](image)

1. (10 points) Run Prim’s algorithm, starting at vertex 0. List the edges in order of their addition to the MST.

2. (10 points) Now do the same for Kruskal’s algorithm.

**Question 5. (35 points) Installing fire stations**
Suppose we are given a model of a city as a directed graph $G = (V, E)$, where we have $n$ neighbourhoods and $m$ streets, represented by the vertices and edges respectively. We will assume that the streets are one-way. Currently our city does not have any fire stations so it is really unsafe. To keep the city safe, we need to install fire stations at some strategic locations, so that each neighbourhood will be protected.

Let us say that neighbourhood $i$ protects neighbourhood $j$, if we have installed a fire station at $i$ and there is a path to reach $j$ from $i$. In such a case we will say that $j$ is protected.

**Note 5.1.** Any neighbourhood can protect itself, if it has a fire station installed there.

1. (15 points) We want to install the minimum number of fire stations possible so that each neighbourhood is protected.
   
   (a) (5 points) Assume that the structure of the city is a DAG. Give an algorithm which takes such a city structure and returns the minimum number of fire stations to install, so that each neighbourhood is protected, in $O(n + m)$.
   
   (b) (10 points) Now solve the above problem for a general city structure.

2. (10 points) Now suppose that there is a cost, $c_i$, for installing a fire station at neighbourhood $i$. Give an algorithm which takes a general city structure and returns the minimum cost to set up fire stations in such a way that each neighbourhood is protected, in $O(n + m)$. 


3. (10 points) The fire stations have decided that they will only protect a neighbourhood \( j \) if it is possible for them to get to \( j \) from their neighbourhood, say \( i \), and get back to \( i \) from \( j \). Solve the previous problem for this situation, with the same time complexity.

**Question 6. (15 points) Nearest fire station**
Suppose that we are given a model of a city as a directed, weighted graph \( G = (V, E), w : E \to \mathbb{R}_{\geq 0} \), where we have \( n \) neighbourhoods and \( m \) streets, represented by the vertices and edges respectively. We will assume that the streets are one-way.
We are also given that \( k \) of these neighbourhoods have fire stations installed. We want to find the nearest fire station for each neighbourhood, where we measure the distance **from the fire station to the neighbourhood**.
Give an algorithm which finds the nearest fire station for each neighbourhood in \( O((n+m+k) \log n) \).

**Note 6.1.** If the fire stations is at the same location then the distance is 0.

**Question 7. (15 points) Fulfilling the network**
We are given the structure of an internet network in a city as an undirected, weighted graph \( G = (V, E), w : E \to \mathbb{R}_{\geq 0} \), where the nodes denote the neighbourhoods and the edges denote possible FIOS connections between neighbourhoods.
Currently only some of the connections have been installed, while others are not. We denote the state of a connection by the function \( \beta : E \to \{0, 1\} \), where 0 means disconnected and 1 means connected.
Neighbourhoods \( i \) and \( j \) can communicate to each other, if there is a sequence of connections \( i \leftrightarrow v_1 \leftrightarrow v_2 \leftrightarrow \ldots \leftrightarrow v_k \leftrightarrow j \). We want to make sure that all neighbourhoods are able to communicate to each one, but we want to minimize the total cost of connections.
Given an algorithm which tells us what connections to install in \( O((n+m) \log n) \).

**Question 8. (15 points) All shortest paths**
Given a directed, weighted graph \( G = (V, E) \) and a source \( s \), we want to find the number of shortest paths from \( s \) to every other vertex.
Give an algorithm which returns the number of shortest paths from \( s \) to every other vertex in \( O((n+m) \log n) \).

**Question 9. (20 points) Minimum Dijkstra tree**
Given a directed, weighted graph \( G = (V, E) \) and a source \( s \), if we perform a run of Dijkstra’s algorithm starting from \( s \), there are multiple possible trees that we may get. Give an algorithm which finds the Dijkstra tree with minimum total weight, in \( O((n+m) \log n) \).

![Figure 4: Graph to get minimum tree](image-url)
E.g. for the graph in fig. 4, we can get two trees starting from A, as shown in fig. 5, both of which give the correct minimum distance to all nodes. Yet the tree on the left has total weight 7 + 8 + 5 = 20, while the tree on the right has total weight 7 + 8 + 6 = 21, which means that we should return the tree on the left.

**Question 10. (20 points) A dangerous gamble**

Suppose that we are given a model of a city as a directed, weighted graph \( G = (V, E) \), where we have \( n \) neighbourhoods and \( m \) streets, represented by the vertices and edges respectively. We will assume that the streets are one-way. The weight of a street represents the time it takes for us to cross this street.

We are currently in the process of building a time machine in our garage but we are still missing a component, made of two parts. Due to their highly classified status, they not available to buy out in the open, so they are only sold in certain shady neighbourhoods throughout the city. Also because of the NSA spying on the black markets, no single neighbourhood sells both the parts. In addition to this, the two parts are also highly volatile in nature, when kept individually and explode if we don’t put the two parts together and finish the component, before a small time \( T \).

We are given a list of the shady neighbourhoods which sell the first part and the ones which sell the second part. When we buy the first part at any point we will go to a neighbourhood which sells the second one and put them together, before heading to the garage.

We start from our house at neighbourhood \( s \) and want to reach the garage, with the finished component, at node \( t \).

Give an algorithm which takes the city structure as described and gives the fastest way to reach from \( s \) to \( t \), while also buying the parts along the way and making sure they don’t explode, in \( O(n^3) \).

**Question 11. (25 points) Super independent**

Given a tree with costs on each vertex, we define an independent set as a subset of the vertices none of which are connected to each other and the cost of the set as the sum of costs of each vertex in the set.

Give an algorithm which takes such a tree and returns the maximum cost independent set in \( O(n) \).

**Question 12. (30 points - Extra credit) Collecting flowers**

Suppose that we are given a rectangular field, which is cut into \( n \times n \) gardens, parallel to its sides. We have planted \( m \) different types of flowers in this field, with the restriction that each garden has a single type of flower in it, e.g. in fig. 6, we have 5 different types of flowers.

We want to select the maximum number of gardens possible such that they are in a contiguous
region and they have at most two different types of flowers in between them, e.g. in fig. 6, we can select 9 gardens in total.

Give an algorithm which solves this problem in $O(n^2 \log n)$. 