Step 1: Read and draw a graph

Visual appearance: Make sure you use same scales for both axes. As one of the results of that your circular nodes will really appear circular. Make sure you set the nodes to have reasonable size – you can make this procedure automatic based on the graph's size and the bounding box. "Hide" the edge ends behind the nodes to which they are incident. Do not label nodes if this is not necessary to refer to them. Do not show arrow markers on the edges if they are not directed. On the other hand, choose the arrow size and style so that it is visible, if edges are directed. Consider using blending (1 at start, 0 at end) instead of showing arrows explicitly. Choose a reasonable scaling of your graph drawing images. Do not make images unnecessarily large – this makes your report not to look neat and professional. Play with node and edge colors to make them look better, consider using dark gray instead of black both for nodes outlines and edges – black creates very cluttered pictures. You can also consider using transparency.

Notation: Introduce and use notation. We usually use G for a graph, u or v for nodes/vertices, $\{u, v\}$ for an undirected edge and (u, v) for a directed one, n and m for the number of nodes and edges, respectively, Greek Γ is used to refer to a graph drawing, which you can also call *layout*. Become acquainted with the use of basic graph-theoretical notions. Some examples are: *degree* of a node - number of edges incident to a node, denoted by $\deg(u)$; *path* in a graph - sequence of vertices and edges incident to each other, denoted by either listing the vertices of the path or the edges or both, like $P = v_1 e_1 v_2 e_2 \dots e_n v_n$; the *length* of a path $\text{length}(P)$ which is commonly defined as the number of edges in the path. For more notions we recommend to consult the Graph Theory book of Diestel[1].

Software: In one paragraph describe the software, programming language and libraries you are using. Consider describing the structure of your software – how you plan to organize the data structures and algorithm classes. This of course can change as you work on the implementation.

[1] Reinhard Diestel, Graph Theory (Graduate Texts in Mathematics), Springer, 2005.