CSE100 Algorithm Design and Analysis — Homework set #4
Fall semester 2020 — Miguel Á. Carreira-Perpiñán

Total possible marks: 100. Homeworks must be solved individually. This set covers chapters 22–24 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al. References to pages and figures correspond to the textbook. Do not distribute. For use only of UC Merced CSE100 Fall 2020 students.

Exercise 1: breadth-first search (20 points). Consider the breadth-first search algorithm BFS(G) of p. 595 in the book.

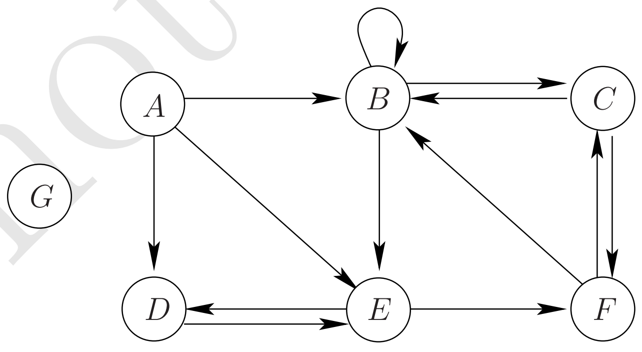
1. (4 points) That algorithm uses the adjacency list representation of the graph $G = (V, E)$. Rewrite its pseudocode, calling it BFS2(G), but assuming we use the adjacency matrix representation. That is, calling $n = |V|$ the number of vertices, we have a binary matrix $\mathbf{A}$ of $n \times n$ where $a_{uv} = 1$ if $(u, v) \in E$ and 0 otherwise.

2. (5 points) Give the runtime in asymptotic notation of BFS2(G) and compare it with that of BFS(G). Comment on the result.

3. (6 points) Modify BFS2 into a BFS2-Connected-Components(G) algorithm that finds all the connected components of an undirected graph G (represented by its adjacency matrix). *Hint*: see exercise 22.3-12 for undirected connected components with depth-first search.

4. (5 points) Give the runtime in asymptotic notation of BFS2-Connected-Components(G). What would its runtime be if we used instead an adjacency list representation? Comment on the result.

Exercise 2: depth-first search, topological sort and strongly connected components (35 points). Consider the following directed graph $G = (V, E)$:



1. (3 points) Draw the adjacency-list representation of G as in fig. 22.2, with each list sorted in increasing alphabetical order.

2. (2 points) Give the adjacency matrix of G.

3. (5 points) Draw the graph, the adjacency-list representation (with each list sorted in increasing alphabetical order), and the adjacency matrix for the transpose graph G^T.

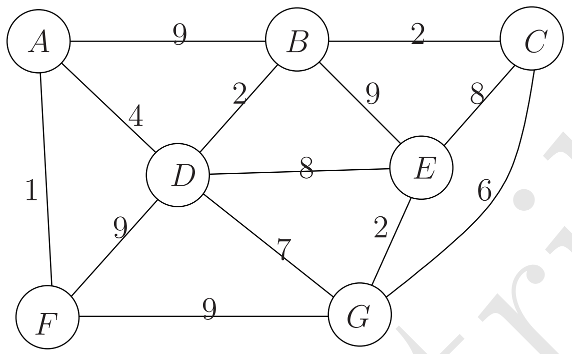
4. (8 points) Do depth-first search in G, considering vertices in increasing alphabetical order. Show the final result, with vertices labeled with their starting and finishing times, and edges labeled with their type (T/B/F/C) as in fig. 22.5(a).

5. (8 points) Based on your results, proceed to run the algorithm in p. 617 to find the strongly connected components of G (show the result of the DFS in G^T, with vertices labeled with their starting and finishing times).

6. (4 points) Draw the component graph G^{SCC} of G. Is it a dag/tree/forest/none of these?

7. (5 points) Find a topological sort of G^{SCC} using the algorithm in p. 613 (label each vertex with its DFS finishing time).

Exercise 3: minimum spanning trees (20 points). Consider the following weighted undirected graph:



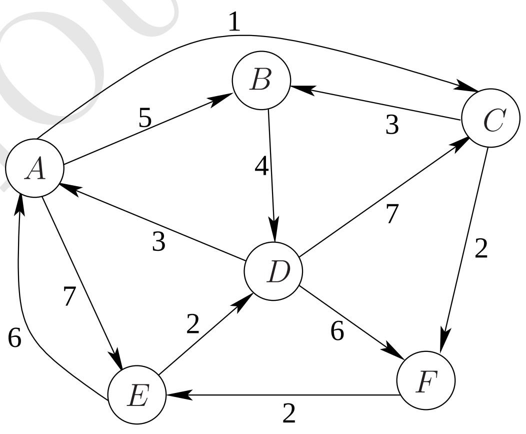
Show the minimum spanning tree obtained by:

1. (10 points) Kruskal's algorithm.

2. (10 points) Prim's algorithm using A as initial vertex.

Do not show any intermediate stages (as in figs. 23.4 or 23.5), just show the final MST. However, for each algorithm, show the list of MST edges in the order selected by the algorithm, e.g. (B, D), (E, A), (E, B), etc.

Exercise 4: shortest paths (25 points). Consider the following weighted directed graph:



1. (13 points) Show the intermediate stages of Dijkstra's algorithm as in fig. 24.6 to find the single-source shortest paths starting from vertex A.

2. (12 points) Consider the same graph but ignoring the weights. Use breadth-first search starting from vertex A to label each vertex with its shortest distance (in number of edges) from A, and mark the edges in the BFS tree. Do not show any intermediate results (as in fig. 22.3 in the book), just show the final BFS tree, labelling the nodes with the shortest-path distances. Also, show the contents of the BFS queue just before F is enqueued.