

EC504 — Fall 2021 -- Homework 4 Solutions

Reading Assignment: Graph Algorithms Chapters 22, 23, 24, 25 and appendix B.4

1. Consider the directed graph shown below in Fig. 1
 - (a) (10 pts) Perform breadth first search (BFS), starting from node 1 and draw the tree on the figure and label each node by the depth of the node, $d(i)$, and its parent (or predecessor), $p(i)$ for $i = 1, 2, \dots |E|$. Give a list of the order in which the nodes leave the queue. **Assume that arcs out of a node will be examined in order of increasing end node; i.e. break ties in order of arc selection by examining the arc with the smallest value of end node first.**
 - (b) (10 pts) For the same directed graph in Fig. 1, use depth first search (DFS), starting from node 1. **Give two lists for the order in which nodes entered into the stack and the order they leave the stack.** (An extra figure is appended to this Homework below.) Draw the stack in enough detail to understand its function. **Again assume that arcs out of a node will be examined in order of increasing end node; i.e. break ties in order of arc selection by examining the arc with the smallest value of end node first.** Is the reverse order a topological sort? (Explain)

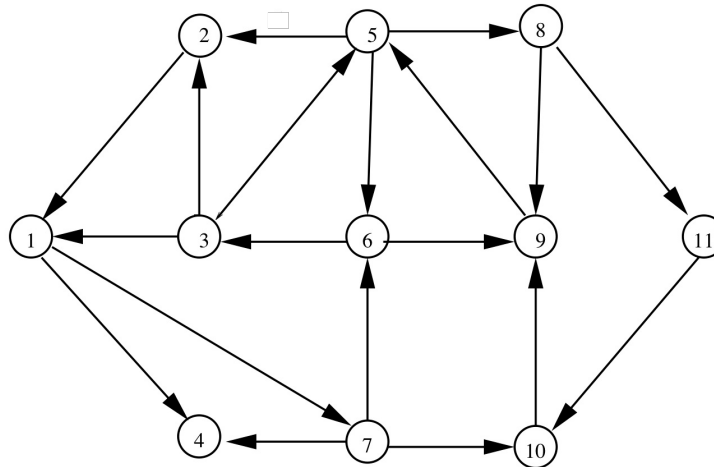


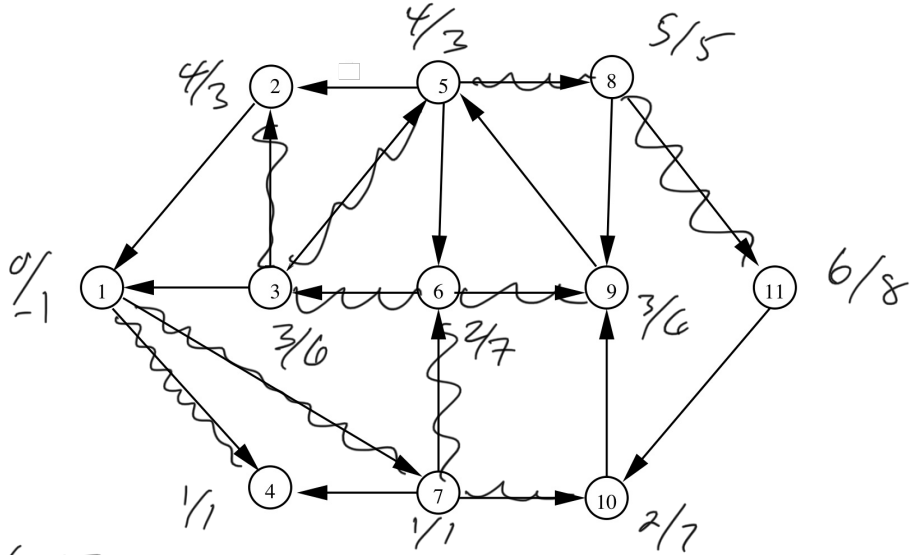
Figure 1

Solution:

Given:

[1 | 4 | 7 | 6 | 10 | 3 | 9 | 2 | 5 | 8 | 11] i.e.

d/p →



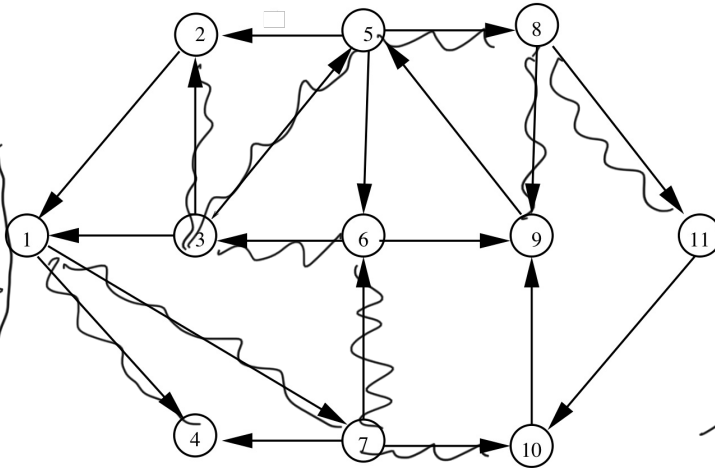
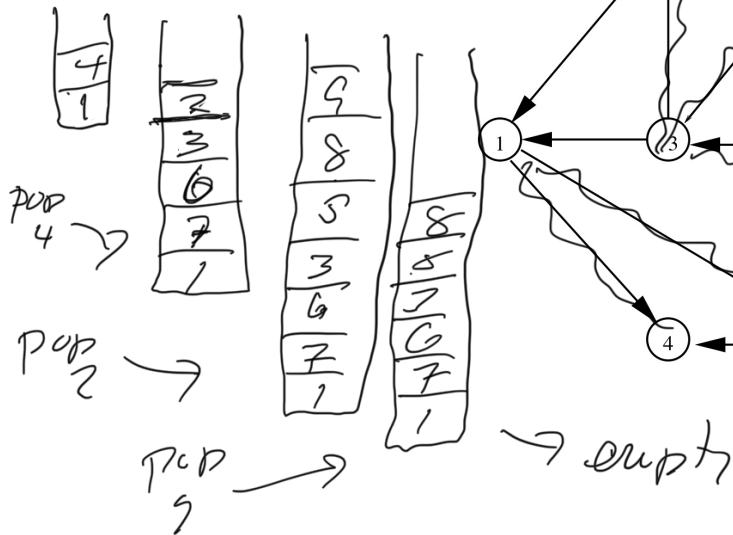
Exit: 1 4 7 10 3 9 2 5 8 11
order

Figure 1

push order 1 4 7 6 3 2 5 8 9 11 IN →

pop order 4 2 9 11 8 5 3 6 7 1 OUT →

Start



There are
cycles so
Not Top Sort

Figure 1

2. (10 pts) Compute a minimum spanning tree for Fig. 2 below, and **evaluate the weight** of that spanning tree using Kruskal's algorithm. Draw the tree and list in order the edges that are added to arrive at the tree. (HINT: You may check this by using Prim's algorithm, which will get the same MST but of course the order of adding edges is very different.)

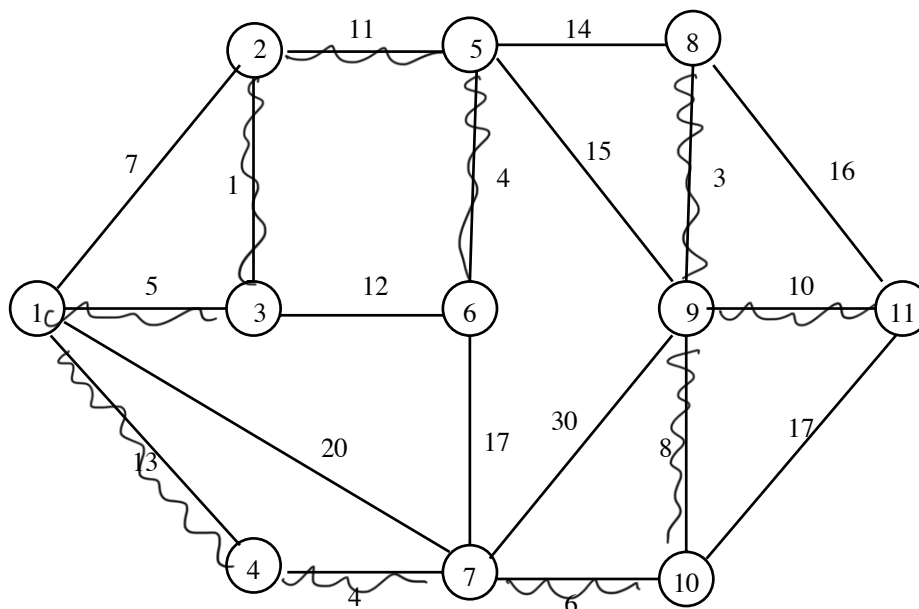


Figure 2:

Kruskal Order

(2,3) 1
 (8,9) 3
 (4,7) 4
 (5,6) 4
 (1,3) 5
 (7,10) 6
 (9,11) 8
 (9,10) 8
 (2,5) 11
 (1,4) 13

Prim's Order (check)

(1,3) 5
 (2,3) 1
 (2,5) 11
 (5,6) 4
 (1,4) 13
 (4,7) 4
 (7,10) 6
 (9,10) 8
 (8,9) 3
 (9,11) 10

3. (20 pts) This is a forward star representation for a directed graph with $|V| = 11$ vertices and $|E| = 16$ edges.

Vertex Number:	1	2	3	4	5	6	7	8	9	10	11	12					
Array First:	{ 1,	3,	4,	5,	7,	8,	12,	12,	14,	14,	15,	17 }					
Edge Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Array Edge:	{ 2,	6,	6,	7,	3,	7,	8,	5,	8,	9,	10,	9,	11,	9,	9,	10,	-1 }
	✓	1	2	3	4	5	6					8	10	11			

(a) Draw the graph on the template in Fig. 3. (HINT: You may want to do part b first.)

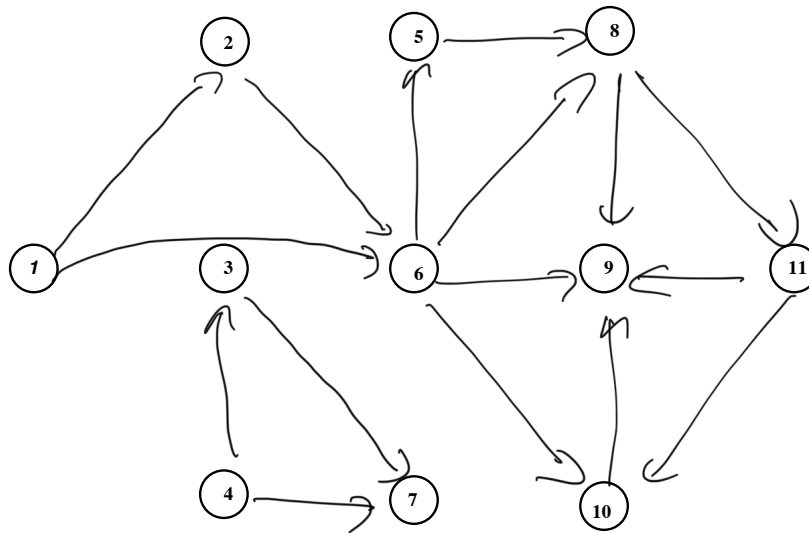


Figure 3:

(b) Represent this graph as an adjacency list.

(c) Is this graph a DAG?

Yes There are No cycles.

4. An undirected bipartite graph $G(V, E)$ is one where its nodes can be partitioned into two disjoint sets $V = V_1 \cup V_2$, such that that every edge $e \in E$ is an arc $\{v_1, v_2\}$, where $v_1 \in V_1$ and $v_2 \in V_2$. Note that, in a bipartite graph, the length of every cycle must be an even number.

Define a pseudo code for an algorithm to determine whether an undirected graph is bipartite with worst case complexity $O(m)$, where m is the number of edges in the graph.

Solution: We will do a variation of breadth-first search.

- (a) Mark all nodes as parity 0. Set all parents of nodes to null. Initialize *Do* queue to empty, *Done* queue to empty.
- (b) While there are nodes of parity 0, select one such node n , set its parity to 1, and perform a breadth first search as follows by adding n to a *Do* queue, first in, first out.
 - While there are nodes in *Do*, remove a node k from *Do*.
 - For each edge $\{k, j\}$ in E such that $\text{parent}(j) \neq k$,
 - If $\text{parity}(j) == \text{parity}(k)$, the graph is not bipartite, so exit and declare not bipartite..
 - else set $\text{parity}(j) = -1 * \text{parity}(k)$. If j is not in *Do* or *Done*, add j to *Do*.
 - Add node k to *Done*.
- (c) end while loop
- (d) If you get here, the graph is bipartite.

The complexity of this algorithm is similar to the complexity of breadth-first search, which is $O(m)$, where m is the number of edges. Each edge may be examined at most twice, once by each node in the edge.

5. While Euler's theorem gives a characterization of planar graphs in terms of numbers of vertices, edges and faces, it is hard to establish whether a graph is planar or not if it is difficult to count faces. There are a couple of other properties of simple, connected planar graphs that derive from Euler's theorem:

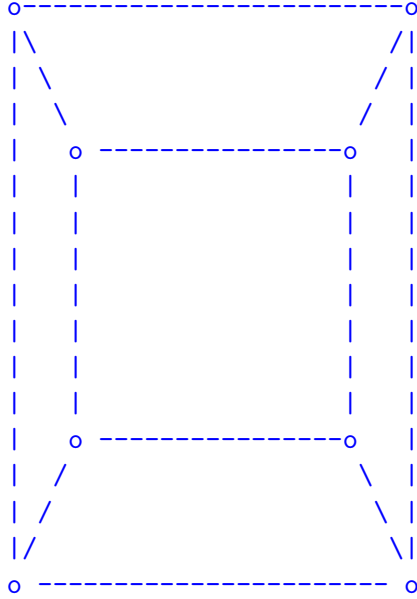
- A simple, connected planar graph with $n \geq 3$ vertices and e edges must satisfy $e \leq 3n - 6$
- A simple, connected planar graph with $n \geq 3$ vertices, e edges and no cycles of length 3 must satisfy $e \leq 2n - 4$

A popular architecture for parallel computers is a hypercube. A hypercube of dimension k , denoted by Q_k , has 2^k nodes, and each node is connected to k other nodes. The nodes can be embedded into a k -dimensional boolean vector, and nodes are connected to other nodes that differ along one of its coordinates. Thus, Q_2 has nodes $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$, and has 4 edges. The node $(0,0)$ is connected to $(0,1)$ and $(1,0)$. Q_3 has nodes $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$, $(1, 1, 0)$, $(1, 1, 1)$. Node $(1,1,1)$ is connected to nodes $(0,1,1)$, $(1,0,1)$ and $(1,1,0)$. Note that the number of edges in a hypercube of dimension k is $k * 2^{k-1}$, since each node has k edges, and we divide by 2 so as not to count the number of arcs twice. Other important facts about hypercubes is that every hypercube is a bipartite graph.

- (a) Using the above facts, verify that Q_3 is a planar graph.

Solution: In a hypercube, all cycles have even length. The hypercube Q_3 has 8 nodes, and 12 edges. This satisfies $12 \leq 16 - 4 = 12$.

Another way of seeing this is that we can draw Q_3 as a planar graph, as follows:



- (b) Using the above facts, show that Q_4 cannot be a planar graph.

Solution: For Q_4 , there are 16 nodes, and there are 32 edges. Thus, $32 > 2(16) - 4$ so Q_4 cannot be a planar graph.

6. Suppose you have computed a minimum spanning tree T for a graph $G(V, E)$. Assume you now add a new vertex n and undirected arcs $E_n = \{\{n, v_i\}, \text{ for some } v_i \in V\}$, with new weights w_{nv_i} . Provide the pseudocode for an algorithm to find the minimum weight spanning tree in the augmented graph $G_a(V \cup \{n\}, E \cup E_n)$. Estimate the complexity of this algorithm.

Solution:

Easy algorithm. Consider the subgraph consisting of the minimum spanning tree T plus the extra arcs E_n . Note that the total number of arcs in this graph is $O(n)$. One can show, by Kruskal's algorithm, that a minimum spanning tree in $G_a = (V \cup \{n\}, E \cup E_n)$ can be found with only the arcs in $T \cup E_n$. Finding a minimum spanning tree in $G_b = (V \cup \{n\}, T \cup E_n)$ can now be done in $O(n \log n)$ with either Kruskal's algorithm or Prim's algorithm.