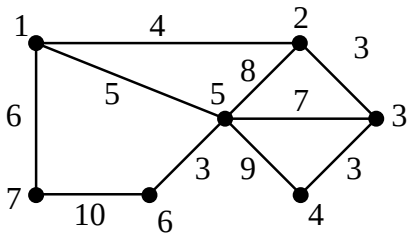
Exercises 4

5 February

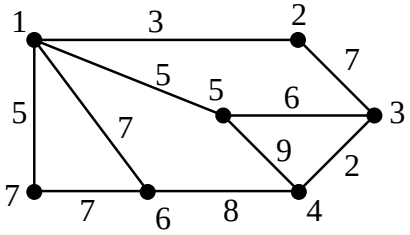
All questions are unassessed.

1.



(a) Use Prim's algorithm, starting at node 1, to find a minimum spanning tree (MST) for the weighted graph shown. State the order in which the nodes are added. Also draw the MST.
(b) Use Kruskal's algorithm to find an MST for the weighted graph shown. State the order in which the arcs are added, giving arcs as pairs of nodes, e.g. $(1,2)$. Also draw the MST.
(c) Does the graph have a unique MST? Justify your answer.
(d) Use Dijkstra's shortest path algorithm to find the shortest path from 1 to 4 in the graph. Give the tree generated by the algorithm. Is it an MST? State the order in which the nodes are added.

2. The same as Q1 for the graph shown.



3. Let G be a graph with n nodes ($n \geq 2$), and where every node has degree at least 3. Show that G has a cycle of length $\leq 2\lceil \log n \rceil$.
[Hint: consider a breadth-first search tree.]

4. Let T be a spanning tree for a graph G. Suppose that an arc a of G joining node x to node y is added to T to form T'.
(a) Show that T' has a cycle C containing a.
(b) Let a' be any arc of C other than a. Remove a' (joining x' to y') from T' to create T''. Show that T'' is a spanning tree for G. [Hint: You must show three things - (1) $\text{nodes}(T'') = \text{nodes}(G)$ (2) T'' is connected (3) T'' is acyclic]

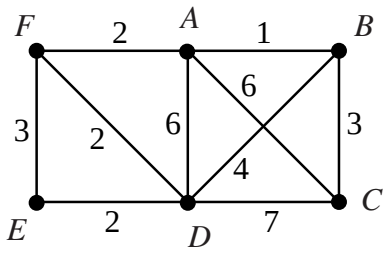
5. [from 2004 exam] Let $G = (G, W)$ be a connected weighted graph, and let T be a minimum spanning tree for G. Let $\text{nodes}(G)$ be partitioned into two nonempty sets X and Y. Call an arc a *crossing arc* if it joins a node in X to a node in Y.
(a) Explain why T must contain a crossing arc.
(b) Show that at least one crossing arc of minimum weight (among all crossing arcs of G) must belong to T.

6. Let (G, W) be a weighted graph where no two arcs of G have the same weight. Show that G has a unique MST. [Hint: Assume for a contradiction that G has two MSTs, T_1 and T_2. Let a be an arc of G *of minimum weight* such that a belongs to one MST but not the other. Then use Q4.]

7. [adapted from Baase and Gelder] Given a weighted graph (G, W) and a start node, a *shortest path tree* is a spanning tree for G such that the tree gives the shortest path to any node from the

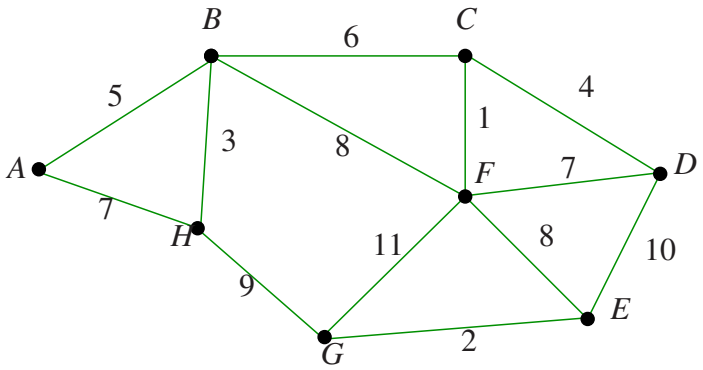
start. Give an example of (G,W) such that neither depth-first nor breadth-first search (from the start node) gives a shortest path tree, no matter what order the adjacency lists are presented in. Note that neither DFS nor BFS as presented in the lectures takes any account of the weights of the arcs.

8. [adapted from Baase and Gelder] The weighted graph shown has three different shortest paths from C to E.



(a) State them as sequences of nodes.
(b) Which one will Dijkstra's algorithm pick, and why?

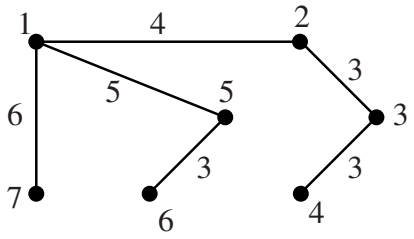
9. Consider the following weighted graph (lectures slide 116).



Use Kruskal's algorithm with union-find (slide 118) and weighted union (on size, as on slide 122) to find an MST for the above graph. Do not use path compression (slide 125). In the case that sizes of equivalence trees are the same, the new leader can be taken to be the node earlier in the alphabet.
State the values stored in the parent array at each stage of the algorithm.

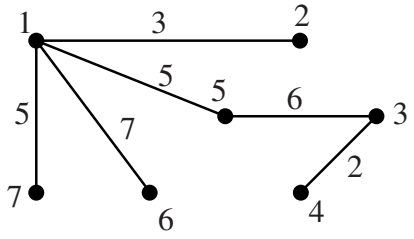
Answers to Exercises 4

1. (a) Prim Nodes added in the order: 1,2,3,4,5,6,7

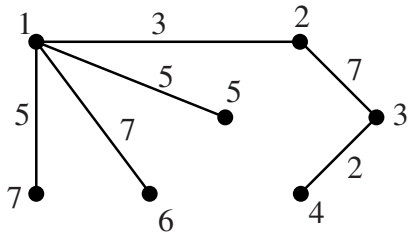


(b) Kruskal. Same MST as (a). Arcs added in the order: (2,3), (3,4), (5,6), (1,2), (1,5), (1,7)
(c) Unique. All STs have the same number of arcs, which is 6. Call the MST obtained T. Any other ST would have to use (at least) one of the arcs not used by T. These have weights $7, 8, 9$ and 10. Since no existing arc of T has weight > 6, this can only give a more costly ST.
(d) Dijkstra. Shortest path $1, 2, 3, 4$ of length 10. Same tree as (a). Note that this is not true in general. See answer to Q2. Nodes added in the order: $1, 2, 5, 7, 3, 6, 4$.

2. (a) Prim. Weight 28. Nodes added in the order: 1,2,5,7,3,4,6 (assuming that we choose numerically least when there are two choices - e.g. between 5 and 7).



(b) Kruskal: arcs added in the order (3,4), (1,2), (1,5), (1,7), (3,5), (1,6). Same MST as in (a).
(c) Not unique: could have (7,6) instead of (1,6).
(d) The shortest path is 1,2,3,4. Length 12.

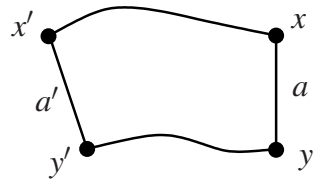


Not an MST - weight is 29. Nodes added in the order 1,2,5,7,6,3,4. The table below shows the successive tree/fringe nodes and values of distance and parent during the computation.

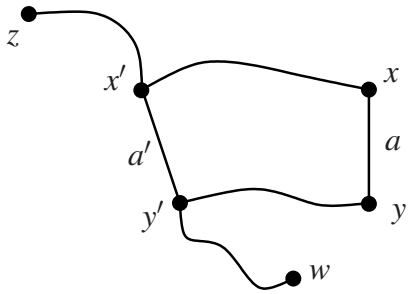
node	t/f	d	p	t/f	d	p	t/f	d	p	t/f	d	p	t/f	d	p	t/f	d	p
1	t			t			t			t			t			t		
2	f	3	1	t	3	1	t	3	1	t	3	1	t	3	1	t	3	1
3				f	10	2	f	10	2	f	10	2	f	10	2	t	10	2
4							f	14	5	f	14	5	f	14	5	f	12	3
5	f	5	1	f	5	1	t	5	1	t	5	1	t	5	1	t	5	1
6	f	7	1	f	7	1	f	7	1	f	7	1	t	7	1	t	7	1
7	f	5	1	f	5	1	f	5	1	t	5	1	t	5	1	t	5	1

3. Take any node s as the start node for breadth-first search. We build the search tree breadth-first, level by level, by putting as children of each node all its neighbours apart from the node we just came from (the parent node).
Unlike in standard BFS, for simplicity and clarity we allow repeated nodes in the tree (by contrast, in standard BFS as in the lectures we keep track of which nodes we have already visited and do not put them into the search tree twice). Each node encountered has at least two successors in the search tree, since every node has degree at least 3. So we are creating at least a full binary tree.
By the time the tree has $n+1$ nodes (say at depth k), by the Pigeonhole Principle there must be a repeated node. Let x be a "first" repeated node, more precisely one where the sum of the depths of the two occurrences of x is a minimum. This means that there are two different ways of getting from s to x, which gives a cycle of length $\leq 2k$.
Hence it remains to show that $k \leq \lceil \log n \rceil$. By depth $\lceil \log n \rceil$ there are $2^{\lceil \log n \rceil + 1} - 1$ nodes in a full binary tree (lecture notes p40). So our tree has at least that number. Now $2^{\lceil \log n \rceil + 1} - 1 = 2(2^{\lceil \log n \rceil}) - 1 \geq 2n - 1 \geq n + 1$ (using $n \geq 2$). So depth $\lceil \log n \rceil$ is good enough to get $n+1$ nodes into the tree.
Remark. We could slightly improve the result by noticing that the root node must have at least 3 children. Hence by depth k we have at least $1 + 3(2^k - 1)$ nodes and to get at least $n+1$ nodes we need $k = \lceil \log(1 + n/3) \rceil$.

4. (a) Since T is connected and includes every node of G, there must be a path P from x to y in T. This path does not use a. So when a is added, P together with a forms a cycle.
(b) Suppose wlog (without loss of generality) that when travelling along P from x to y we reach x' before y'. See diagram.



We must show three things:
(1) nodes$(T'') =$ nodes(G). We know that nodes$(T) =$ nodes(G) since T is a ST for G. We know that nodes$(T') =$ nodes(T) since we are adding an arc between existing nodes. So we just need to be sure that removing a' does not remove x' or y' from T'. But T' contains a path from x to x' and a path from y to y' (see diagram). These paths are still present in T''. Hence nodes$(T'') =$ nodes$(T') =$ nodes(G).
(2) T'' is connected. Take any two nodes z, w. There is a path from z to w in T. If this path does not use a' then it is contained in T'' and we are done. So suppose that the path does use a'. Then there is a path in T'' avoiding a' and using the route $x', \ldots, x, y, \ldots, y'$. See the diagram.

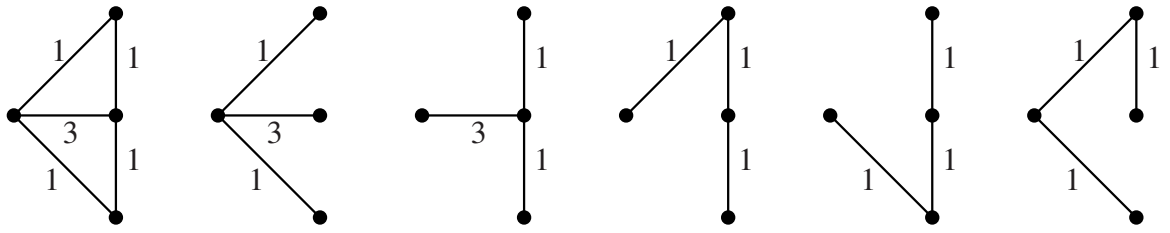


(3) T'' is acyclic. Suppose for a contradiction that T'' has a cycle. Then this cycle must involve the new arc a, since T is acyclic. The remainder of the cycle constitutes a path from x to y already in T. But there is a unique path from x to y in T, since otherwise we would have a cycle in T. And this path uses a', which has now been removed. Hence this path no longer exists in T''. Contradiction.

5. (a) Take a node $x \in X$ and $y \in Y$. Since T is connected, there must be a path from x to y in T. One arc of this path must cross from X to Y.
(b) Suppose for a contradiction that no crossing arc of minimum weight belongs to T. Let $a = (x,y)$ be a crossing arc of minimum weight. Then $a \notin \text{arcs}(T)$. There must be a path from x to y in T. This path crosses from X to Y using arc a'. Clearly $W(a') > W(a)$. Now add a to T, giving a cycle. Remove a' to remove the cycle. We have a spanning tree T', by Q4. But $W(T') < W(T)$. Contradiction of T being minimum.

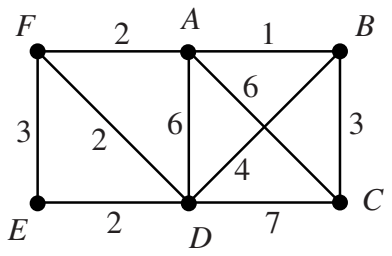
6. Assume for a contradiction that G has two MSTs, T_1 and T_2. Let a be an arc of G of minimum weight such that a belongs to one MST but not the other. Suppose wlog that a belongs to T_1 but not T_2. Then add a to T_2. This creates a cycle. Now some arc a' in the cycle must not belong to T_1, since T_1 is acyclic. Considering T_2 we must have $W(a') \leq W(a)$, since otherwise we could remove a' from T_2 and add in a to get a spanning tree (by Q4) of lower weight, contradicting T_2 being a MST. Since all weights are different we have $W(a') < W(a)$. But this contradicts the minimality of a, since a' belongs to one MST but not the other and has lower weight than a.

7. For instance the graph in (a).

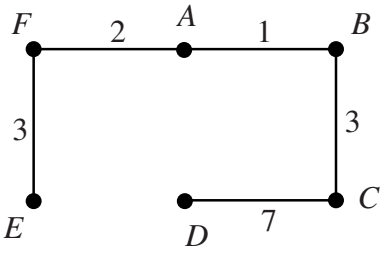


BFS produces (b). DFS produces (c), (d) or (e). The shortest path tree is (f), or the symmetric graph. Note that if we decided to use a variant of DFS where we choose an arc of minimal weight when there is a choice, we would get (d) or (e) and not (c).

8. (a) Shortest paths: (1) $CBAFE$ (2) CDE (3) $CBDE$.



(b) Dijkstra's algorithm will pick (1). The shortest path tree generated by the algorithm is shown in the diagram.
Nodes are added to the tree in the order: C, B, A, F, D, E. This is in increasing order of shortest distance from C. So E is in the fringe with parent F before D is added. This means that E will stick with F as parent even when the alternative of D becomes available - if changing parent offers no advantage the algorithm doesn't change parent.



9. Each column shows the values of parent[x] for $x \in$ Nodes after the arc on the top row is added to the MST.

node\arc		CF	EG	BH	CD	AB	BC	EF
A	A	A	A	A	A	B	B	B
B	B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	B	B
D	D	D	D	D	C	C	C	C
E	E	E	E	E	E	E	E	B
F	F	C	C	C	C	C	C	C
G	G	G	E	E	E	E	E	E
H	H	H	H	B	B	B	B	B

The final overall leader is B.
At each stage only one node (shown in red) has its parent updated. Also each node gets its parent updated at most once during the whole computation.
Note that the tree given by the parent array is not the same as the MST. At the final stage when EF is added to the MST, the parent of E becomes B, since the leader of F is B (though the parent of F remains C).