Chapter 3. Section 2

Prim's Algorithm for Minimum-Weight Spanning Tree

The MWST algorithm was inefficient because each edge might be checked many times to check if it was a lowest weight edge leaving the current tree.

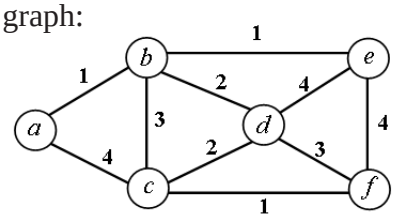
Prim's algorithm avoids rechecking by labeling each vertex x with a pair of values $p(x)$ and $q(x)$:

***p*(*x*)**: the tree vertex having the lowest weight edge to x;
***q*(*x*)**: the weight of this edge.

Algorithm **Prim**
Input: connected graph $G = (V(G), E(G))$ with edge weights in $\mathbb{R}$
Output: a minimum-weight spanning tree T *of* G

 # Initialise tree:
1. Choose $v \in V(G)$, $T := (\{v\}, \emptyset)$
 # Initialise vertex label pairs:
2. **for** $x \in V(G)\backslash\{v\}$ **do** label x with pair $(p(x), q(x))$, where $p(x) := v$ and $q(x) := \infty$
3. $u := v$ *# u records the vertex most recently added to T*
4. **while** $|V(T)| < |V(G)|$ **do**
 # Update vertex labels:
5. **for** each edge ux leaving T from vertex u **do**
6. **if** $w(ux) < q(x)$ **then** $p(x) := u$; $q(x) := w(ux)$
 # Extend the tree:
7. $u :=$ vertex $x \in V(G)\backslash V(T)$ with minimum value of $q(x)$; *# u becomes the new T vertex*
8. $V(T) := V(T) \cup \{u\}$; $E(T) := E(T) \cup \{p(u)u\}$;
return(T)
end

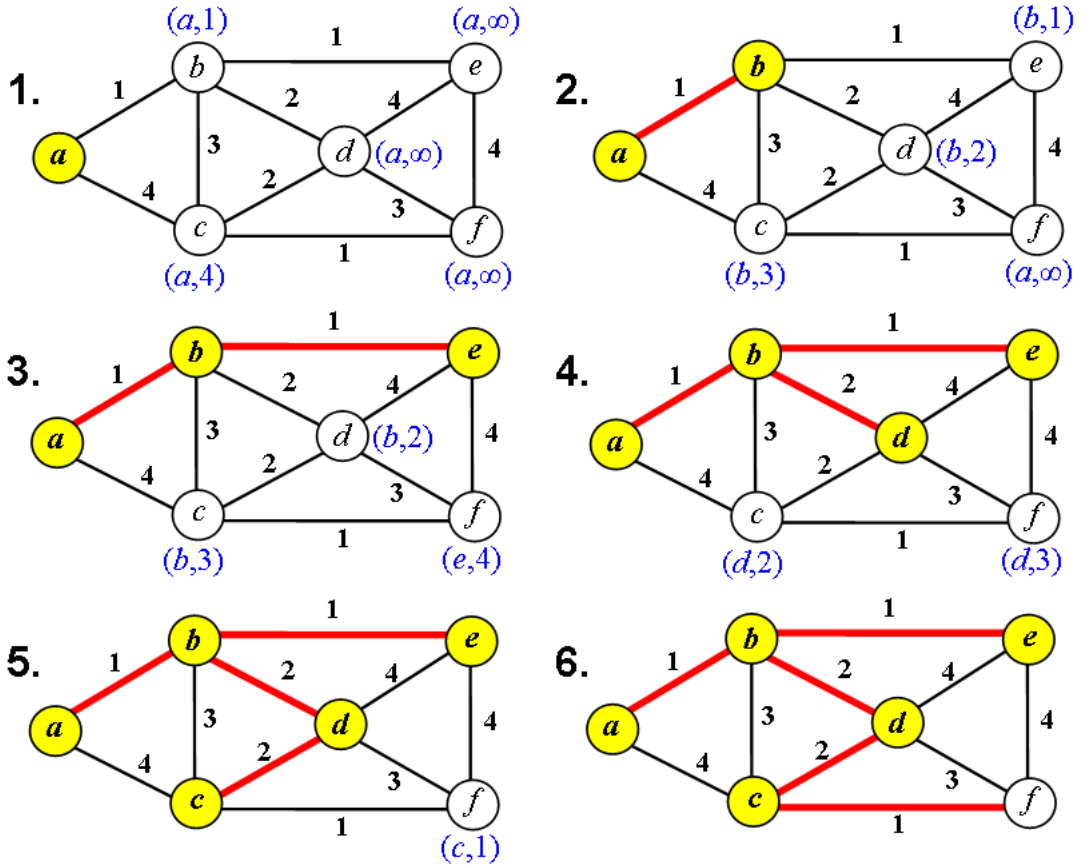
Example: Consider the following graph:



The action of Prim's algorithm is conveniently displayed in a table:

	Vertices:	a	b	c	d	e	f	u	$V(T)$	$E(T)$
	Initialise:		(a, ∞)	(a, ∞)	(a, ∞)	(a, ∞)	(a, ∞)	a	$\{a\}$	$\emptyset$
Iteration										
1	Update labels:		$(a, 1)$	$(a, 4)$	(a, ∞)	(a, ∞)	(a, ∞)			
	Extend tree:		✔					b	$\{a, b\}$	$\{ab\}$
2	Update labels:			$(b, 3)$	$(b, 2)$	$(b, 1)$	(a, ∞)			
	Extend tree:					✔		e	$\{a, b, e\}$	$\{ab, be\}$
3	Update labels:			$(b, 3)$	$(b, 2)$		$(e, 4)$			
	Extend tree:				✔			d	$\{a, b, e, d\}$	$\{ab, be, bd\}$
4	Update labels:			$(d, 2)$			$(d, 3)$			
	Extend tree:			✔				c	$\{a, b, e, d, c\}$	$\{ab, be, bd, dc\}$
5	Update labels:						$(c, 1)$			
	Extend tree:						✔	f	$\{a, b, e, d, c, f\}$	$\{ab, be, bd, dc, cf\}$

For the sake of comparison we also display the steps of Prim's algorithm diagrammatically:



Complexity of Prim's Algorithm

For a graph with n vertices and m edges, Prim's algorithm makes the following steps:

Action		**Steps**
Initialisation:	n steps to label each vertex	n
i-th iteration:	approx. $d(u)$ steps to relabel each neighbour of u	$d(u)$
	a maximum of $n-i$ steps to locate smallest $q(x)$ value	$n-i$
Total:		$n + \sum_{u \in V(G)} d(x) + \sum_{i} (n-i)$
	(note the use of the Handshaking Lemma!)	$= n + 2m + \frac{1}{2}n(n-1)$
		$= O(n^2 + m)$

For a graph with many edges, i.e. $m = O(n^2)$, Prim's algorithm is **linear** in input size: the input size is $O(n+m) = O(n+n^2) = O(n^2)$, Prim takes $O(n^2+m) = O(n^2)$ steps. If there are fewer edges then the n^2 is the main contribution to the running time. A reduction can be achieved by maintaining an ordered list of $q(x)$ values so that the minimum value can always be selected in constant time. Such refinements are beyond the scope of this book but can reduce the running time of Prim's algorithm to $O(m + n \log n)$ time.

Correctness of Prim's Algorithm

We are still just applying Minimum-Weight_Spanning-Tree: all we are doing is controlling the way in which the next edge is added to the spanning tree. So the proof of correctness carries over and guarantees that Prim's algorithm will produce an MWST. We should convince ourselves that, when we extend the tree with the instruction $E(T) := E(T) \cup \{p(u)u\}$, the edge $p(u)u$ does actually belong to $E(G)$. This is always the case because label pairs only updated when we are checking an edge ux, and then $p(x)$ is set to the endpoint u of this edge. So $p(x)x \in E(G)$ whenever $(p(x), q(x))$ is the label of x and $q(x) < \infty$.