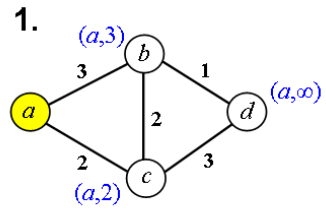
Chapter 4. Section 2

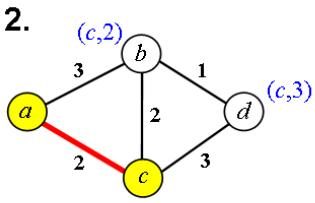
Dijkstra's Algorithm: Analysis

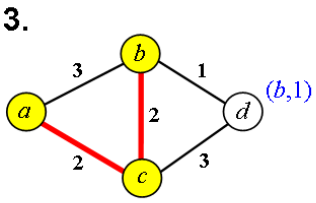
Minimum total weight tree vs minimum path weight tree

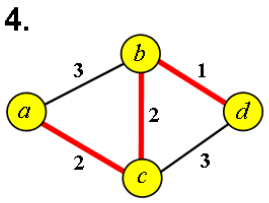
Since they are so similar, it is useful to compare Prim's Algorithm with Dijkstra's on the same input graph:

Prim's Algorithm:

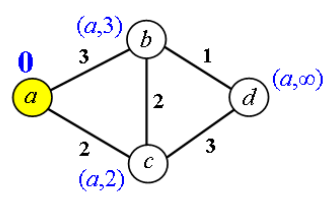


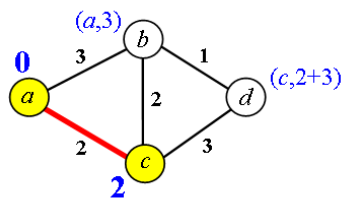


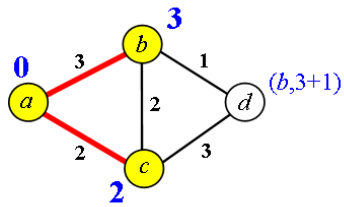


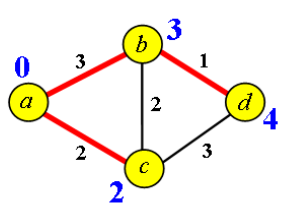


Dijkstra's Algorithm:







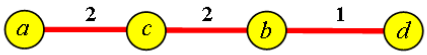


Both algorithms start off with the same update from a.

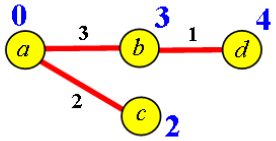
The first edge chosen will also necessarily be the same.

But now a cheapest leaving edge, chosen by Prim's, does not give the cheapest route to b.

The algorithms have now diverged although, by coincidence, they both choose the same final edge.



Output of Prim



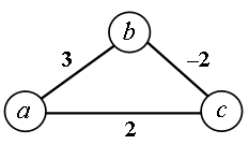
Output of Dijkstra

Note the extra output consisting of distances $d_G(a, x)$ for each vertex x (often presented as a table).

The Prim tree in our example does not give the shortest path from a to d. The Dijkstra tree has weight 6 which is not minimum.

Negative edge weights

We apply Dijkstra's algorithm to a second example. This time for contrast we record the iterations of the algorithm in tabular form, on the right



	a	b	c	u	tree edges
Initialise:		(a, ∞)	(a, ∞)	a	$\emptyset$
Iteration					
1 Update labels:		$(a, 3)$	$(a, 2)$		
Extend tree:			**2**	c	$\{ac\}$
2 Update labels:		$(c, 2-2)$			
Extend tree:		**0**		b	$\{ac, ab\}$

The presence of negative weight edges is handled automatically **but incorrectly** by Dijkstra's Algorithm. The output records a distance from a to c of 2, but the actual distance, via b, is $d_G(a, c) = 1$.

We conclude that we cannot search for minimum-weight paths **greedily** when negative weights are

present. At least not using the tree growing approach. If Dijkstra could be applied with negative weights then we could find **maximum-weight paths** in graphs using the same approach as for maximum-weight spanning tree (Chapter 3, Section 1): replace each weight $w(e)$ by $-w(e)$ and minimise. But, in fact no non-exhaustive method is known for maximum-weight paths either, even if all the weights are 1. However, sophisticated modern algorithms have been found that can significantly outperform Dijkstra and handle negative weights. Start at this Ben Brubaker article for Quanta magazine and follow the onward links.

Correctness of Dijkstra

Since Dijkstra builds a tree in the same way as Prim we can assume it will correctly produce a spanning tree of G. We must prove that each new vertex u that is assigned to the tree T is being given the correct value of $d_G(v, u)$, where v is the chosen root vertex.

At each iteration of the algorithm we add to our tree T an edge, say $p(y)y$, where $p(y)$ and y are chosen to minimise

$$q(y) = d_G(v, p(y)) + w(p(y)y).$$

Suppose $p(y)$ is vertex x in T. We must prove that

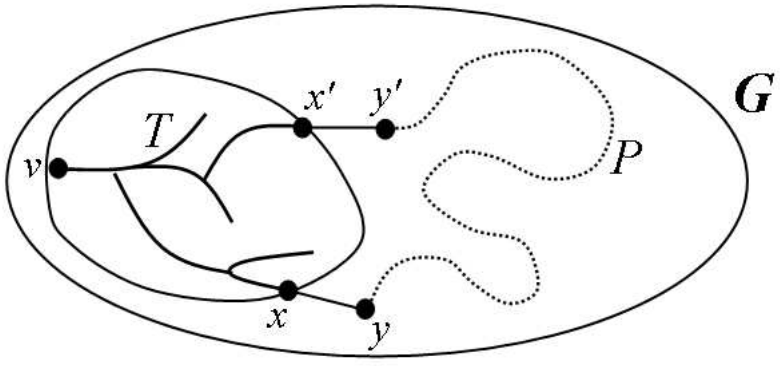
$$\mathrm{d}_G(v, y) = \mathrm{d}_G(v, x) + w(xy).$$

Suppose for a contradiction that $\mathrm{d}_G(v, y) < \mathrm{d}_G(v, x) + w(xy)$. Then there is some path P in G from v to y with total edge weight $w(P)$ less than $\mathrm{d}_G(v, x) + w(xy)$. Now P must leave T by some edge since $v \in T$ and $y \notin T$. Suppose this edge is $x'y'$. We have

$$\begin{aligned} w(P) &\geq \mathrm{d}_G(v, x') + w(x'y') \quad \text{since } P \text{ contains at least these edges} \\ &= q(y') \leq q(y) \quad \text{since } q(y) \text{ was chosen to be minimum} \\ &= \mathrm{d}_G(v, x) + w(xy) \quad \text{by definition of } q(y) \end{aligned} \tag{1}$$

which contradicts our assertion that $w(P)$ was less than $\mathrm{d}_G(v, x) + w(xy)$. So $\mathrm{d}_G(v, y) = \mathrm{d}_G(v, x) + w(xy)$ as required. □

The situation described in the proof is illustrated below (this drawing playing no part in the actual proof, of course).



This proof must contain an error if negative weights are allowed, but there seems to be no assumption made that weights are nonnegative. Here the drawing does help: looking at P we can imagine some edge on P between y' and y which has large negative weight. Then inequality (1)in the proof need no longer hold: this negative weight may reduce $w(P)$ below the value of $\mathrm{d}_G(v, x') + w(x'y') = q(y')$.

Complexity:

Dijkstra's algorithm carries out exactly the same steps as Prim's, just with different vertex labels. So the analysis given in the Chapter 3, Section 2 applies here: the complexity is $O(|E| + |V|^2)$.