Prim's Algorithm — Detailed Explanation, Proof Sketch, and a Challenging Example

Generated for: Study / Assignment

August 24, 2025

Overview

Prim's algorithm is a greedy algorithm that finds a Minimum Spanning Tree (MST) of a connected, undirected, weighted graph. Starting from an arbitrary vertex, it grows a tree by repeatedly adding the least-weight edge that connects a vertex in the tree to a vertex outside the tree (i.e. a cheapest edge crossing the cut). The algorithm terminates when all vertices are included.

Pseudocode (using a priority queue)

Algorithm 1 Prims algorithm (priority-queue implementation)

procedure $\textsc{Prim}(G = (V, E), w(\cdot), s)$ ▷ s = start vertex
  $S \leftarrow \{s\}$ ▷ vertices already in the tree
  $MST \leftarrow \emptyset$
  For each $v \in V \setminus \{s\}$ set $key[v] \leftarrow +\infty$, $parent[v] \leftarrow$ null
  For each edge (s, v) set $key[v] \leftarrow w(s, v)$ and $parent[v] \leftarrow s$
  Insert all vertices $v \in V \setminus \{s\}$ into a min-priority queue keyed by $key[v]$
  while $S \neq V$ **do**
    $u \leftarrow$ extract-min from priority queue (vertex with smallest key)
    Add edge $(parent[u], u)$ to MST and add u to S
    for each edge $(u, v) \in E$ with $v \notin S$ **do**
      if $w(u, v) < key[v]$ **then**
        $key[v] \leftarrow w(u, v)$
        $parent[v] \leftarrow u$
        Decrease-key of v in the queue
      end if
    end for
  end while
  return MST
end procedure

Correctness (Sketch)

At each step Prim's algorithm selects a minimum-weight edge that crosses the cut $(S, V \setminus S)$ where S is the set of vertices already included in the growing tree. By a standard cut argument (similar to Kruskal's correctness proof):

- Take any minimum spanning tree T^*. If T^* does not contain the chosen light edge e crossing the cut, then there exists an edge in T^* crossing the cut; replacing that edge with e (which is no heavier) yields another spanning tree with no greater total weight. Thus there always exists an MST containing e.
- Repeating this invariant shows every edge Prim picks can be extended to some MST. Hence the final tree is an MST.

Time Complexity

Using a binary heap for the priority queue: the algorithm runs in $O((V+E)\log V) = O(E\log V)$ time for connected graphs. Using a Fibonacci heap reduces the complexity to $O(E + V\log V)$.

1 Discussion and variants

- If multiple equally light edges cross the cut, Prim may choose any; different choices may lead to different but equally optimal MSTs.
- For dense graphs an adjacency-matrix + simple $O(V^2)$ selection (no heap) is often faster in practice and simpler to implement.
- Prim is particularly efficient in implementations where decrease-key is fast (Fibonacci heaps) or when using specialized data structures (e.g. for planar graphs or geometric MST approximations).

2 Exercises (challenge)

1. Modify edge weights slightly so that a tie appears at Step 3 between (A, B) and (D, E). Run Prim and show both possible MSTs.
2. Implement Prim on the example using an indexed binary heap and produce the evolution of the `key[]` array after each step.
3. Show how the same MST would be produced by Kruskal's algorithm and compare the order in which edges are selected.

A challenging example (step-by-step)

We present a moderately complex graph with 8 vertices and many edges so the greedy choices and updates are visible. We start Prim at vertex A. The graph and edge weights are chosen to illustrate how the cut and keys evolve.

Graph description

Vertices: A, B, C, D, E, F, G, H. Edges (undirected) with weights:

$$\begin{aligned} &A{-}B: 4, \quad A{-}D: 3, \quad A{-}H: 7, \\ &B{-}D: 2, \quad B{-}C: 6, \quad B{-}E: 5, \\ &C{-}F: 2, \quad C{-}G: 3, \\ &D{-}E: 4, \quad D{-}H: 6, \\ &E{-}F: 1, \quad E{-}G: 6, \\ &F{-}G: 5, \quad H{-}C: 8. \end{aligned}$$

This graph is connected and contains several cycles and tie-like situations that make the algorithm's maintenance of ‘key‘ values meaningful.

Initial picture

Step-by-step execution (start at A)

We run Prim starting from vertex A. At each step we show the selected edge, the updated set S, and the cumulative weight.

Step	Edge added	New S	Running total weight
0	(start) $S = \{A\}$	$\{A\}$	0
1	(A, D) weight 3	$\{A, D\}$	3
2	(D, B) weight 2	$\{A, D, B\}$	5
3	(D, E) weight 4	$\{A, D, B, E\}$	9
4	(E, F) weight 1	$\{A, D, B, E, F\}$	10
5	(F, C) weight 2	$\{A, D, B, E, F, C\}$	12
6	(C, G) weight 3	$\{A, D, B, E, F, C, G\}$	15
7	(D, H) weight 6	$\{A, D, B, E, F, C, G, H\} = V$	21

Table 1: Sequence of edges Prim chooses when started from A.

Illustrative diagrams for the steps

We provide a small sequence of figures. Selected edges (already in MST) are drawn thick and in **red**; the current candidate crossing edges are dashed. Each figure is labelled by the step number.

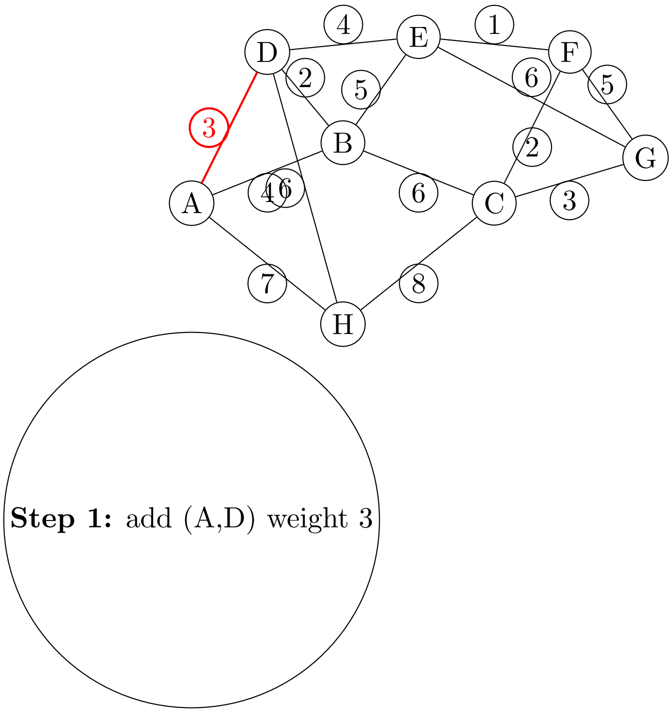

Figure 2: After adding (A, D).

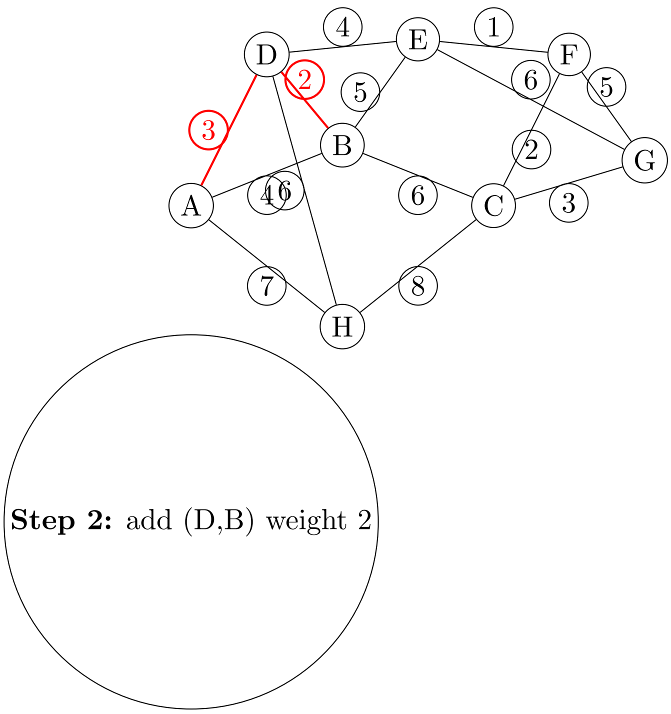

Figure 3: After adding (D, B).

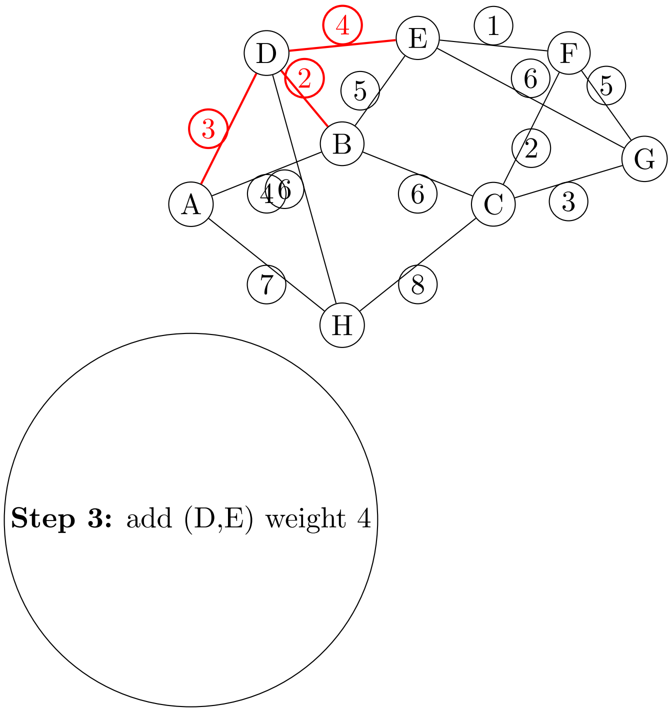

Figure 4: After adding (D, E).

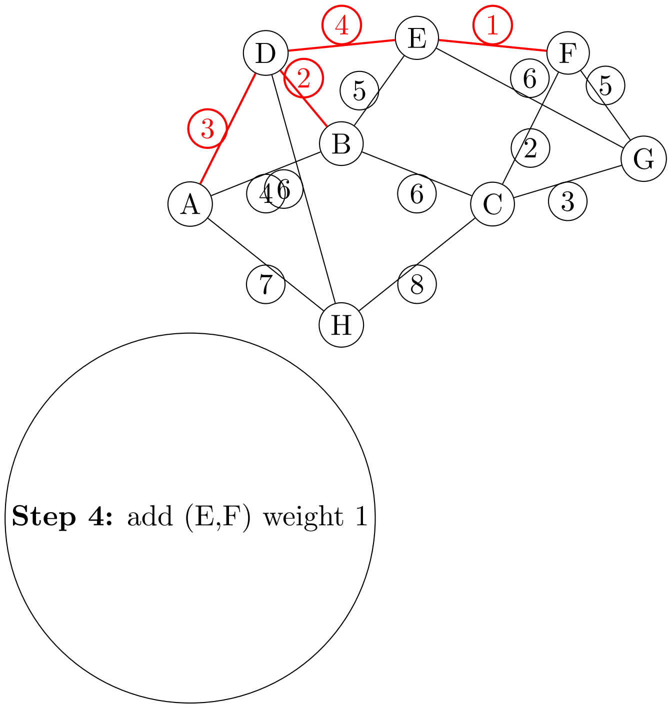

Figure 5: After adding (E, F).

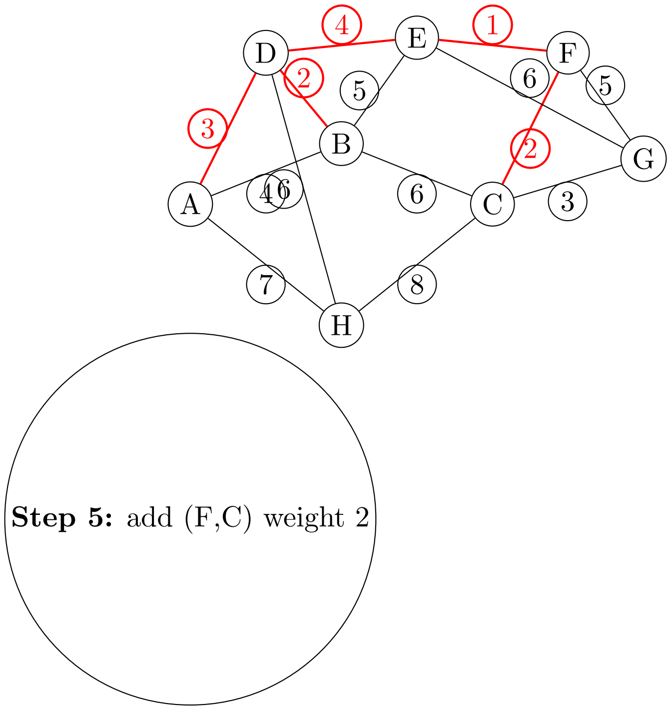

Figure 6: After adding (F, C).

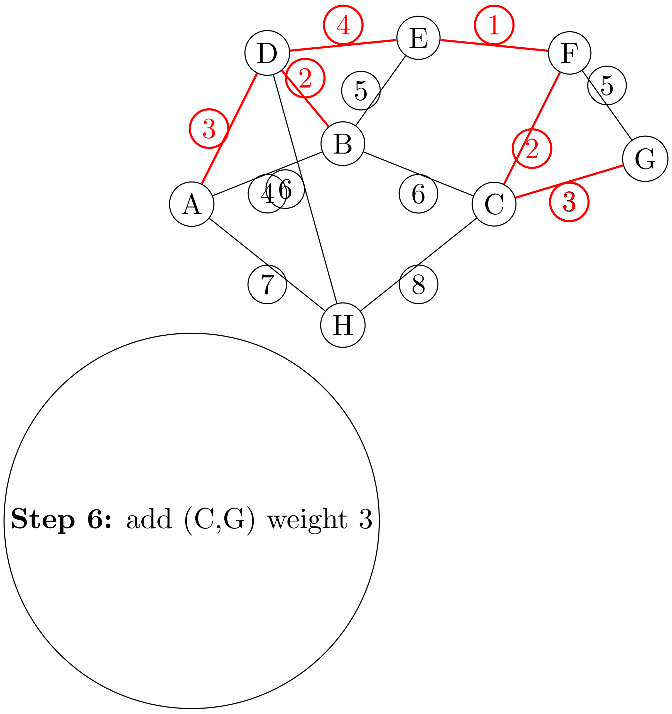

Figure 7: After adding (C, G).

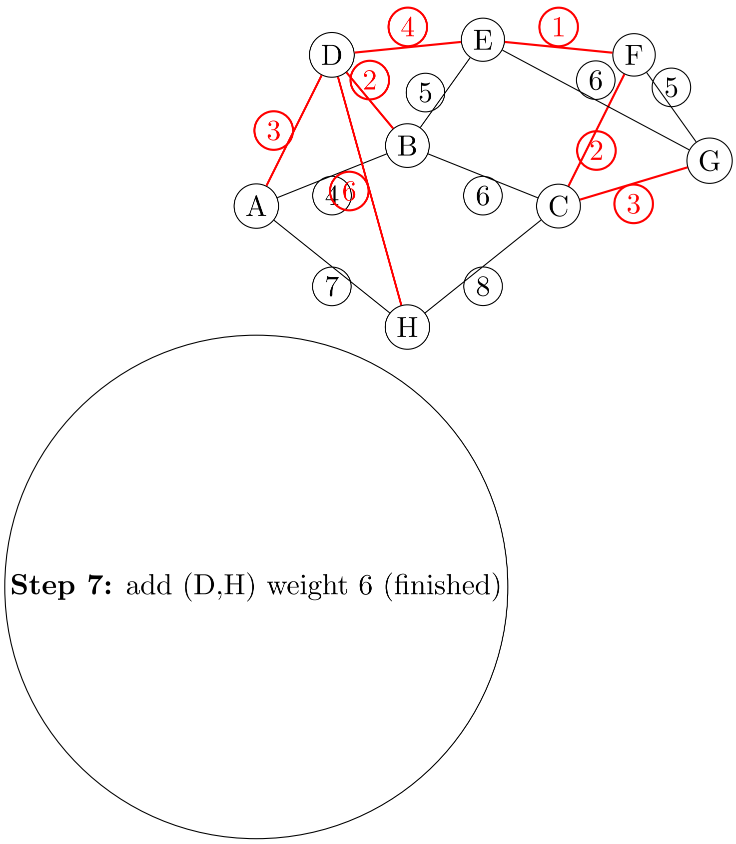

Figure 8: Final MST constructed by Prim (total weight 21).