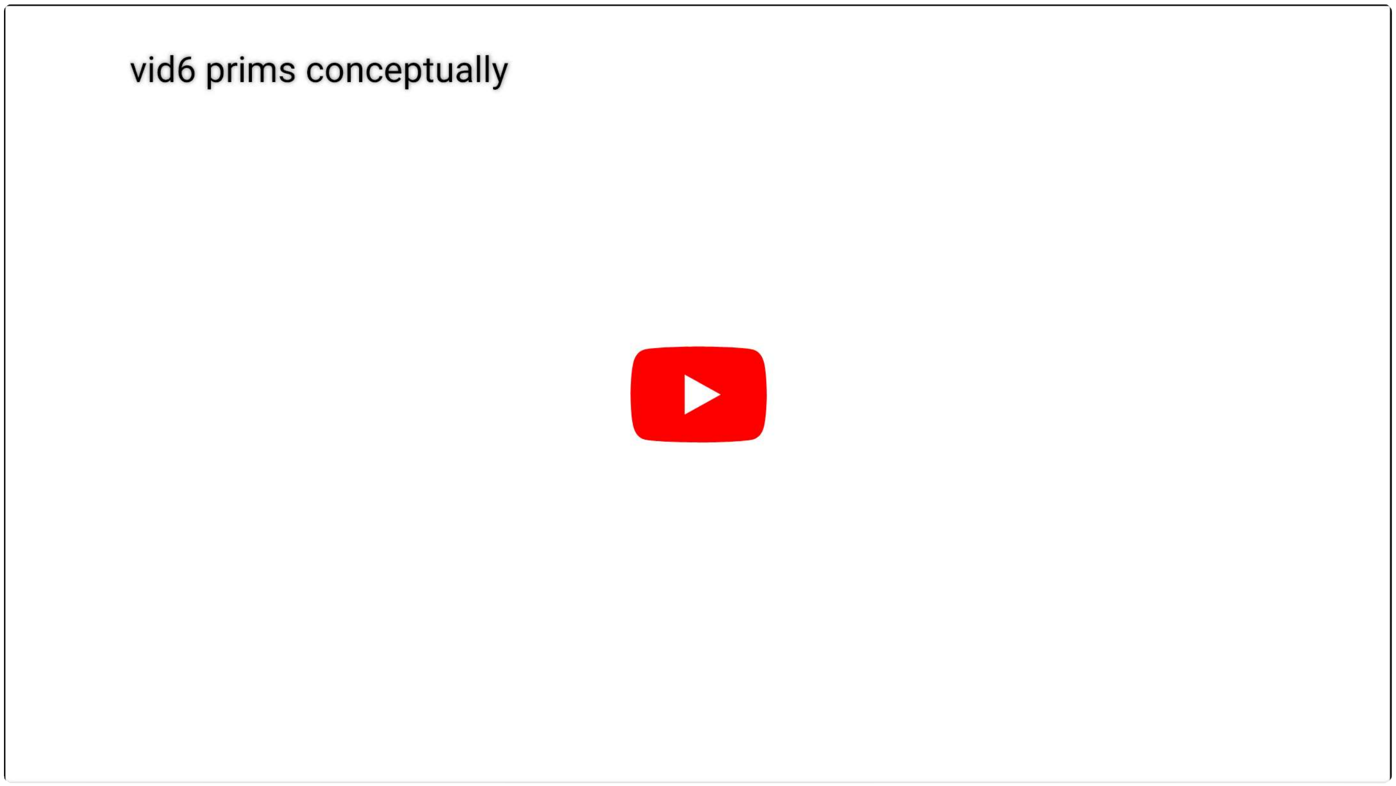
25.2 Prim's Algorithm

Finding MST.

One way to find the MST of a graph is the Prim's Algorithm. In this section, we will discuss both the conceptual and concrete implementation of this algorithm, as well as its runtime analysis.

Conceptual Visualization



Prim's Algorithm is one way to find a MST from a graph. It is as follows:

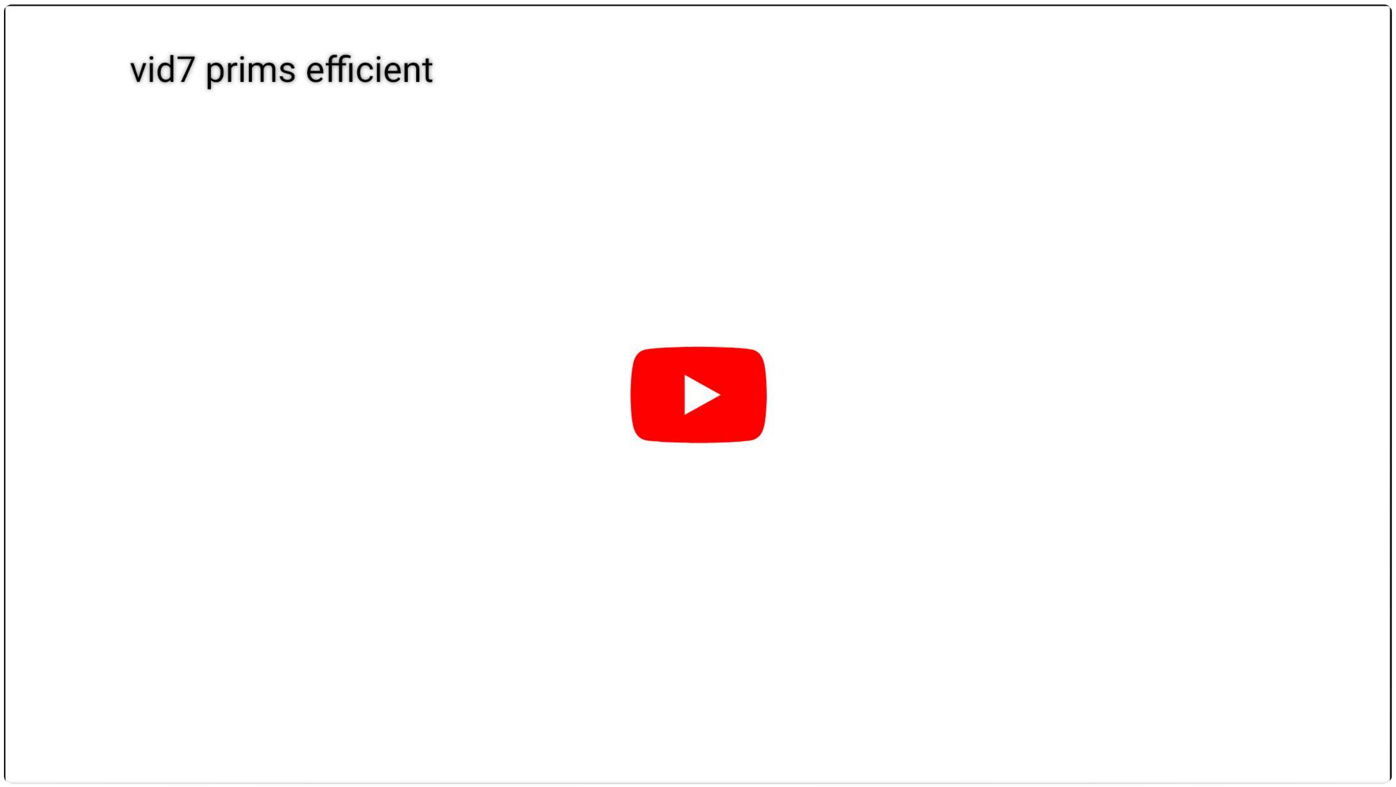
1. Start from some arbitrary node.
2. Repeatedly add the shortest edge that has **one node inside the MST in construction.**
3. Repeat until there are V - 1 edges.

Why does it work?

Prim's algorithm works because at all stages of the algorithm, we can reason as follows:

- Consider dividing all the nodes in the graph into two sets:
    - Set 1: nodes that are part of the existing MST that's under construction
    - Set 2: all other nodes
- According to the algorithm, we always add the **lightest, or minimally weighted, edge** that crosses this cut.
- By **the Cut Property**, the added edge is necessarily part of the final MST.

Implementation



Essentially, this algorithm runs via the same mechanism as Dijkstra's algorithm.

The only difference is that while Dijkstra's considers candidate nodes by their distance from the source node, Prim's looks at each candidate node's **distance from the MST under construction.**

Runtime Analysis

Because this algorithm runs through the same mechanism as Dijkstra's algorithm, its runtime is also identical to Dijkstra's:

$$O((|V| + |E|)log(|V|))$$

Remember, this is because we need to add to a priority queue fringe once for every edge we have, and we need to dequeue from it once for every vertex we have.

Previous
25.1 MSTs and Cut Property

Next
25.3 Kruskal's Algorithm

Last updated 1 year ago