1. B and D, with priorities 3.0 and 9.0 respectively.

2. B and D, with priorities 3.0 and 9.0.

3. Dijkstra's algorithm assigns priorities based on distance from the source, whereas Prim's uses distance from the tree.

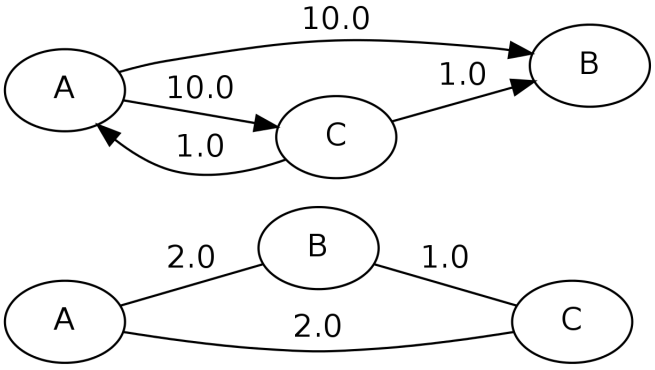
4. A vertex is only put into the PQ when the edge is successfully relaxed. Since the distTo[A] is going to be smaller than distTo[E] + 1, the relaxation will fail.

5. True, because all vertices of distTo[k] point only at vertices of distance k+1. k+2 distance vertices cannot be uncovered until at least one of the distance k+1 vertices has been removed.

6. No, because of the proposition above. Dijkstra's will be alittle slower, since it builds an unnecessary PQ, but this is unlikely to have a real impact, since V + E log V is about the same as E + V.

7. Just run Dijkstra's algorithm, it can handle cycles just fine. If there were negative edges, then we have negative cycles, which are hard to deal with.

8. No. Consider the graphs below for the directed and undirected cases respectively.



9. It is an unweighted graph that is never explicitly built in its entirety (since it is exponential in size). It is an entirely different graph algorithm, where edges were considered according to some special heuristic based on the to-vertex.

10. Dijkstra's, acyclic graph algorithm, Bellman-Ford. Use acyclic graph algorithm on DAGs. Dijkstra's if there are no negative edges. Bellman-Ford if there are negative edges but no cycles. Choice is based on picking the fastest algorithm for the given graph.