* Reminder: A component is a maximum sub-graph.
* Prim’s Algorithm to Compute MST:
  + Starting with any vertex, say vertex 0, and greedily grow tree T. At each step, add the cheapest edge that has exactly one endpoint in T.
  + T represents the MST, and initially is empty.
  + S is the set of vertices in current T.
  + T and S represent the same thing, T has the edges and S has the vertices.
  + Candidate Edges: The edges that have exactly one endpoint.
  + Steps:
    - Find the candidates.
    - Identify the weight of each of the candidate edges. Then pick the one with the cheapest (smallest) weight.
    - Add the cheapest edge to T. Add the new vertex of that cheapest edge into S (the vertex that was not already in S).
    - Example of T and S: (“After step 2 in the lecture” / “2 iterations of this process”)
      * T = { e(0, 2), e(0, 7) }
      * S = { 0, 2, 7 }
  + Prim’s Algorithm correctness proof:
    - Let S be a set of vertices in current tree T.
    - Prim adds the cheapest edge e with exactly one endpoint in S.
    - e is in the MST (due to cut property).
  + Implementation:
    - How to find the cheapest edge with exactly one endpoint in S?
      * Answer 1: Brute force (exhaustive search). Try all edges.
        + Linear-search, O(E), per each spanning tree edge (per each step of Prim’s Algorithm).
        + Overall time complexity for Prim’s Algorithm, using this solution: O(V\*E) == O(n2).
      * Answer 2: Maintain a Priority Queue (min-heap) of vertices connected by an edge to S.
        + Time complexity:

Log(V), per step of Prim’s Algorithm.

Overall for Prim’s Algorithm, using this solution: O(E \* log(V)).