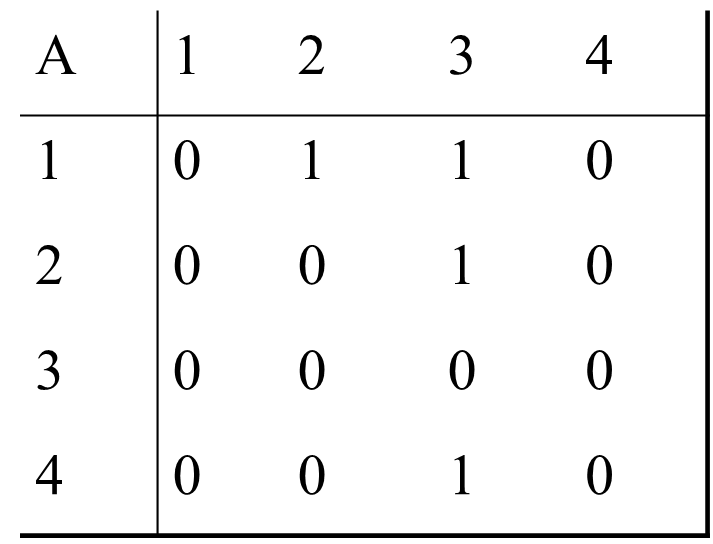
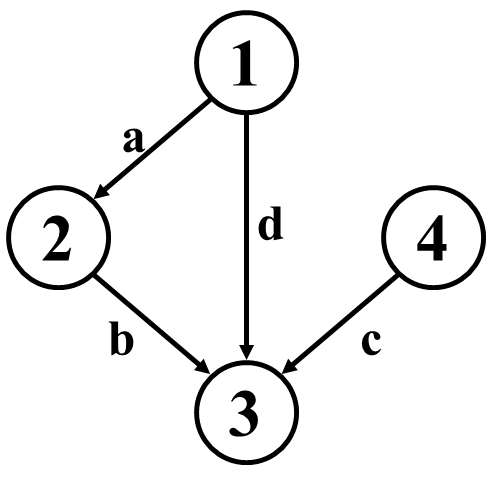
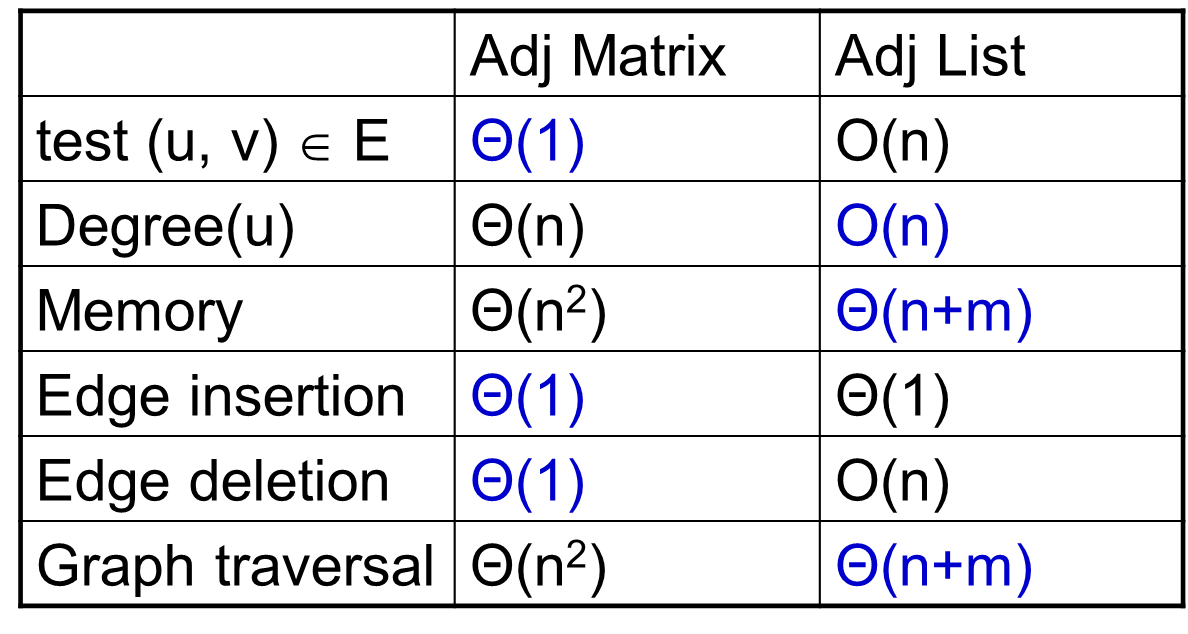
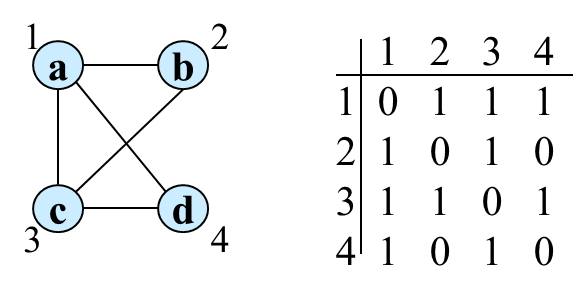
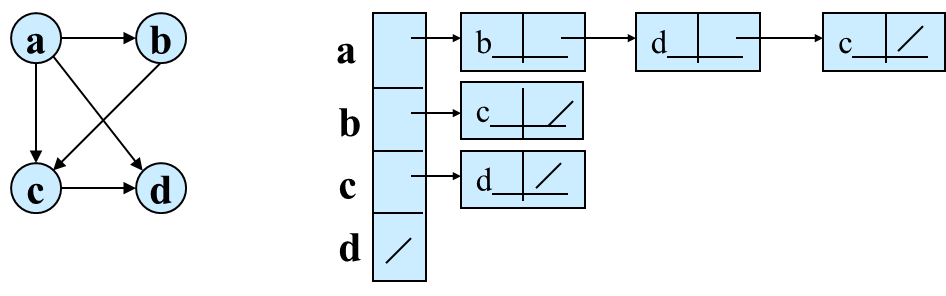
**Lecture 9: BFS**

* **Adjacency Matrices:**

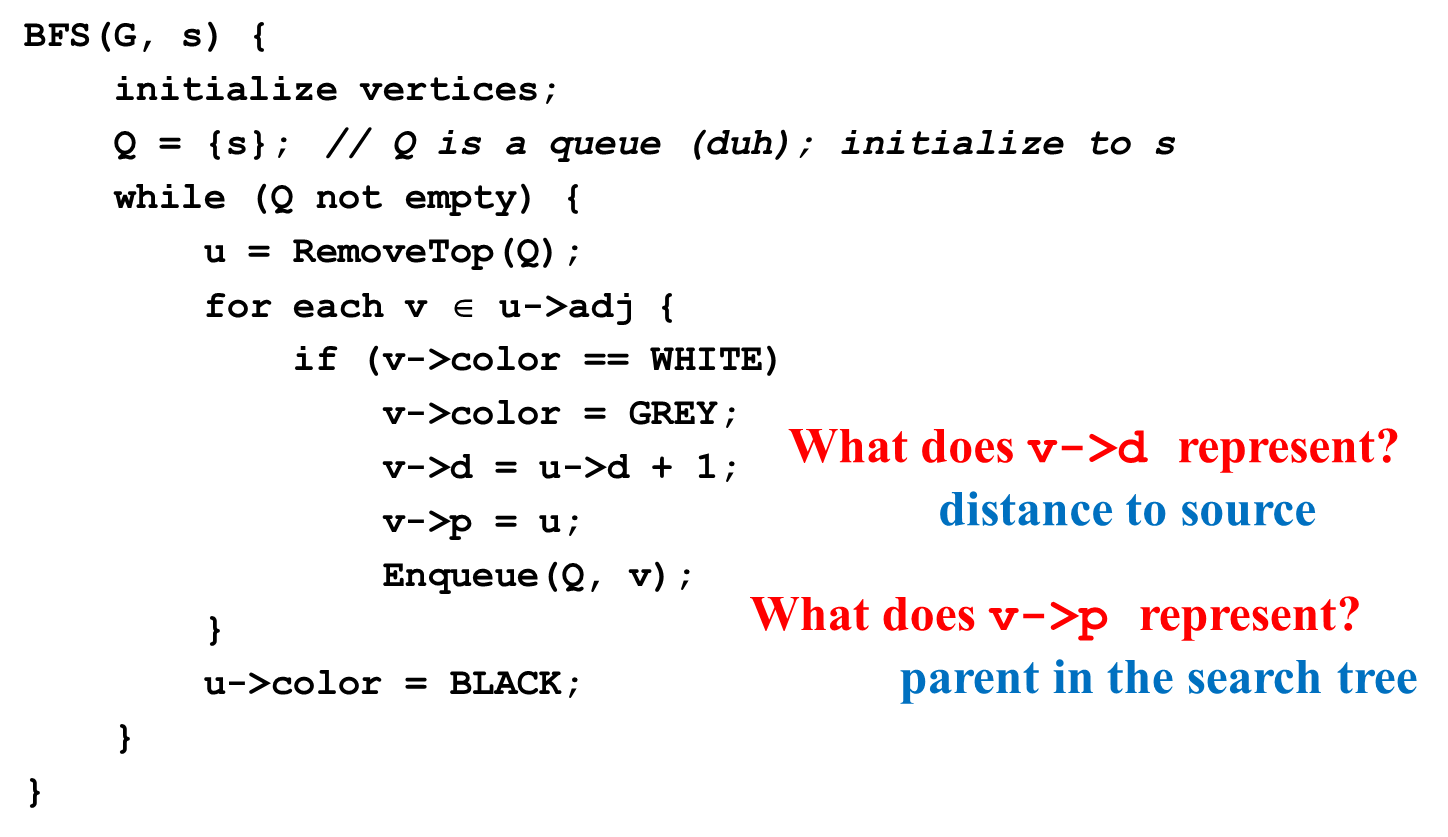
 

* **Adjacency Lists:**

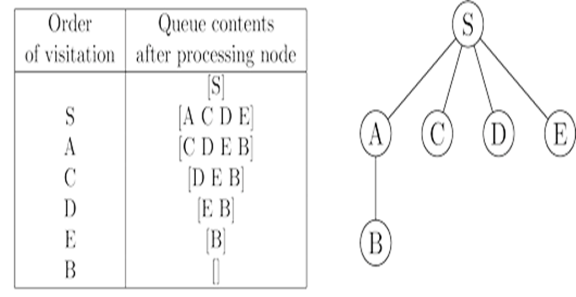
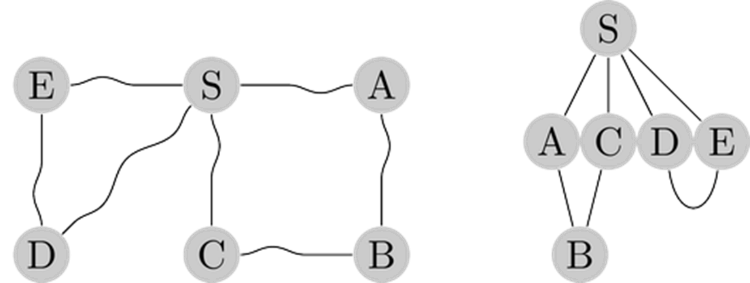
 A screenshot of a graph

Description automatically generated with low confidence

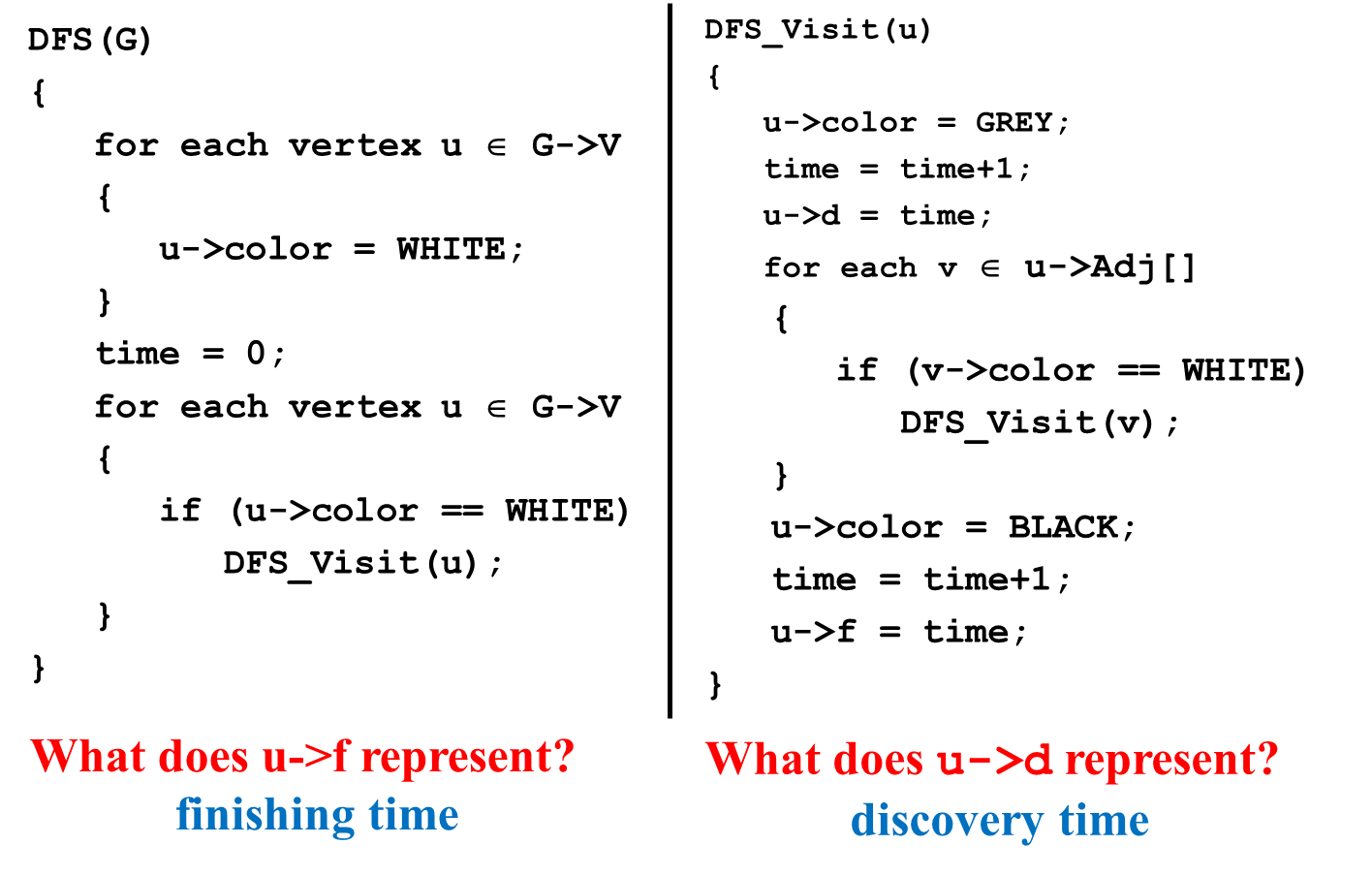
* **BFS:**



* BFS calculates the *shortest-path distance* to the source node
  + Shortest-path distance δ(s,v) = minimum number of edges from s to v, or ∞ if v not reachable from s
  + BFS builds *breadth-first tree*, in which paths to root represent shortest paths in G
  + Thus can use BFS to calculate shortest path from one vertex to another in O(V+E) time

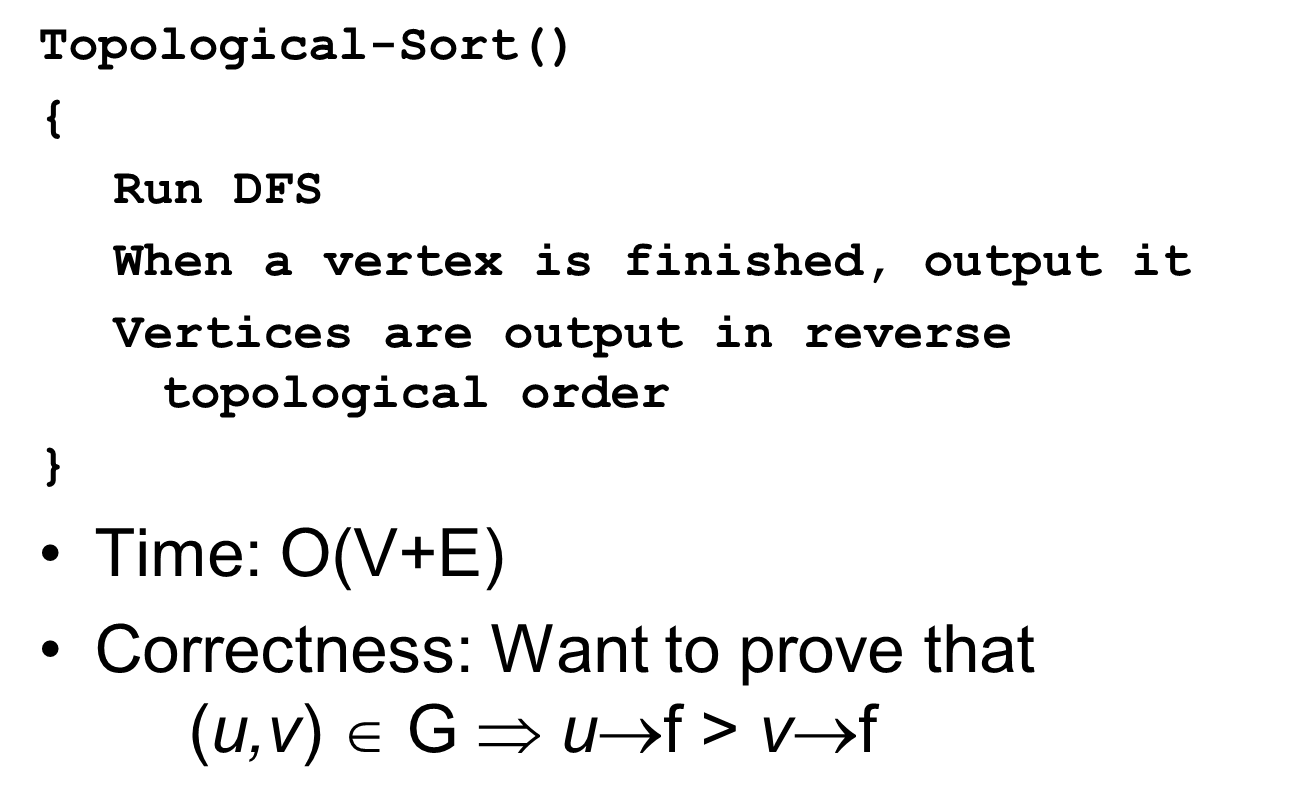


**Lecture 10: DFS**

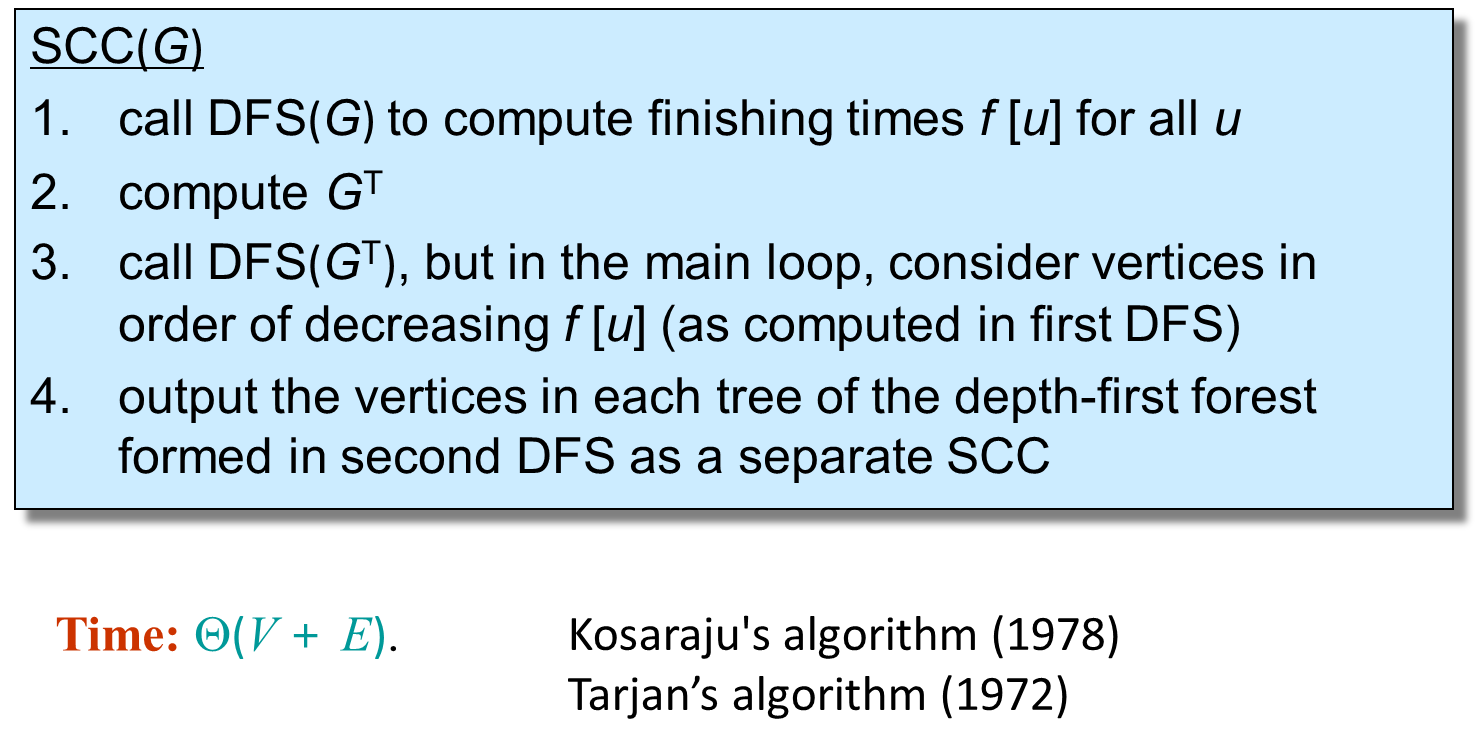


* + *Tree edge*: encounter new (white) vertex
  + *Back edge*: from descendent to ancestor
  + *Forward edge*: from ancestor to descendent
  + *Cross edge*: between a tree or subtrees
    - From a gray node to a black node

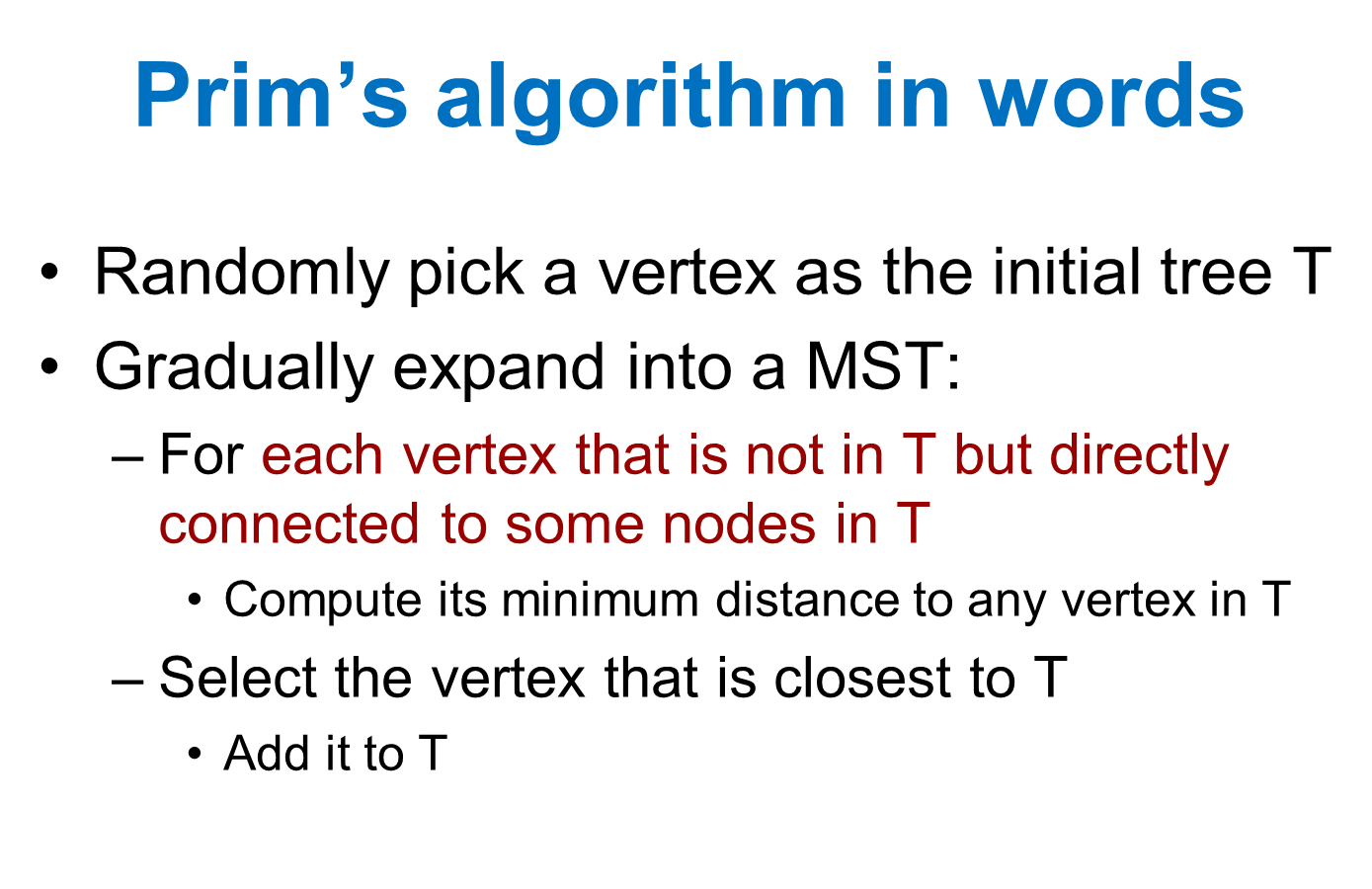
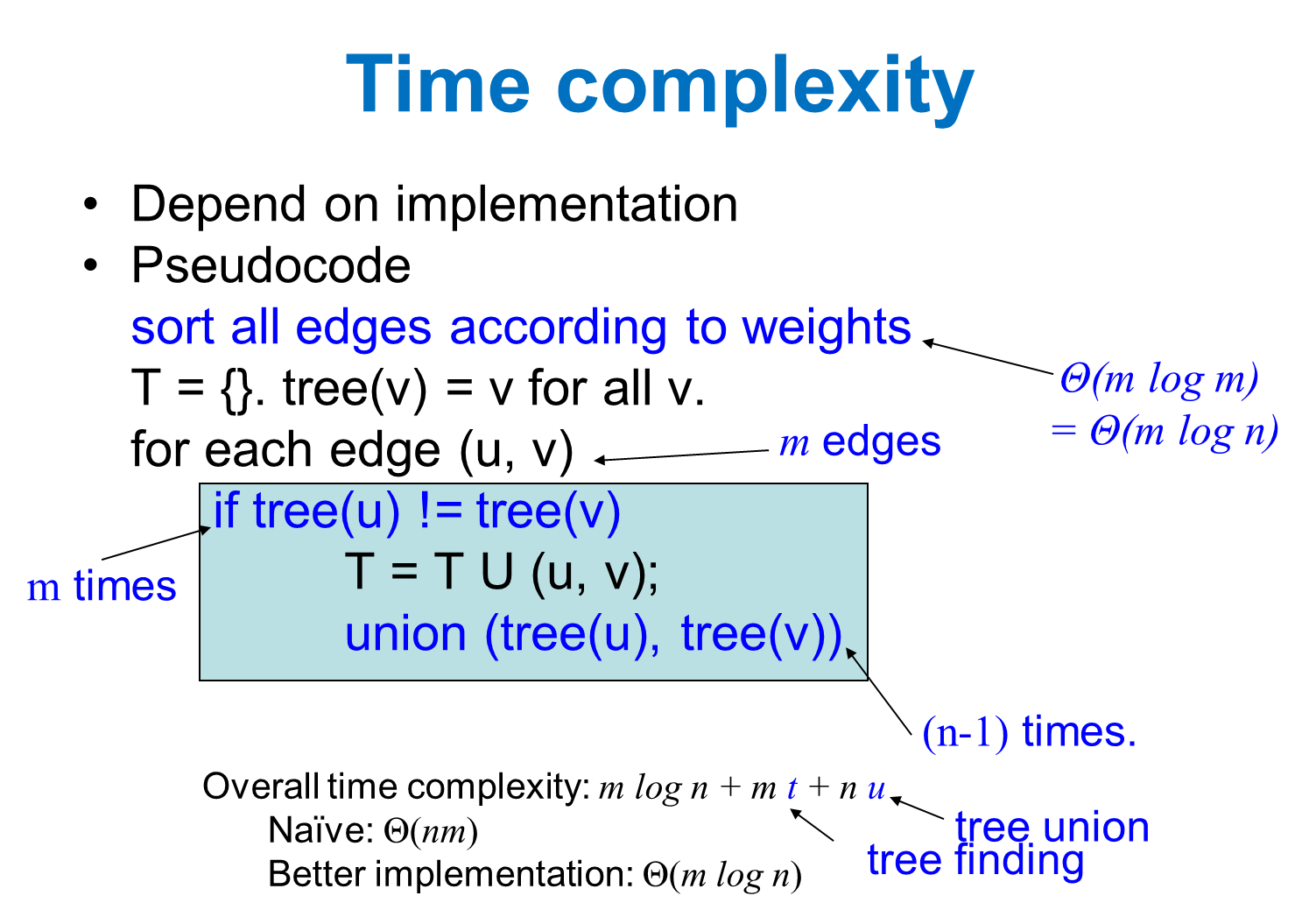
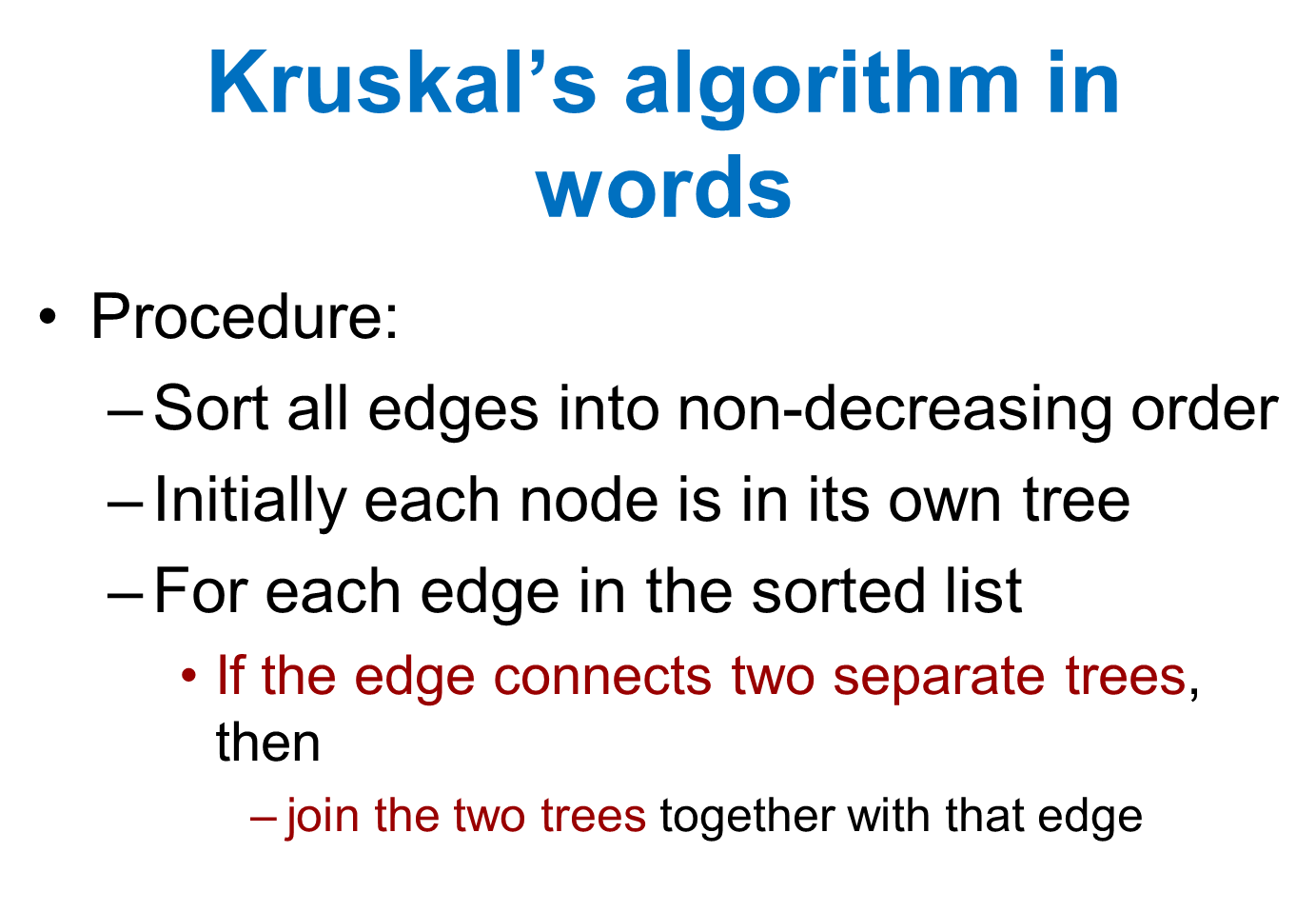
**Topological Sort:**

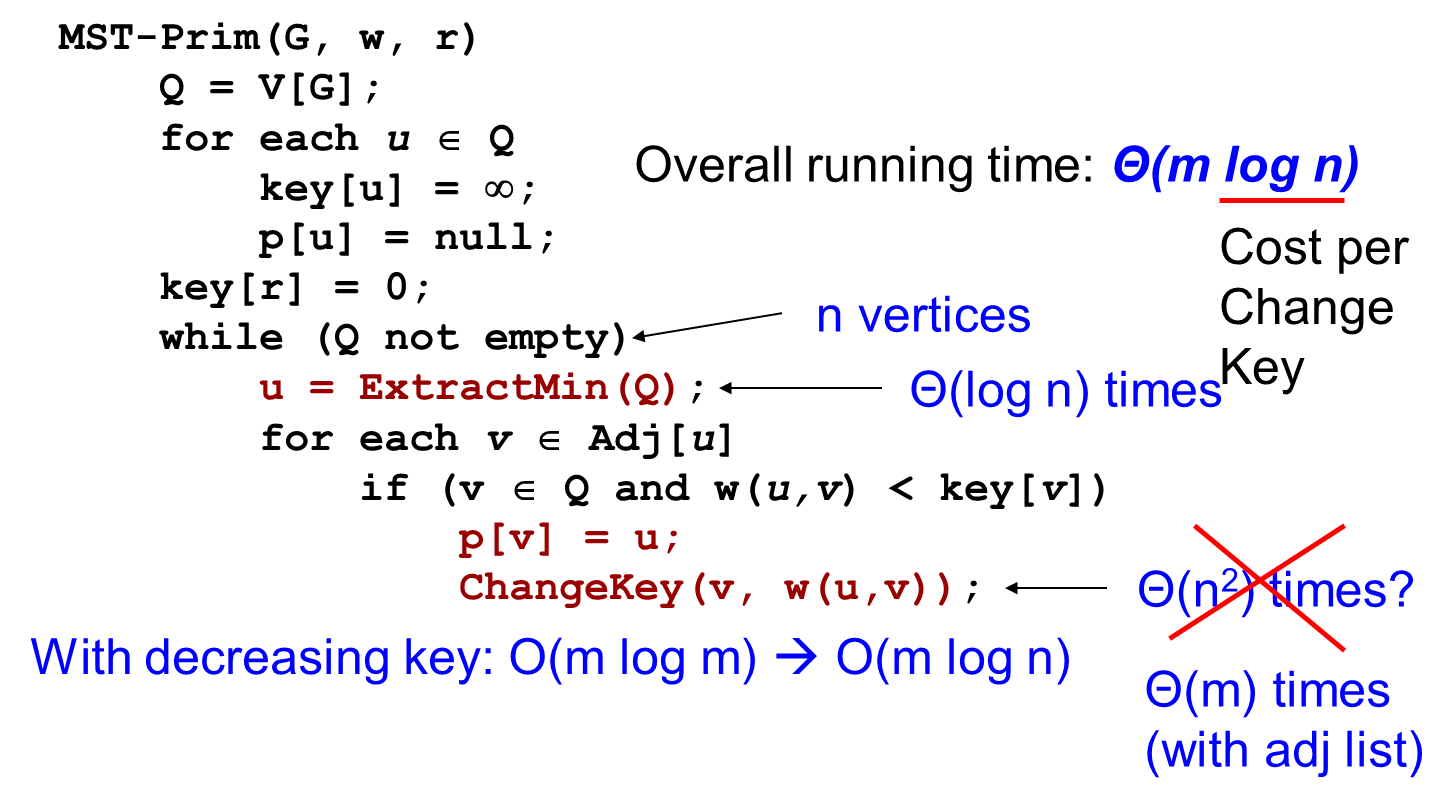


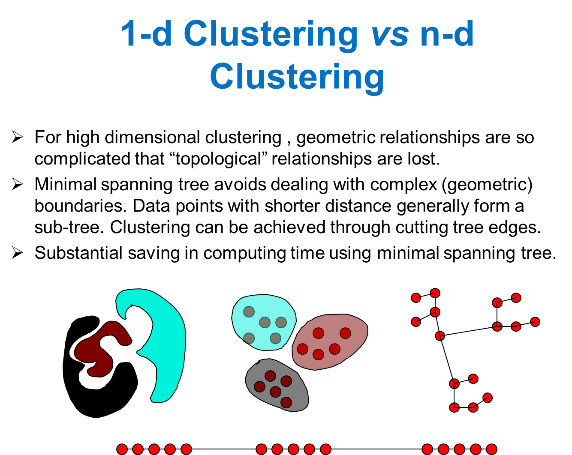
**Strongly Connected Components (SCC):**



**Lecture 11: MST**

**A picture containing diagram

Description automatically generated**

**HW 6**

1. Suppose that a graph G has a minimum spanning tree already computed. Design an efficient algorithm (in pseudo code) to quickly update the minimum spanning tree if we add a new vertex and incident edges (new edges are all connected to the new vertex) to G?

**For each vertex:**

**If smallest incident edge not within T:**

**Replace edge with existing edge in T**

**Break**

**End if**

**End for**

3. Suppose that all edge weights in a graph are integers in the range from 1 to |V|. How fast can you make Kruskal's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W?

**If the range is from 1 to |V|, Kruskal’s algorithm will run in O(VlogV + E) time. If the range is from 1 to some constant W, Kruskal’s algorithm will run in O(VlogV + E + W) time.**

