Fundamental Algorithmic Techniques VII

November 7, 2025

Outline

Search on Graphs

Advanced notions

Algorithms

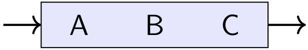
Graph Traversals: BFS vs DFS

Breadth First Search

Queue (FIFO)

Level-order

Shortest path (unweighted)



queue (FIFO)

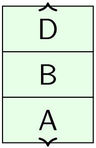
Depth First Search

Stack (LIFO)

Deep-first, backtrack

Discovery/finish times

SCCs = Strongly Connected Components



stack (LIFO)

Both solve reachability — BFS: wide, DFS: deep

Analysis of Search

Search on graph: $\mathcal{G} = (V, E)$,

- Each edge uv in the component traversed twice $\implies 2E + 1$
- Search in sparse! Adjacency matrix $\mathcal{O}(V)$, $\mathcal{O}(V^2)$ if not sparse!

Time complexity: $\mathcal{O}(V + ET)$

Cliques

A **clique** is a subset of vertices in an undirected graph such that:

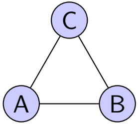
- Every two distinct vertices are **adjacent**
- The induced subgraph is **complete**

Every two distinct vertices are **adjacent**

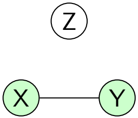
The induced subgraph is **complete**

Examples:

3-clique (size 3)



2-clique (size 2)



Note:

- Any single edge is a clique of size 2. The largest clique in a graph is the *maximum clique* (NP-hard to compute).
- Bron-Kerbosch algorithm for finding maximum clique

Minimum Spanning Tree (MST)

A **spanning tree** of a connected, undirected graph $G = (V, E)$ is:

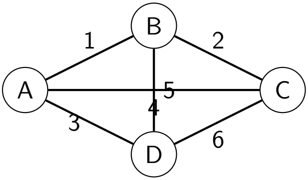
A subgraph that is a **tree**

Includes **all vertices** ($|V|$ nodes)

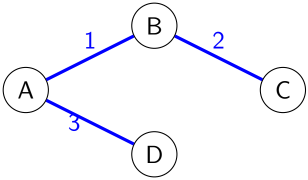
Has exactly $|V| - 1$ edges (no cycles)

A **minimum spanning tree** (MST) is a spanning tree with the **smallest possible total edge weight**.

Example:



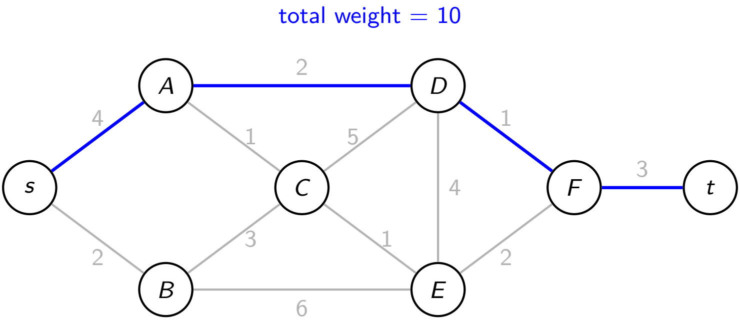
Original graph



MST (total weight = 6)

Used in network design, clustering, and approximation algorithms.

Shortest Path

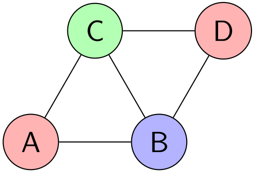


The **shortest path** from s to t minimizes the sum of edge weights.

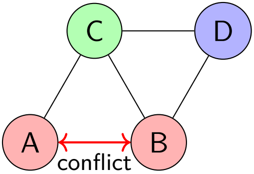
Graph Coloring

A **proper coloring** assigns colors to vertices so that **no two adjacent vertices share the same color**.

Valid 3-coloring



Invalid coloring

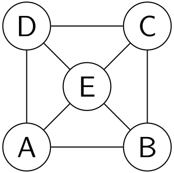


The smallest number of colors needed is the **chromatic number**.

Planar Graphs

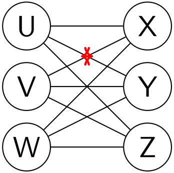
A graph is **planar** if it can be drawn in the plane **without edge crossings** (except at vertices).

Planar graph



Drawing without crossings → planar.

Non-planar graph



$K_{3,3}$ (complete bipartite) is non-planar.

Strongly Connected Components (SCCs)

Definition: In a directed graph $G = (V, E)$, a strongly connected component is a maximal subset $C \subseteq V$ such that for every pair $u, v \in C$, there is a directed path from u to v **and** from v to u.

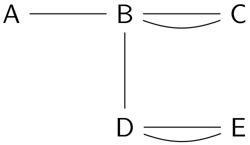
Key ideas:

Every vertex belongs to exactly one SCC.

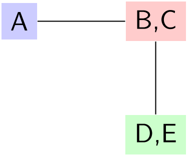
SCCs partition the vertex set.

The *condensation* of G (contracting each SCC to a node) is a DAG.

Original graph

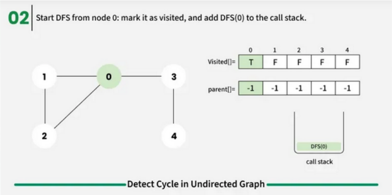


SCCs (condensation)

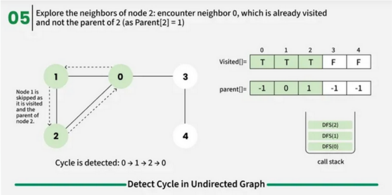


Algorithms: Kosaraju's, Tarjan's, or Gabow's (all linear time: $O(|V| + |E|)$).

Cycle Detection: DFS vs BFS — Complexity



Depth-First Search step 0



Depth-First Search step 3

Both detect the cycle when exploring the back edge (e.g., $D \to A$):
since the target node is already visited and not the immediate parent (in undirected) or is on the recursion stack (in directed).

Complexity:

Time: $O(V + E)$ for both
Every vertex and edge is processed at most once.

Space: $O(V)$ for both

- **DFS**: Call stack depth V (worst-case path).
- **BFS**: Queue may hold up to $O(V)$ nodes (e.g., wide level).

Bron–Kerbosch Algorithm: Maximal Clique Enumeration

Undirected graph $G = (V, E)$, $N(v)$ = neighbors of v in G,

Key idea: Backtracking with pruning using three disjoint sets:
R: current clique being built, P: prospective vertices (can extend R), X: excluded vertices (already processed).

Initial call: BronKerbosch1($\emptyset$, V, $\emptyset$)

Pseudocode:

```
algorithm BronKerbosch1(R, P, X) is
    if P and X are both empty then
        report R as a maximal clique
    for each vertex v in P do
        BronKerbosch1(R ∪ {v}, P ∩ N(v), X ∩ N(v))
        P := P \ {v}
        X := X ∪ {v}
```