# Graph

- Consists of nodes/vertices and edges.
- Edges may be directed or undirected.
- Edges may be weighted.
- Representation: adjacency matrix or adjacency list

# Graph

- Adjacency matrix
  - Adjacency determination: $O(1)$
  - Finding neighbours: $O(n)$
  - $O(n^2)$ space
- Adjacency list
  - Adjacency determination: $O(n)$
  - Finding neighbours: $O(\text{deg})$
  - $O(n+m)$ space

# Implicit Graphs

- Sometimes the graph is too big to build compared to the portion that is explored.
- If there is a way to generate neighbours given any vertex, we do not have to explicitly build the graph.
- e.g. configurations of some puzzle or game

# Depth-first Search

- Can be used to process connected components.
- Need to watch out for stack overflow.
- Relatively simple to write.
- $O(n^2)$ or $O(n+m)$ depending on representation.

# Breadth-first Search

- Similar to DFS but uses a queue.
- Guaranteed to examine closest vertices first (by number of edges).
- Can be used to find shortest paths when each edge has same weight.
- $O(n^2)$ or $O(n+m)$ depending on representation.

## Bipartite Check

- Colours must alternate between black and white.
- Use DFS to colour nodes, and make sure that no coloured neighbour has the same colour.

# Topological Sort

- Given a directed acyclic graph, returns an order of the vertices such that if there is a path from $v_1$ to $v_2$ then $v_1$ occurs earlier than $v_2$ in the order.
- e.g. satisfying prerequsite requirements
- `top_sort.cc` in library.

# Biconnected Component

- A graph is **biconnected** if removing any single vertex from the graph (and all adjacent edges) leaves a graph with a single component.
- An **articulation point** is a vertex such that when it is removed, the remaining graph is disconnected.
- A biconnected component is a maximal subgraph that is biconnected.
- A **bridge** is a biconnected component with a single edge.
- Application: critical points in networks, converting two-way streets to one-way streets.
- Biconnected components can be obtained from DFS.
- `bicomp.cc` in library.

# Strongly Connected Component

- A directed graph is **strongly connected** if between each pair of vertices $u$ and $v$, there is a path from $u$ to $v$ and vice versa.
- **Strongly connected components** is a maximal subgraph that is strongly connected.
- The SCCs in a graph form a directed acyclic graph.
- Obtained from DFS. See `scc.cc` in library.

## Minimum Spanning Tree

- Given a weighted undirected graph, choose a subset of edges so that the graph is connected and the total weight is minimum.
- Kruskal's algorithm is the easiest (based on union-find).
- $O(m \log m)$

# Applications of MST and MST Algorithms

- Cheapest way to connect a set of computers, cities, etc.
- Minimum spanning forests.
- Second-best MST.
- Minimax path problems.