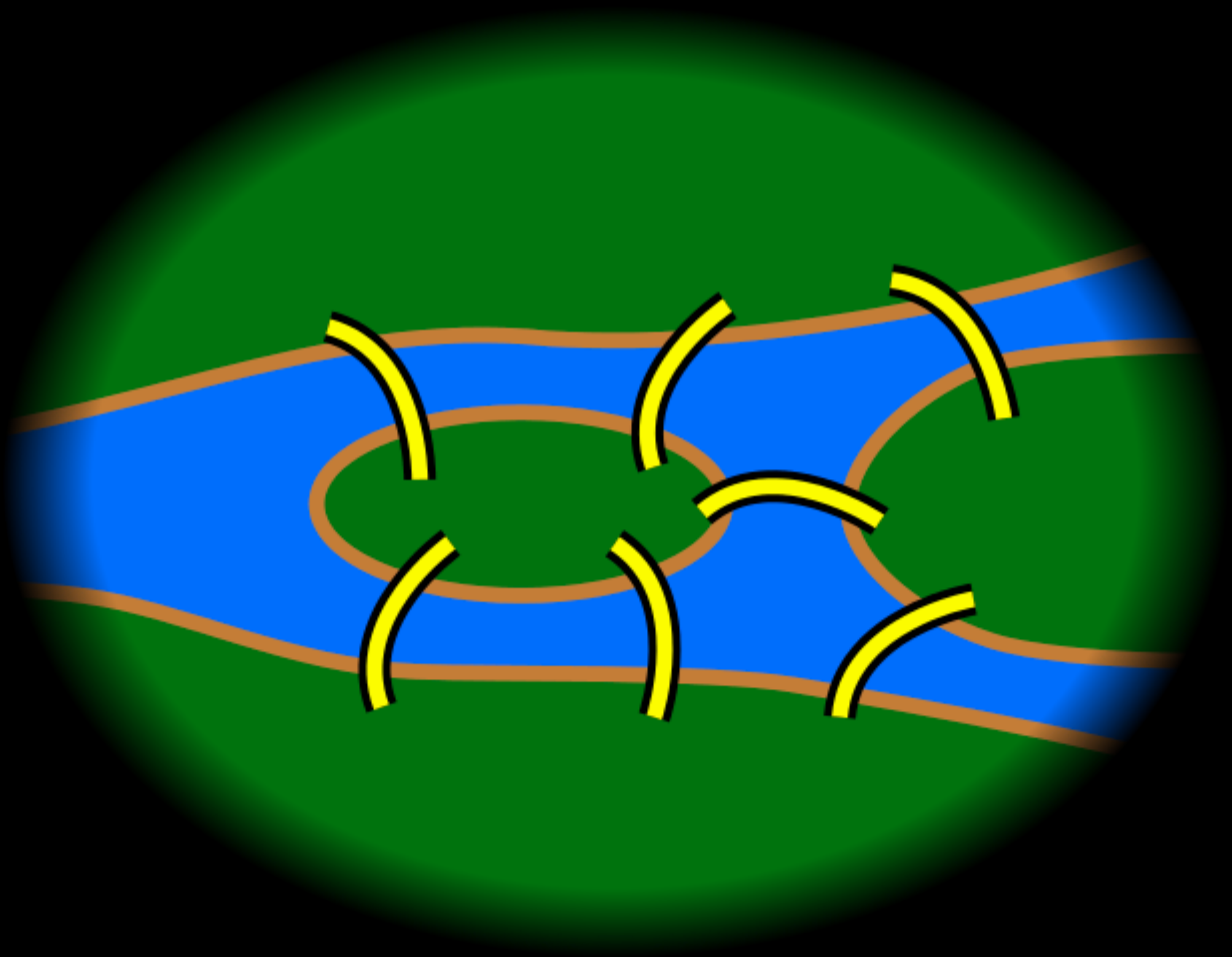
Intro to Graph Theory

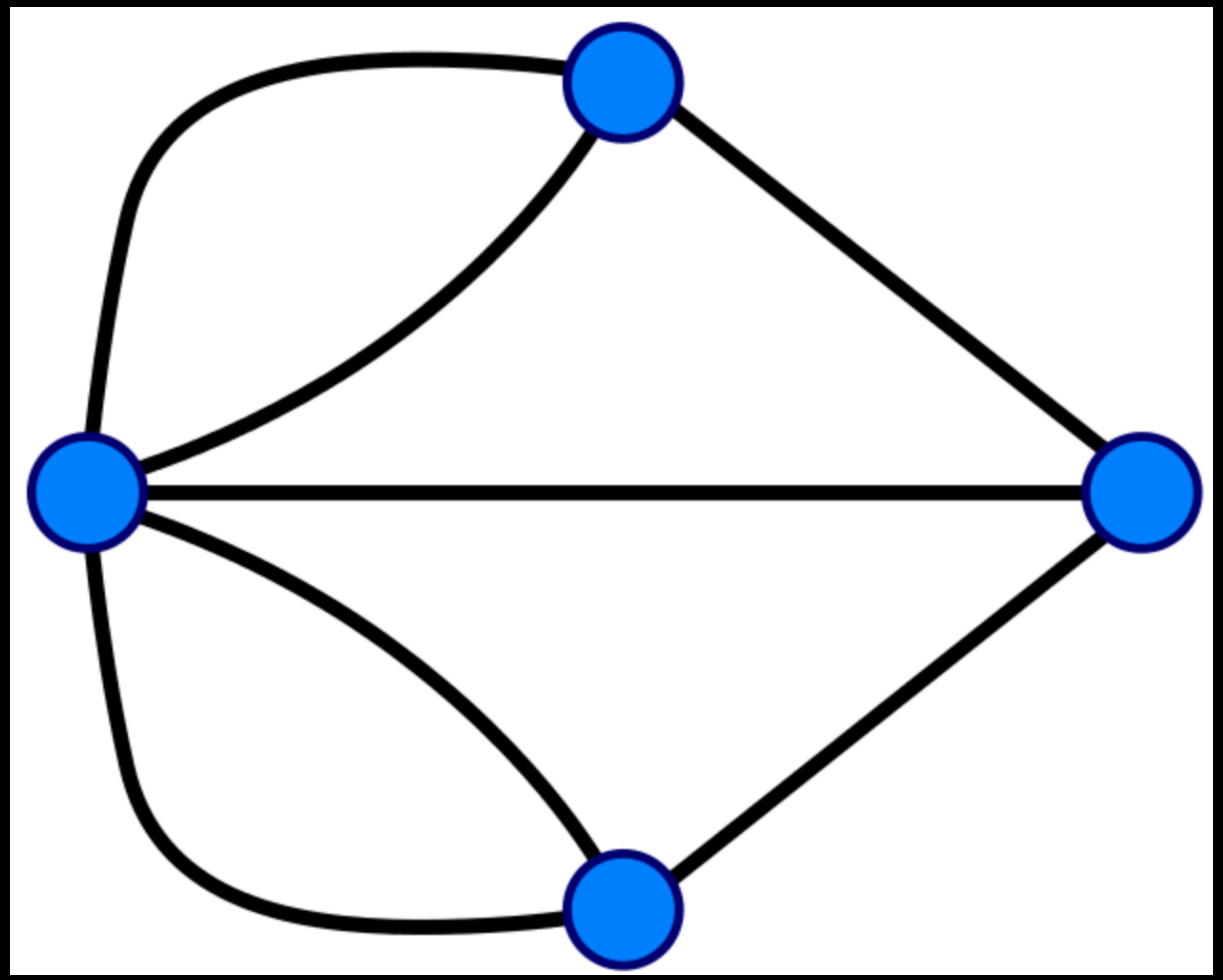
UoA CP Lecture 5&6

What's Graph

- Used to represent objects and their relationships
- Directed/Undirected
- Weighted/Unweighted
- Tree/Directed Acyclic Graph(DAG)/Bipartite

Example: Seven Bridges of Königsberg





- Each node must have an even degree so that a circuit uses each edge exactly once is possible

Graph Representation

- Adjacency List: O(m)
- Adjacency Matrix: O(n^2)

Directed Acyclic Graph(DAG)

Longest Path

- Given a weighted DAG, how to find the longest path?
- DP[i] = max(DP[j] + w) given there is an edge from j to i of weight w.

- Actually all the DP problem can be reduced to this model.

Directed Acyclic Graph(DAG)

Topological Sort

- While there are nodes left, pick a node v of zero in-degree
- Update the DP values of all the successors of v
- Think: Why do we not need not do this for DP problems

Shortest Path I (Unweighted Graph)

- Breadth First Search
- Divide nodes by the distance from the origin to it
- Key observation: dis[i] = min(dis[j] + 1) where j loops through all the predecessors of i.

Shortest Path II (Weighted Graph, Positive Weight)

Dijkstra's Algorithm

- Idea is similar to BFS
- dis[i] = min(dis[j] + w(j,i)) => Circular Dependency
- One thing is sure: dis[start] = 0
- In every round, pick the closest vertex hasn't been processed, use it to update its neighbours' distances.

Shortest Path III (Weighted Graph, Any Weight)

Bellman-Ford Algorithm

- The shortest path (if exists) contains at most |V|-1 edges (Why?)
- DP[i][j] = the shortest path from origin to i using j edges
- DP[i][j+1] = min(DP[k][j] + e(k,i))
- DP[start][0] = 0, others are initialised as ∞
- If we can still update after |V|-1 rounds, there must be a negative cycle => shortest path does not exist.

Shortest Path IIII (APSP, any weight)

Floyd's Algorithm

- Probably the shortest algorithm you ever learn.
- All Pair Shortest Path
- DP[k][i][j] = the shortest path from i to j using vertices only from the {1,2, …,k } as intermediate points along the way.
- DP[k+1][i][j] = min(DP[k][i][k+1]+DP[k][k+1][j], DP[k][i][j])
- DP[0][i][j] = e(i,j)

Minimum Spanning Tree

How to connect an undirected graph with the lowest cost?

- Observation 1: For a node i, using the edge (i,j,w) from i with the lowest cost will never be non-optimal
  - Assume the MST doesn’t use (i,j,w). If we add it to the tree, a cycle will appear. Now we can remove the other edge from i without making the sum of weights larger.
- Observation 2: Adding a self-loop is always unnecessary
- Algorithm:
  - Sort edges by weight and loop through the edges
  - If the endpoints are already connected, skip. Otherwise, add the edge to the tree.