CSCE-42603
Advanced Data Structures

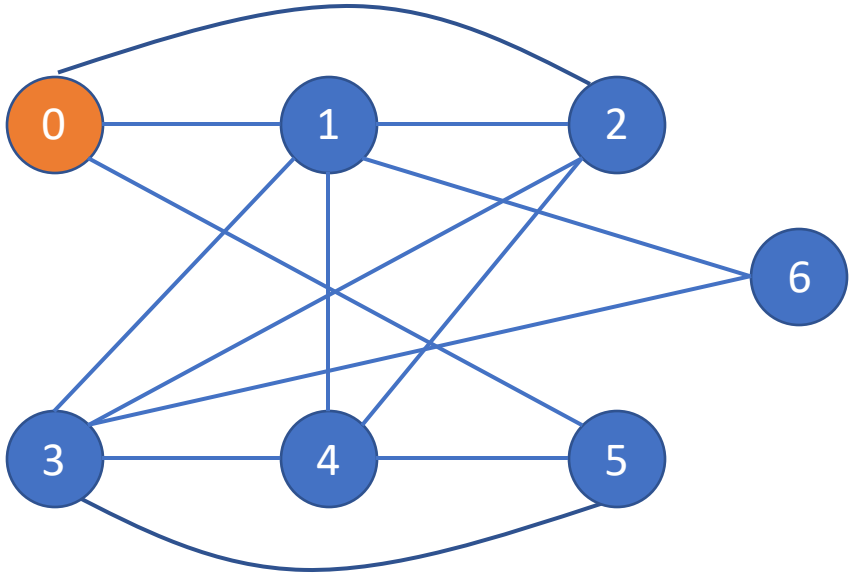
Homework Review Assignment 5

Fall 2025

Prof. Khoa Luu

Dr. Thanh-Dat Truong

BFS

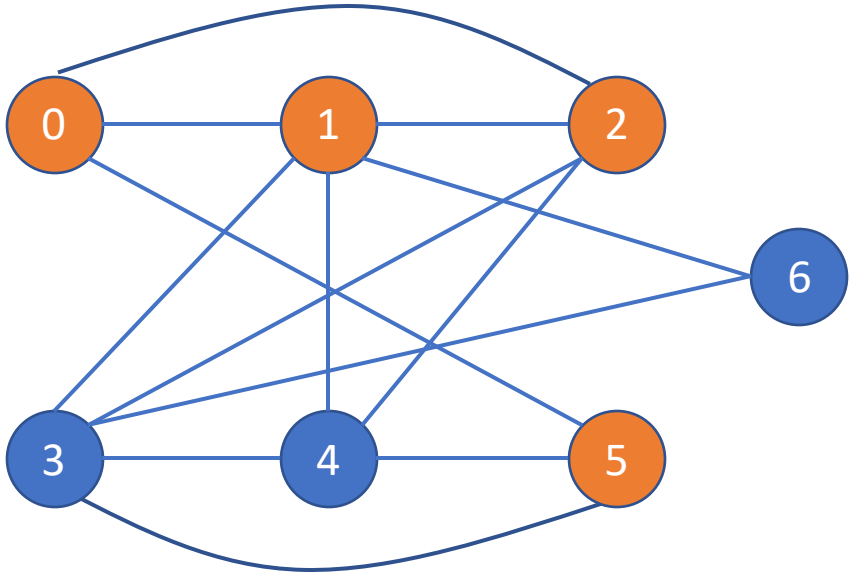


BFS Order: 0

```
void bfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<boo
&visited) {
  std::queue<int> queue;
  queue.push(startVertex);
  visited[startVertex] = true;

  while (!queue.empty()) {
    int u = queue.front();
    std::cout << u << " ";
    queue.pop();
    for (int i = 0; i < adj[u].size(); ++i) {
      int v = adj[u][i]; // v is a neighbor of u
      if (!visited[v]) {
        queue.push(v);
        visited[v] = true;
      }
    }
  }
}
```

BFS

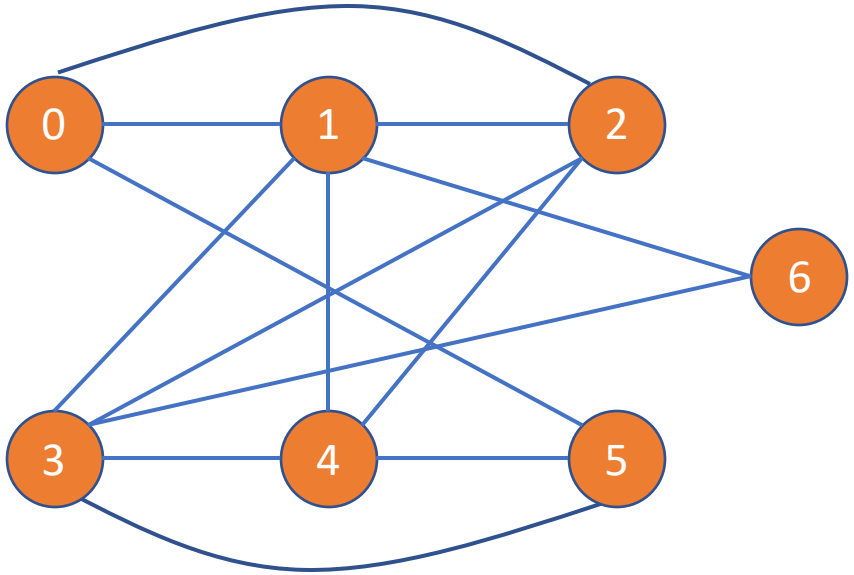


BFS Order: 0 1 2 5

```
void bfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<boo
&visited) {
  std::queue<int> queue;
  queue.push(startVertex);
  visited[startVertex] = true;

  while (!queue.empty()) {
    int u = queue.front();
    std::cout << u << " ";
    queue.pop();
    for (int i = 0; i < adj[u].size(); ++i) {
      int v = adj[u][i]; // v is a neighbor of u
      if (!visited[v]) {
        queue.push(v);
        visited[v] = true;
      }
    }
  }
}
```

BFS

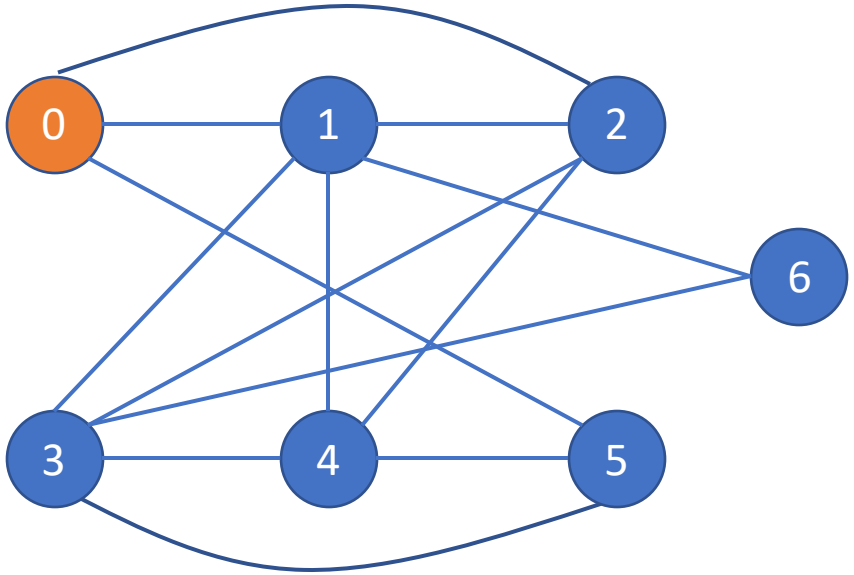


BFS Order: 0 1 2 5 6 4 3

```
void bfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<boo
&visited) {
  std::queue<int> queue;
  queue.push(startVertex);
  visited[startVertex] = true;

  while (!queue.empty()) {
    int u = queue.front();
    std::cout << u << " ";
    queue.pop();
    for (int i = 0; i < adj[u].size(); ++i) {
      int v = adj[u][i]; // v is a neighbor of u
      if (!visited[v]) {
        queue.push(v);
        visited[v] = true;
      }
    }
  }
}
```

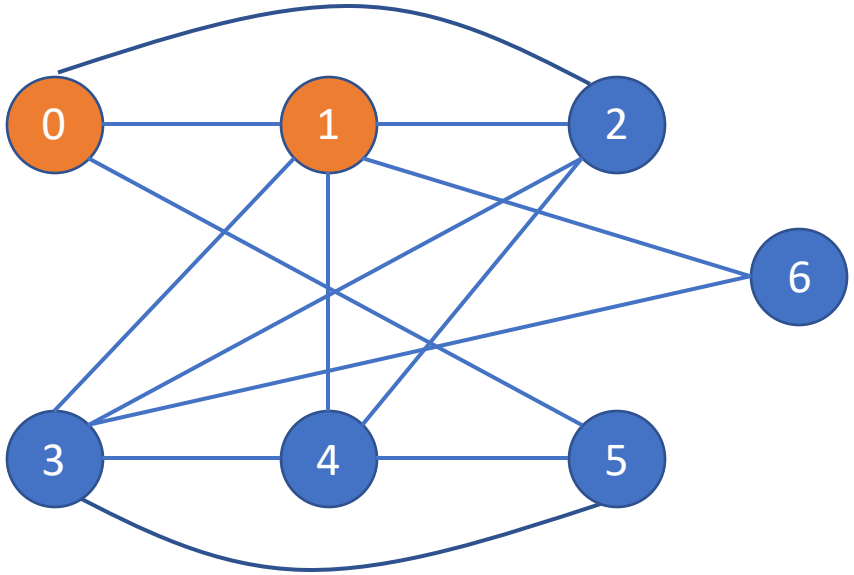
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0

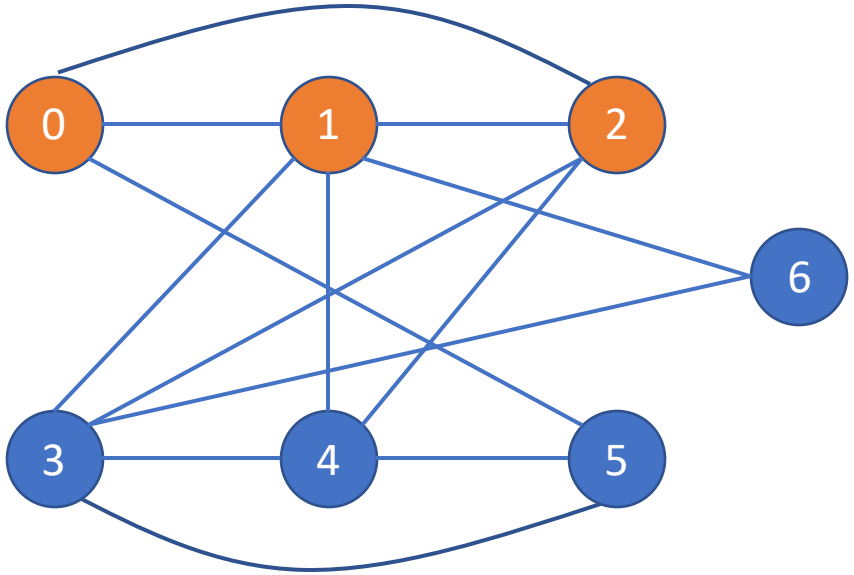
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1

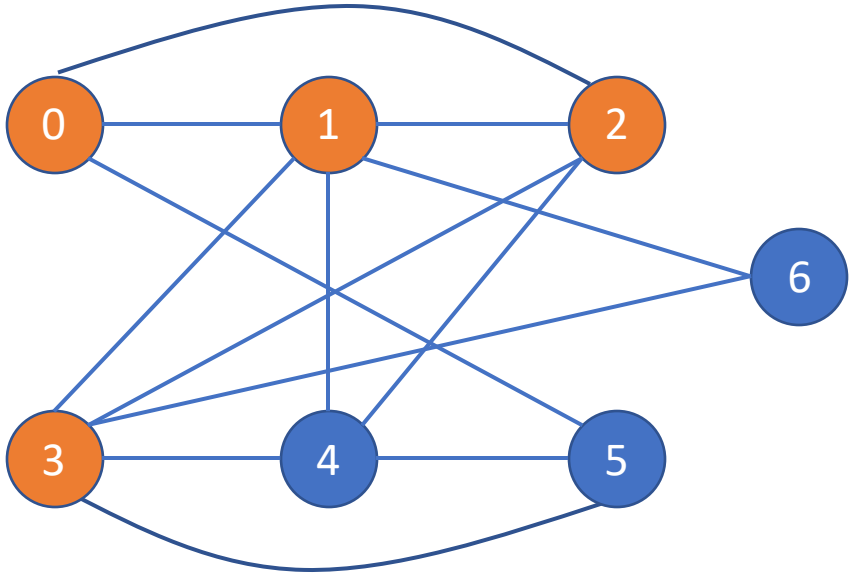
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1 2

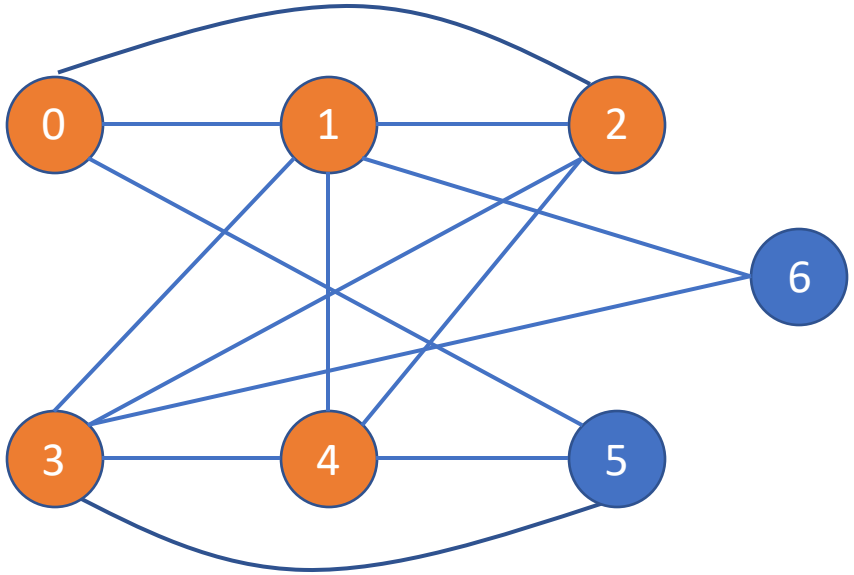
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1 2 3

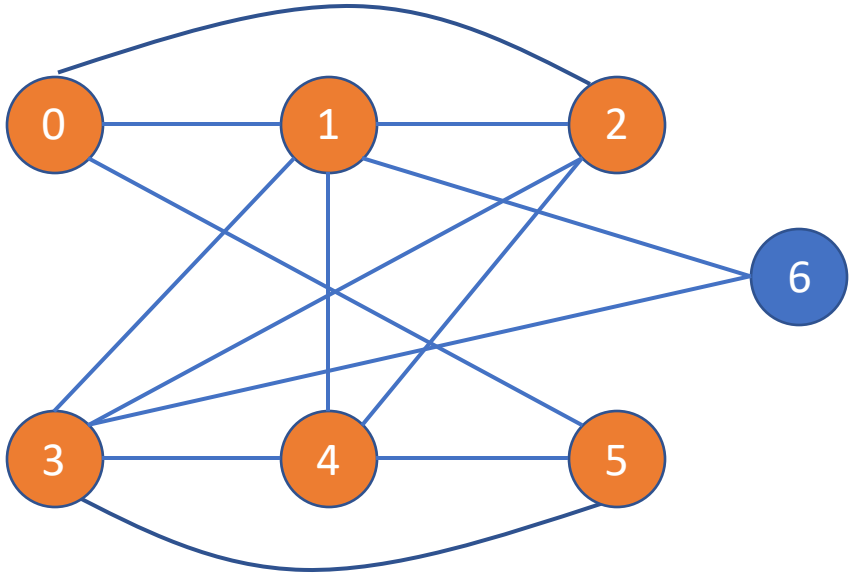
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1 2 3 4

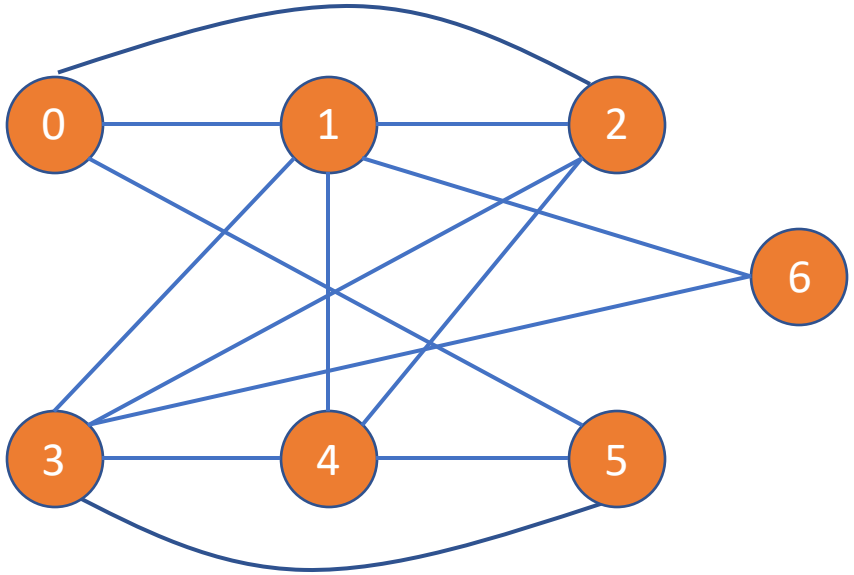
DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1 2 3 4 5

DFS




```
void dfs(int startVertex, int n, std::vector<std::vector<int> > &adj, std::vector<bool
&visited) {
  int u = startVertex;
  std::cout << u << " ";
  visited[u] = true;
  for (int i = 0; i < adj[u].size(); ++i) {
    int v = adj[u][i]; // v is a neighbor of u
    if (!visited[v])
      dfs(v, n, adj, visited);
    }
}
```

DFS Order: 0 1 2 3 4 5 6

Graph Neural Network

- Graph Convolutional Network

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v) \cup \{v\}} \frac{h_u^{k-1}}{\sqrt{\deg(u) + \deg(v)}}\right)$$

- Move To Google Collab For A Demo
  - https://colab.research.google.com/drive/1-SZcoVDCKuYTbSV_9oVbV-q0LOuOSbDk?usp=sharing