Review of Graph

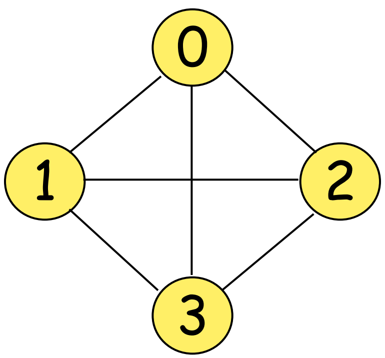
- Sets and disjoint sets,
- union,
- sorting and searching algorithms and

their analysis in terms of space and
time complexity.

Introduction to Graph

- A graph, G, consists of two sets, V and E.
  - V is a finite, nonempty set of vertices.
  - E is set of pairs of vertices called edges.
- The vertices of a graph G can be represented as V(G).
- Likewise, the edges of a graph, G, can be represented as E(G).
- Graphs can be either undirected graphs or directed graphs.

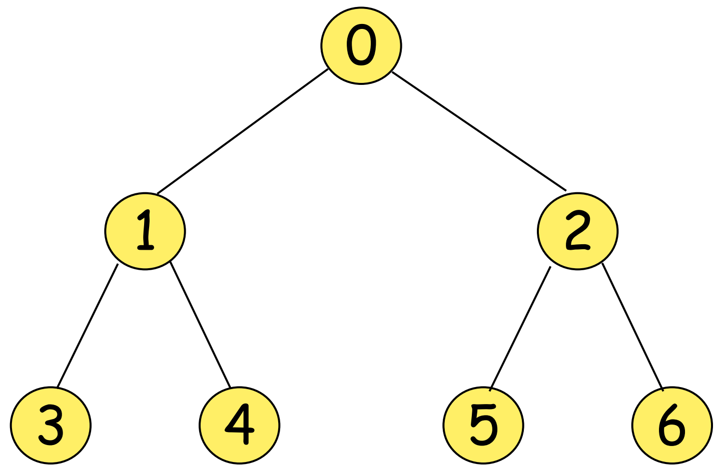
Three Sample Graphs



$V(G_1) = \{0, 1, 2, 3\}$

$E(G_1) = \{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$

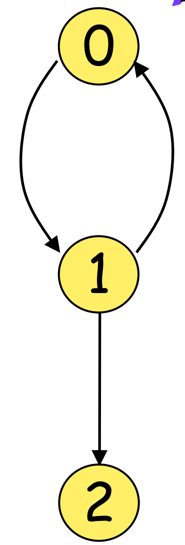
(a) G_1



$V(G_2) = \{0, 1, 2, 3, 4, 5, 6\}$

$E(G_2) = \{(0, 1), (0, 2), (1, 3), (1, 4), (2, 5), (2, 6)\}$

(b) G_2



$V(G_3) = \{0, 1, 2\}$

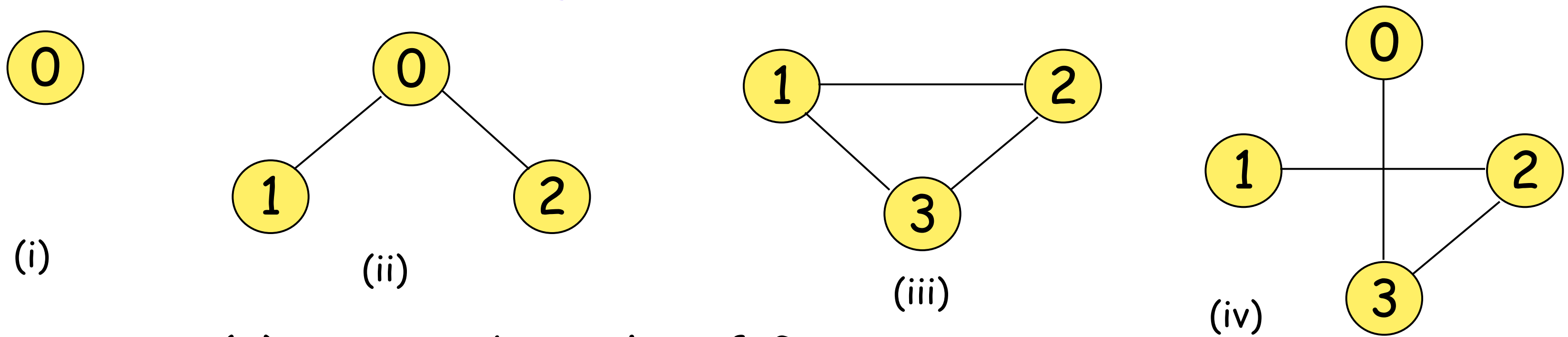
$E(G_3) = \{<0, 1>, <1, 0>, <1, 2>\}$

(c) G_3

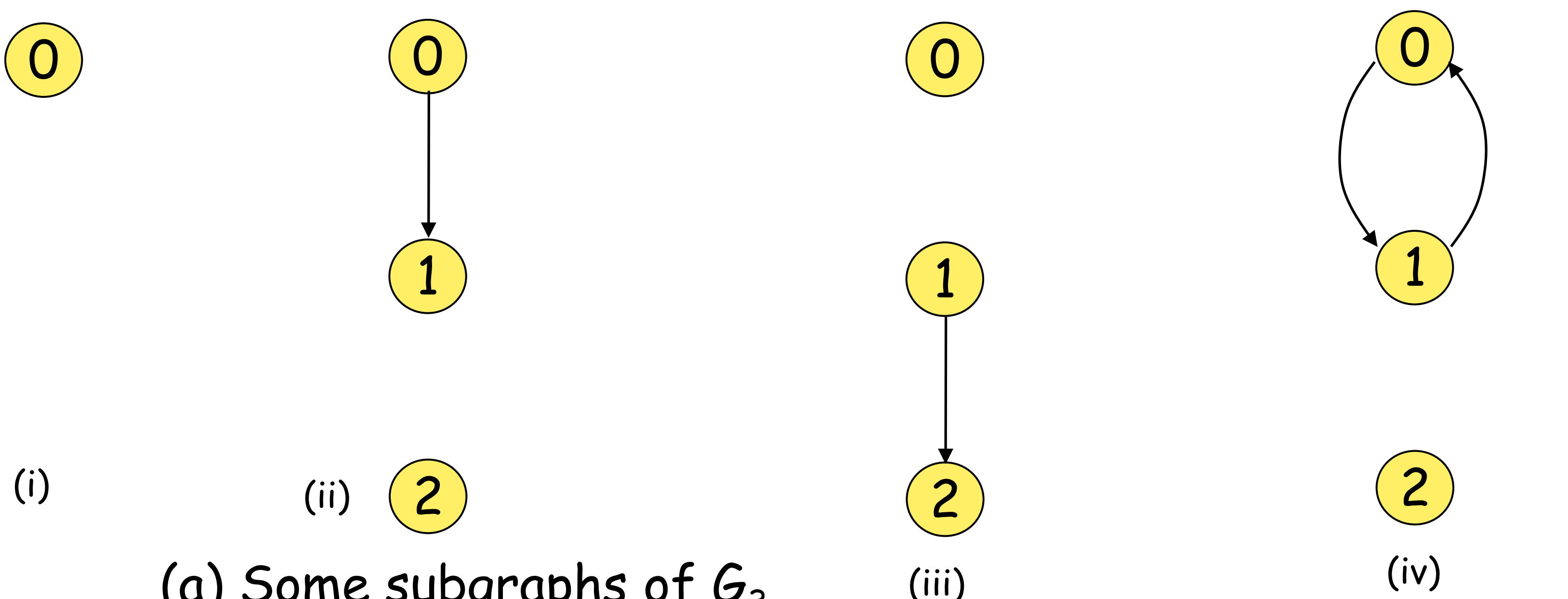
Subgraph and Path

- Subgraph: A subgraph of G is a graph G' such that V(G') $\subseteq$ V(G) and E(G') $\subseteq$ E(G).
- Path: A path from vertex u to vertex v in graph G is a sequence of vertices u, i_1, i_2, …, i_k, v, such that (u, i_1), (i_1, i_2), …, (i_k, v) are edges in E(G).
  - The length of a path is the number of edges on it.
  - A simple path is a path in which all vertices except possibly the first and last are distinct.
  - A path (0, 1), (1, 3), (3, 2) can be written as 0, 1, 3, 2.
- Cycle: A cycle is a simple path in which the first and last vertices are the same.

G_1 and G_3 Subgraphs



(a) Some subgraphs of G_1



(a) Some subgraphs of G_3

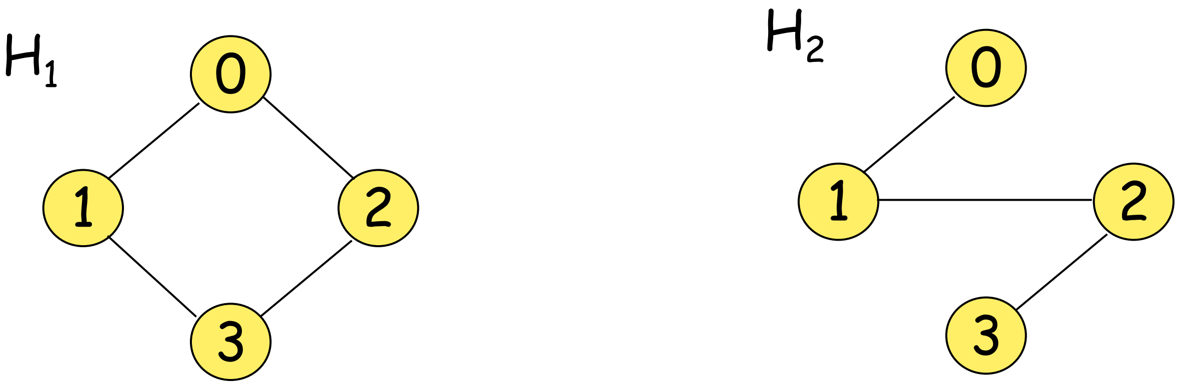
Connected Graph

- Two vertices u and v are *connected* in an graph iff there is a path from u to v (and v to u).
- A tree is a *connected acyclic* graph.

Strongly Connected Graph

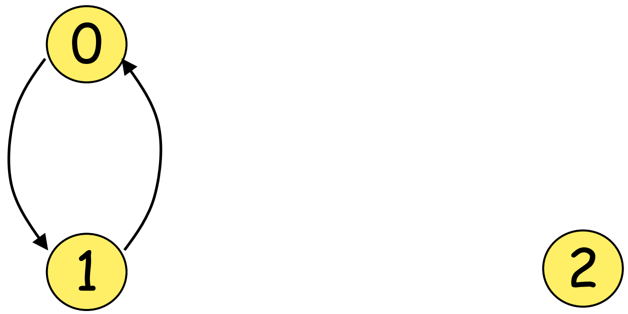
- A directed graph G is *strongly connected* iff for every pair of distinct vertices u and v in V(G), there is directed path from u to v and also from v to u.
- A *strongly connected component* is a maximal subgraph that is strongly connected.

Graphs with Two Connected Components



G_4

Strongly Connected Components of G_3



Degree of A Vertex

- *Degree of a vertex*: The *degree* of a vertex is the number of edges incident to that vertex.
- If G is a directed graph, then we define
  - *in-degree of a vertex*: is the number of edges for which vertex is the head.
  - *out-degree of a vertex*: is the number of edges for which the vertex is the tail.
- For a graph G with n vertices and e edges, if d_i is the degree of a vertex i in G, then the number of edges of G is

$$e = \left(\sum_{i=0}^{n-1} d_i\right) / 2$$

Adjacent Matrix

- Let G(V, E) be a graph with n vertices, n ≥ 1. The adjacency matrix of G is a two-dimensional nxn array, A.
  - A[i][j] = 1 iff the edge (i, j) is in E(G).
  - The adjacency matrix for a undirected graph is symmetric, it may not be the case for a directed graph.

Adjacency Matrices

Vi → Vj then A[i,j]=1

$$
\begin{array}{c} \\ 0\\1\\2\\3 \end{array}
\begin{array}{c}
\begin{array}{cccc} 0 & 1 & 2 & 3 \end{array}\\
\begin{bmatrix} 0 & 1 & 1 & 1\\ 1 & 0 & 1 & 1\\ 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 0 \end{bmatrix}
\end{array}
$$

(a) G_1

$$
\begin{array}{c} \\ 0\\1\\2 \end{array}
\begin{array}{c}
\begin{array}{ccc} 0 & 1 & 2 \end{array}\\
\begin{bmatrix} 0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

(b) G_3

j

$$
i\quad
\begin{array}{c} \\ 0\\1\\2\\3\\4\\5\\6\\7 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}\\
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

Out-degree

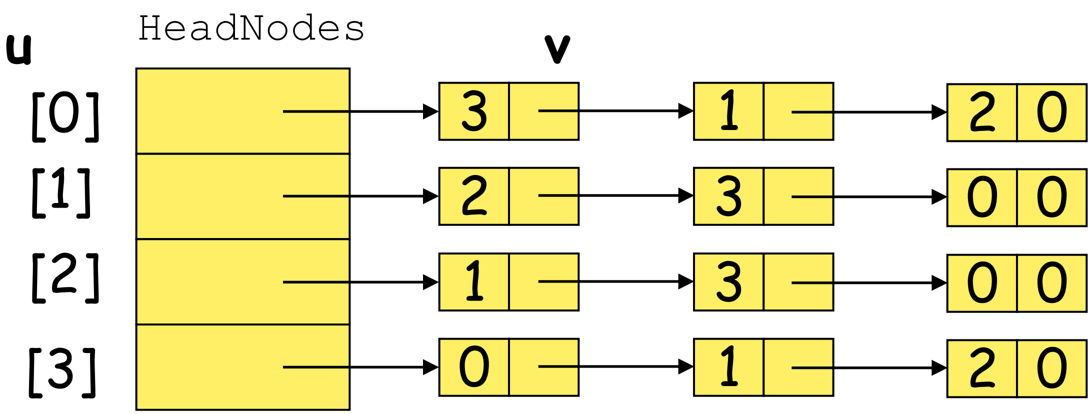
In-degree

(c) G_4

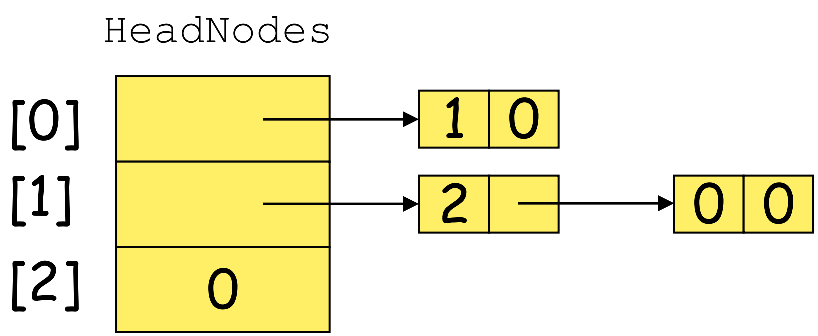
Adjacency Lists

- Instead of using a matrix to represent the adjacency of a graph, we can use n linked lists to represent the n rows of the adjacency matrix.
- Each node in the linked list contains two fields: data and link.
  - data: contain the indices of vertices adjacent to a vertex i.
  - Each list has a head node.
- For an undirected graph with n vertices and e edges, we need n head nodes and 2e list nodes.

Adjacent Lists

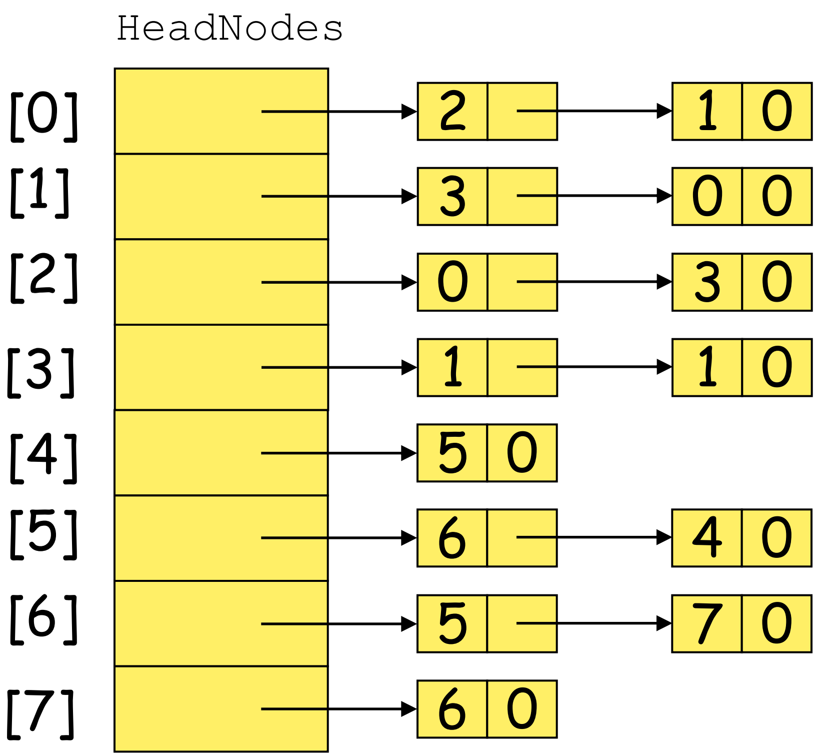


(a) G_1



(b) G_3

Adjacent Lists (Cont.)



(c) G_4

Graph Operations

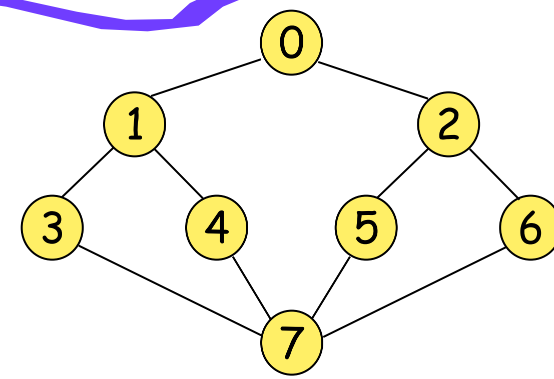
- A general operation on a graph G is to visit all vertices in G that are reachable from a vertex v.
  - Depth-first search
  - Breath-first search

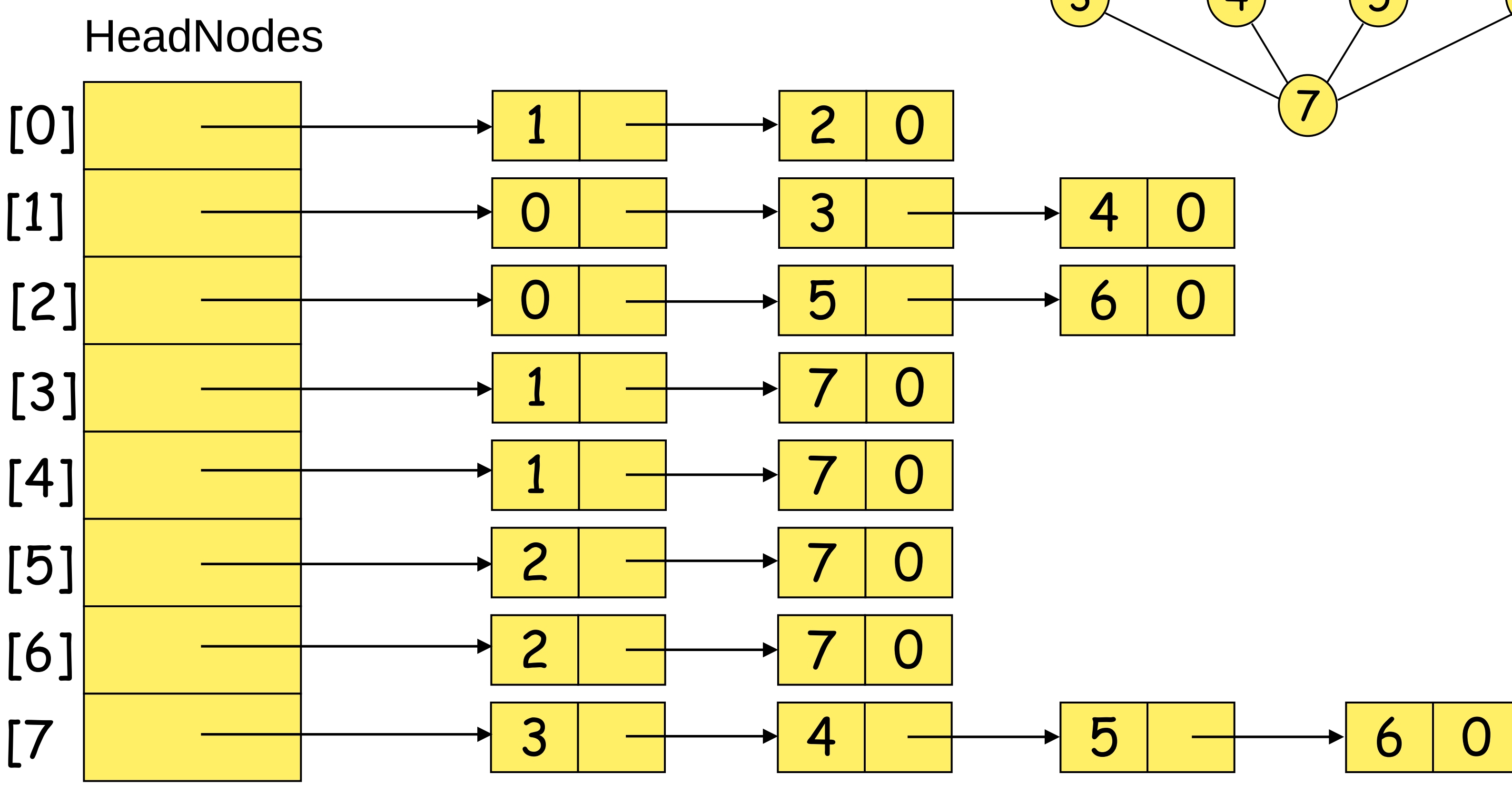
Depth-First Search

- Starting from vertex, an unvisited vertex w adjacent to v is selected and a depth-first search from w is initiated.
- When the search operation has reached a vertex u such that all its adjacent vertices have been visited, we back up to the last vertex visited that has an unvisited vertex w adjacent to it and initiate a depth-first search from w again.
- The above process repeats until no unvisited vertex can be reached from any of the visited vertices.

Graph G and Its Adjacency Lists

DFS(0)=0 1 3 7 4 5 2 6





Analysis of DFS

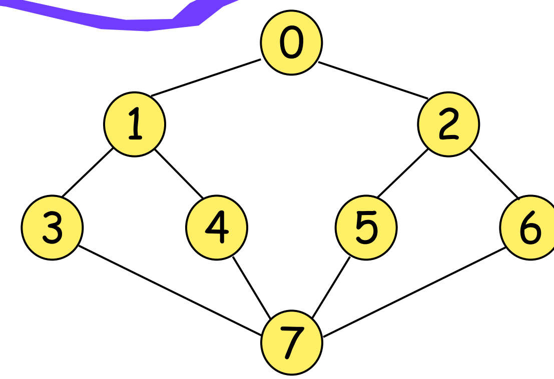
- If G is represented by its adjacency lists, the DFS time complexity is O(e).
- If G is represented by its adjacency matrix, then the time complexity to complete DFS is $O(n^2)$.

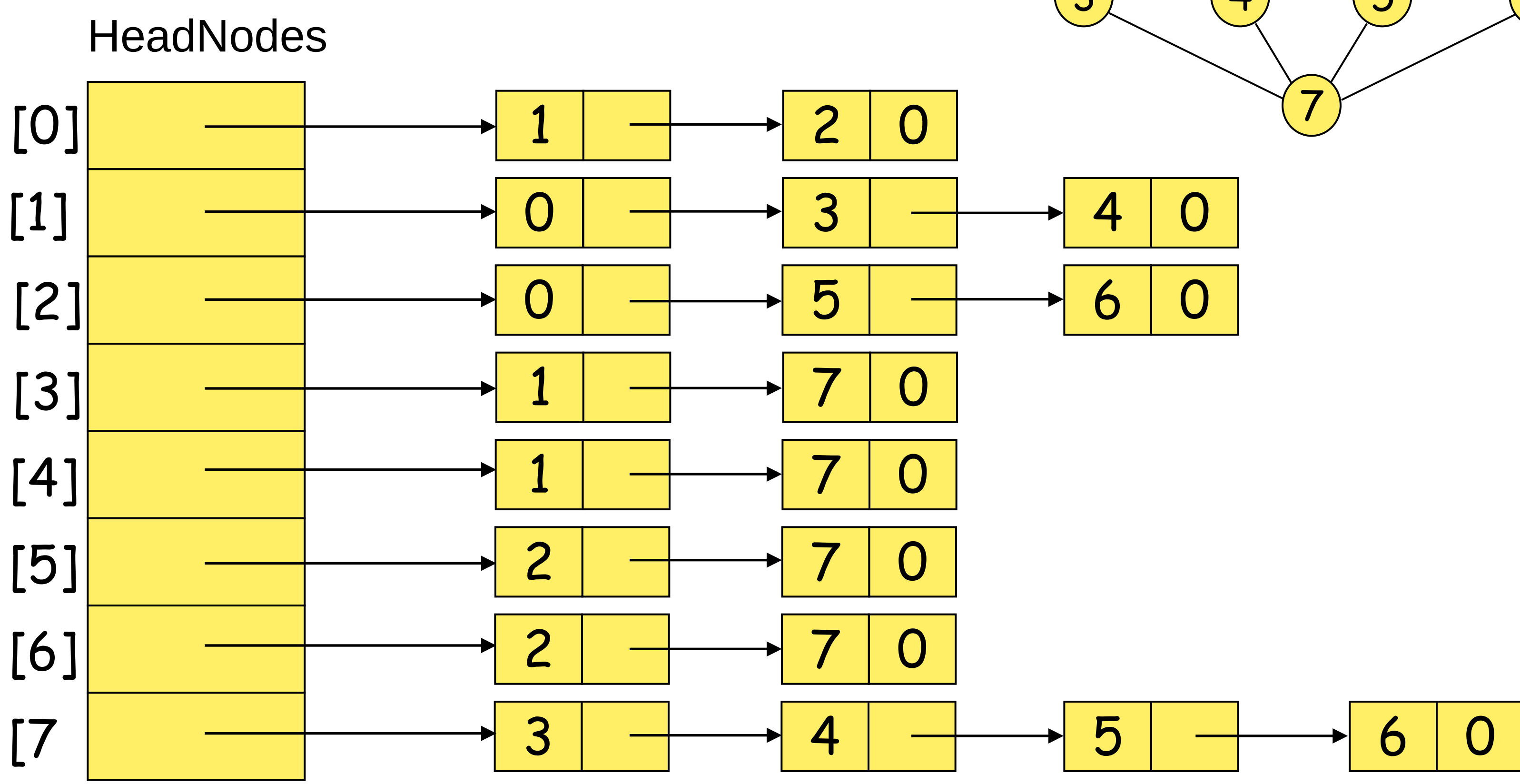
Breath-First Search

- Starting from a vertex v, visit all unvisited vertices adjacent to vertex v.
- Unvisited vertices adjacent to these newly visited vertices are then visited, and so on.
- If an adjacency matrix is used, the BFS complexity is $O(n^2)$.
- If adjacency lists are used, the time complexity of BFS is $d1+d2+\ldots+dn=O(e)$.

Graph G and Its Adjacency Lists

BFS(0)=0 1 2 3 4 5 6 7





Application

- Graph is used to construct a network which is used to find shortest path from source to destination,source to all vertices & to construct MST.
- PERT
- CPM

Scope of Research

- Operation Research

Assignment

Q.1)What is a graph?

Q.2)What is difference between path and cycle?

Q.3)What is difference between DFS & BFS traversal of a graph?