ITS 231: Data Structures and Algorithms Laboratory
Semester 1
School of Information, Computer, and Communication Technology
Sirindhorn International Institute of Technology

Lab 09: Graph

1. Objectives

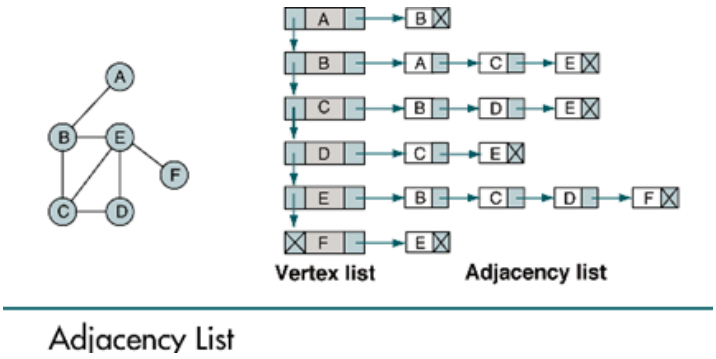
- Understand the concept of a graph data structure and its implementation
- Understand insert/delete operations of vertices and edges in a graph.
- Learn how to manipulate graphs using basic functions that you have implemented.

Reviews

A graph is a collection of **vertices**, and a collection of **edges** connecting pairs of vertices.
Two kinds of graphs: directed and undirected

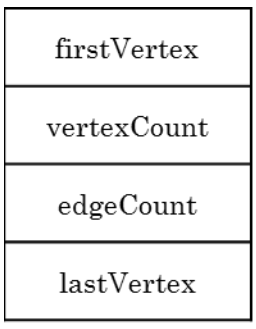
Graph Storage Structures

Adjacency list is a linked list of edges from the vertex.

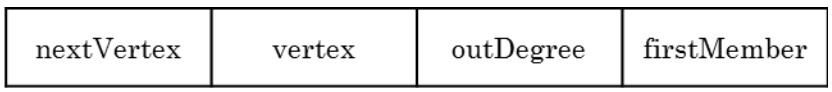


Adjacency List

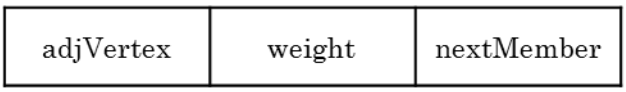
Graph Components



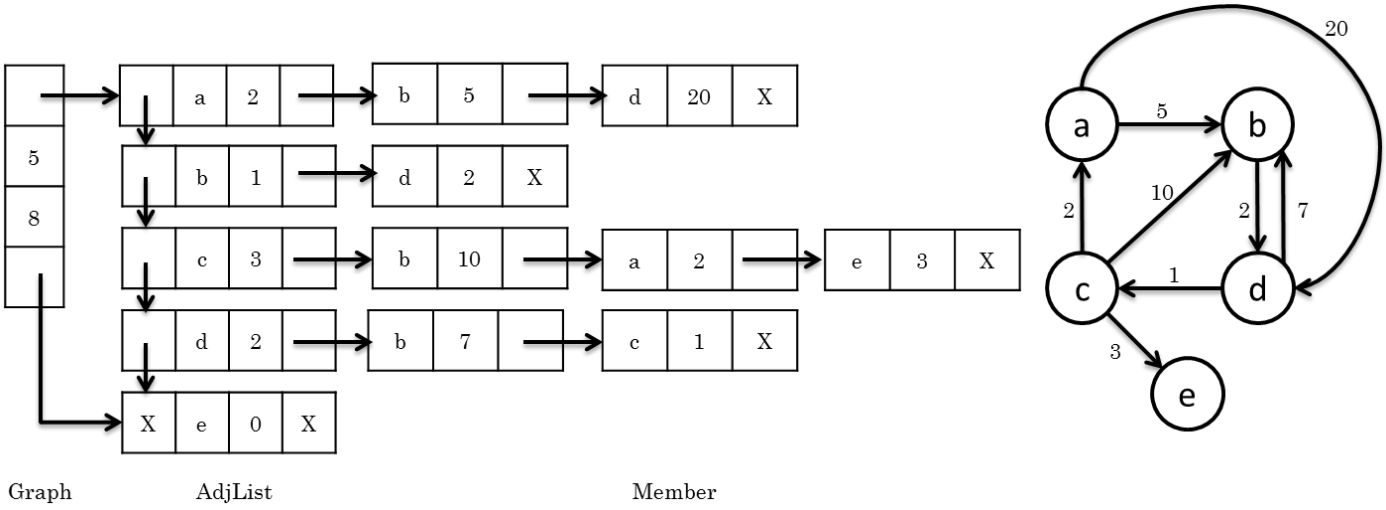
Graph



AdjList



Member



Graph, AdjList, and Member classes structure.

UML of class Graph<T>

Graph<T>
firstVertex: adjList<T> vertexCount: int edgeCount: int lastVertex: adjList<T>
Graph() insertVertex(newVertex: T): void searchVertex(data: T):AdjList insertEdge(fromData: T, toData: T, weight: int): void deleteEdge(fromData: T, toData: T): void deleteVertex(v: T): void print(): void

Graph Operations

void insertVertex(T newVertex) inserts a new vertex into a graph.
Create an adjList to keep a new vertex, and append it as the last of the vertex list. Update the vertex count and last vertex.

AdjList<T> searchVertex(T data) returns an AdjList containing data in a graph.
From the top of the vertex list, scan from the top to the bottom of the vertex list to find the adjList that contains the data. The method stops when it finds the adjList then returns such an AdjList.

void insertEdge(T fromV, T toV, int weight) inserts a toV member with the given weight as a first member of the fromV adjList.
From the first adjList, scan down in the vertex list to get the location of the adjList containing fromV vertex. Once

the location of adjList is found, add a new member with toV information as the first member.

void deleteEdge(T fromV, T toV) deletes an edge from a graph.
From the first adjList, scan down in the vertex list to get the location of the adjList of the fromV vertex, scan to the right of the adjList to find the member of which the next member is the toV. Overpass the member with toV, then update the edge count.

void deleteVertex(T v) deletes a vertex in a graph.
Iterate through every AdjList, delete every member with vertex v using deleteEdge method, and skip AdjList containing v when iterates down on the vertex list.

Exercises

Exercise 1 Implement void insertVertex(T newVertex). Test your code with the main() method.

Exercise 2 Implement AdjList<T> searchVertex. Test your code in main() method.

Exercise 3 Implement void insertEdge(T fromData, T toData, int weight) method. Test your code in main() method.

Exercise 4 Implement void deleteEdge(T fromData, T toData). Test your code in main() method.

Exercise 5 Implement void deleteVertex(T vertex). Test your code in main method.