Graphs

EDUCLASH
Just Another Way To Learn

Terminologies used in graphs

- **Graph:**
  - Is a collection of nodes called **vertices** and a collection of line segments connecting pair of vertices called **lines.**
- **Directed graph or Diagraph:**
  - Is a graph in which each line has a direction to its successor. The lines in a directed graph are known as **arcs.**
- **Undirected graph:**
  - is a graph in which there is no direction on the lines known as **edges.**

EDUCLASH Just Another Way To Learn

Terminologies used in graphs

- **Adjacent vertices:**
  - Two vertices are said to be adjacent vertices is there exists an edge that directly connects them.
- **Path:**
  - it is a sequence of vertices in which each vertex is adjacent to the next one.
- **Cycle:**
  - it is a path consisting of atleast 3 vertices that starts and ends with same vertex.
- **Loop:**
  - Special case of a cycle in which a single arc begins and ends with the same vertex.

EDUCLASH Just Another Way To Learn

Terminologies used in graphs

- **Strongly connected:**
  - a graph is **strongly connected** if there is a path from each vertex to every other vertex in the digraph.
- **Weakly connected:**
  - a graph is **weakly connected** if at least two vertices are not connected.
- **Disjoint graph:**
  - If two graphs are not connected

EDUCLASH Just Another Way To Learn

Terminologies used in graphs

- **Degree of a vertex:**
  - Number of lines incident to it.
- **Outdegree of a vertex:**
  - In a diagraph, it is the number of arcs leaving the vertex.
- **Indegree of a vertex:**
  - In a diagraph, it is the number of arcs entering the vertex.

EDUCLASH Just Another Way To Learn

Applications of graphs

- Cities and highways connecting them form a graph.
- Components on a circuit board with connections among them form a graph.
- An organic chemical compound can be considered as a graph
- Message transmission in a network

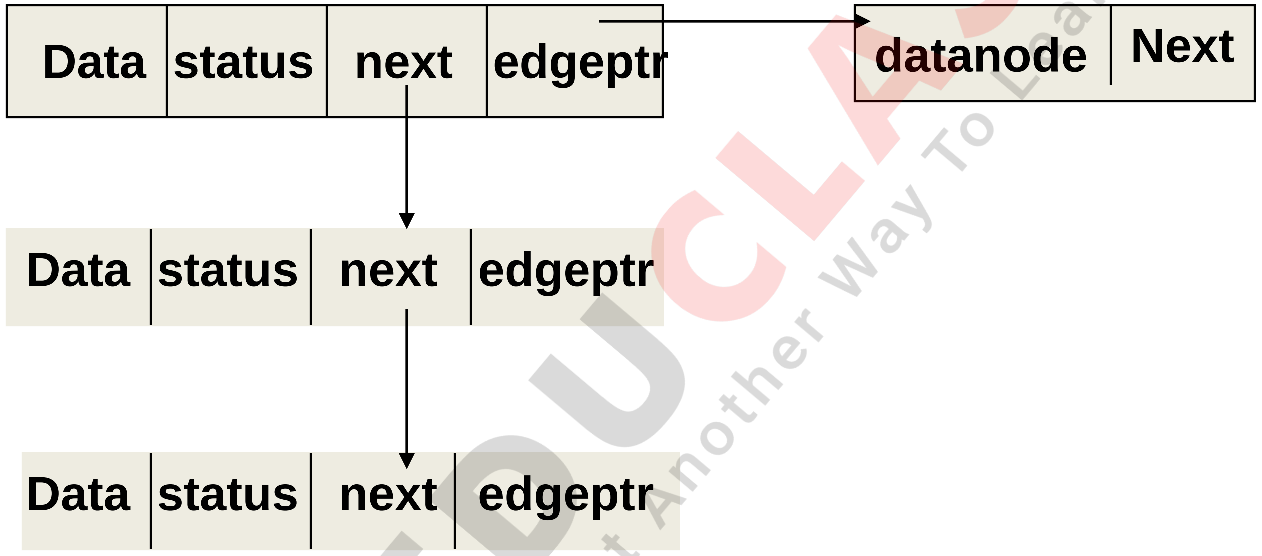
EDUCLASH Just Another Way To Learn

Six primitive graph operations

- Add vertex
- Delete vertex
- Add edge
- Delete edge
- Find vertex
- Traverse graph

EDUCLASH Just Another Way To Learn

Structure of a graph



EDUCLASH Just Another Way To Learn

Create node

Algorithm create_node return newly created node

```
struct node *create_node()
1.[Declare]
    struct node *temp
2.    temp=create a dynamic node
    temp->next=NULL
    temp->edgeptr=NULL
    temp->status=0
    return temp
```

EDUCLASH Just Another Way To Learn

Insert node

```
algorithm insnode(char data)
1.[ Declare]
    static struct node *temp
2. if(start=NULL)
     1.   start=create_node()
     2.   start->data=data
     3.   temp=start
   else
      1. temp->next=create_node()
     2.temp=temp->next
     3. temp->data=data
```

EDUCLASH
Just Another Way To Learn

Find a node

```
Algorithm struct node *findnode(char data)
1. [Declare and initialize]
      struct node *temp
      temp=start
2. Repeat while(temp->data!=data&&temp!=NULL)
          temp=temp->next
     return temp
```

EDUCLASH Just Another Way To Learn

Create edge

```
Algorithm struct edge *create_edge()
1. struct edge *temp
2.    temp=create edge dynamically
3.    return temp
```

EDUCLASH Just Another Way To Learn

Insert and add edge

Algorithm insedge(char source,char dest)

```
1. struct node *locsource,*locdest
2. locsource=findnode(source)
3. locdest=findnode(dest)
4. locsource->edgeptr
       =addedge
   (locsource->edgeptr,locdest)
```

Algorithm struct edge *addedge (struct edge* startedge,struct node *locdest)

```
1.  struct edge *temp=startedge;
2.  if(temp=NULL)
        1. startedge=create_edge()
        2.startedge->datanode=locdest
        3.startedge->next=NULL
        4. Display
              startedge->datanode->data
    else
        1. Repeat while
              (temp->next!=NULL)
            1. temp=temp->next
        2. temp->next=
                create_edge()
        3. temp=temp->next
        4.  temp->datanode=locdest
        5.  temp->next=NULL;
        6. Display
              temp->datanode->data
3. return startedge
```

EDUCLASH Just Another Way To Learn

Insert Edge

```
Algorithm insedge(char source,char dest)
1. struct node *locsource,*locdest
2. locsource=findnode(source)
3. locdest=findnode(dest)
4. locsource->edgeptr=
       addedge(locsource->edgeptr,locdest)
```

EDUCLASH Just Another Way To Learn

Display Graph

```
Algorithm  display_graph()
1. [Declare]
    struct node *tempnode
    struct edge *tempedge
    tempnode=start
2.Repeat A,B,C,D while(tempnode!=NULL)
    A. Display  tempnode->data
    B. tempedge=tempnode->edgeptr
   C. Repeat  1,2while(tempedge!=NULL)
        1. Display tempedge->datanode->data
        2.    tempedge=tempedge->next
   D tempnode=tempnode->next
```

EDUCLASH Just Another Way To Learn

Graph traversals

Depth first traversal

Breadth first traversal

EDUCLASH Just Another Way To Learn

Depth-First Traversal

– All of a vertex’s descendants are processed before we move to an adjacent vertex.
– The **preorder traversal of a tree is a depth-first traversal**

EDUCLASH Just Another Way To Learn

Depth first traversal

- Working:
  - The DFS start by processing the first vertex of the graph.
  - Select any vertex adjacent to the first vertex and process it.
  - Repeat until a vertex with no adjacent entries is reached
- This logic requires **stack** to complete the traversal.
- Example

EDUCLASH Just Another Way To Learn

DFS algorithm

1. Initialize all nodes to the ready state(status=1).
2. Put the starting node A onto STACK and change its status to waiting state(status=2).
3. Repeat steps 4 and 5 until the STACK is empty.
4. Pop the top node N of STACK. Process N and change its statues to the processed status(status=3).
5. Push onto STACK all the neighbours of N that are in the ready state(status=1) and change their status to the waiting state(status=2).
6. Exit

EDUCLASH Just Another Way To Learn

Another DFS Algorithm

```
Algorithm dfs(adj[][],int nodes,int vertex)
where adj[][] -adjacency matrix
    nodes= total number of nodes
    v = vertex
1. [Initialize]
   visited[vertex]=1
2. Repeat for i=0 to nodes
   a. if (adj[v][i] and the vertex is not visited)
        dfs(adj, nodes,i)
```

EDUCLASH Just Another Way To Learn

Breadth first traversal

- All adjacent vertices of a vertex are processed before going to the next level.
- The breadth-first traversal of a graph follows the same concept of the breadth-first traversal of a tree.

EDUCLASH Just Another Way to Learn

Breadth first traversal

- Working:
  - Start with a starting vertex and after processing , process all of its adjacent vertices.
  - When all of the adjacent vertices have been processed, we pick the first adjacent vertex and process all of its vertices then the second adjacent vertex and process all of its vertices and so on.
- This logic requires **queues** to complete the traversal
- Example
- Algorithm of BFS

EDUCLASH Just Another Way to Learn

BFS algorithm

1. Initialize all nodes to the ready state(status=1).
2. Put the starting node A in QUEUE and change its status to waiting state(status=2).
3. Repeat steps 4 and 5 until QUEUE is empty.
4. Remove the front node N of QUEUE. Process N and change its statues to the processed status(status=3).
5. Add to the rear of QUEUE all the neighbours of N that are in the ready state(status=1) and change their status to the waiting state(status=2).
6. Exit

EDUCLASH Just Another Way To Learn

Another DFS Algorithm

```
Declare  nodes
Algorithm DFS(int vertex)
1.[initialize]
    q[],front=rear=-1
    visited[vertex]=1;
 2.
    a. Increment rear
    b. q[rear]=vertex
    c. Repeat while front!= rear
           1. Increment front
           2. Vertex =q[front]
           3. Display vertex
           4. Repeat for i=0 to nodes
               1.if (adj[v][i] and the vertex is not visited)
                     a. set visited = 1;
                     b. increment rear
                     c. q[rear]= i
```

EDUCLASH Just Another Way To Learn

Graph storage structures

- To represent a graph , **two sets are stored:**
  - **Ist set** represents **the vertices of the graph**
  - **IInd set** represents **the edges of the graph**
- The most common structures used to store these sets are
  - **Arrays**
  - **Linked List**

EDUCLASH Just Another Way To Learn

Adjacency matrix

- **The adjacency matrix** uses a **vector(one-dimensional array) for the vertices** and a **matrix (two-dimensional array) to store the edges.**
- **If two vertices are adjacent i.e there is an edge between, the matrix intersect has a value 1.**
- **If there is no edge between them , the intersect is to value 0.**
- **Example:**

EDUCLASH Just Another Way To Learn

Adjacency matrix

- Disadvantage:
  - The size of the graph must be known before the program starts.
  - Only one edge can be stored between any 2 vertices

EDUCLASH Just Another Way To Learn

Adjacency List

- The **vertex list** is a **singly-linked list** of the vertices in the list. Depending on the application , it could also be implemented using **doubly-linked list or circular linked list.**
- **The edges are stored in single linked list.**

EDUCLASH Just Another Way To Learn

Adjacency List

- **The pointer at the left of the list** links the **vertex entries together.**
- **The pointer at the right in the vertex list is a head pointer to a linked list of edges from the vertex.**
- Example

EDUCLASH Just Another Way To Learn

Network

- A **network** is a graph whose lines are weighted.It is also known as **weighted graph**.
- The meaning of weights depends on the application.
- Eg:
  - An airline might use a graph where
    - Nodes: cities
    - Edges: routes
    - Edge’s weights:miles between 2 cities

EDUCLASH Just Another Way To Learn

Spanning Tree

- A **spanning tree** is a tree that contains **all of the vertices in the graph.**
- **Minimum spanning tree** is chosen with following properties:
  - **Every vertex is included**
  - **Total edge weight= minimum cost between vertices**
- **Minimum spanning tree algorithms:**
  - **Kruskal's algorithm**
  - **Prim's algorithm**

EDUCLASH Just Another Way To Learn

Kruskal’s algorithm

- The edges considered for the inclusion in the spanning tree is as per the increasing order of the cost of different edges.
- An edge is included in the spanning tree only **if it does not form a cycle with the edges that are already present in the spanning tree.**
- **Examples:**
- **Algorithm**

EDUCLASH Just Another Way To Learn

Kruskal's Algorithm

```
T={}
1.   Repeat while T <(n-1) edges and E not empty
         1. choose an edge(v,w) from E of lowest cost.
         2. delete (v,w) from E
         3. If(v,w) does not create a cycle in T
              a. Add(v,w) to T
            else
              a. discard(v,w)
     End while

  2. If T <(n-1) edges
       display "no spanning tree exist for this graph"
```

EDUCLASH Just Another Way To Learn

```
program
Define typedef struct
{
   node1
   node2
   wt
}edge
```

```
Algorthim main()
1.[Declare]
      edge e[100]
      parent[100]
      n,i,j,m,cost = 0
2.  Enter number of nodes :n
3. Repeat for i= 0 to  20
         parent[i]=-1
4.
     1. i = 0
     2. Enter number of edges ;m
     3.  repeat for i=0 to m-1
           1. enter an edge and wt
     4. sortedges(e,m)
     5. display Edges of the tree
```

EDUCLASH Just Another Way To Learn

```
6. i = 0
7.  for i=0 to n-1
      A. if(checkcycle(parent,
                 e[i].node1,
                 e[i].node2))
                 1. cost=cost+
                    e[i].wt;
8.  display cost
```

```
Algorithm sortedges(edge a[],int n)
1.    [Declare]
           i,j
           edge temp
2. Repeat for i=0 to n-1
      1.Repeat for j=i+1 to n
           A. if(a[i].wt>a[j].wt)
                   swap a[i] and a[j]

3.  Repeat for i=0 to n
           Display
      a[i].node1,a[i].node2,a[i].wt
```

EDUCLASH Just Another Way to Learn

```
Algorithm checkcycle(int p[],int i,int j)
Return integer
1.   [Declare and initialize]
        v1,v2
      v1 = i
      v2 = j
2. Repeat while(p[i]>-1)
      A.     i = p[i]
3. Repeat while(p[j]>-1)
      A.     j = p[j]
4.  if(i!=j)
        A. p[j]=i;
        B. Display v1,v2
        C.   return 1
5.  return 0
```

EDUCLASH Just Another Way To Learn

Kruskal’s algorithm

- Kruskal's algorithm can be shown to run in
- $O(E \log E)$ time, or equivalently, $O(E \log V)$ time, all with simple data structures.

EDUCLASH Just Another Way To Learn

Prim’s Algorithm

- **Prim's algorithm** is an algorithm that finds a minimum spanning tree for a connected weighted undirected graph.
- This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. Prim's algorithm is an example of a greedy algorithm
- Example

EDUCLASH
Just Another Way To Learn

Prim’s Algorithm

The array structure is used to store the cost of each node

```
Prim(Graph g)
Pre A: Array to store vertices of the graph
1.    Set A =Vertices of the graph g
2.    Repeat for each vertex u
        1.    Cost[u] =∞
        2.    Vertex_list=NIL
3.    Repeat  a and b  while array A not empty
        a. Find node with the smallest key and remove from A
        b. Repeat for each vertex v belonging to adjacent[u]
            1.    If(weight[u,v]<cost[u] then
                1.    Set vertex_list[v]=u
                2.    Set cost[u]=w(u,v)
```

EDUCLASH Just Another Way To Learn

Prim’s algorithm

- A simple implementation using an adjacency matrix graph representation requires O(V^2) running time.
- Efficiency of Prim's algorithm can be shown to run in time

  O(E log V) where E is the number of edges and V is the number of vertices.

EDUCLASH Just another Way To Learn

Shortest Path Algorithms

- Another common application used with graph requires that we find the shortest path between two vertices in a network
- Shortest path algorithms
  - Dijkstra’s algorithm
  - Warshall’ algorithm
- Examples

EDUCLASH Just Another Way to Learn

Relaxation technique

- This technique consists of testing whether we can improve the shortest path found so far.
- The relaxation technique may or may not decrease the value of the shortest path estimate.
- Algorithm relax(u,v,w)

1. If (d[u] +cost(u,v)< d[v])
   a. d[v]= d[u] + cost(u,v)
   b. vertex_list[a] = u
2. Example(pg 433)

EDUCLASH Just Another Way To Learn

Dijkstra’s algorithm

- This algorithm solves the shortest path problem when all edges have non-negative weights.
- **Dijkstra algorithm(G, cost, S)**

```
1. [initialize]
        set S={}  S will contain vertices of final
                  shortest path cost from S
        Queue Q=V(G)   vertices of the graph
2.  repeat a,b,c  while Q not empty
      a. set u= extract_min(Q)   pull out new vertex
      b. set s= s U {u}
      c. repeat  1 for  each vertex v adjacent to u
         1. relax(u,v, cost)
```

- **Analysis Dijkstra’s algorithm runs   O(E log V) times.**

EDUCLASH Just Another Way To Learn

Floyd-Warshall Algorithm

- Basic concept:
  - If for a path(u,v) and its length estimate D[u][v], if we detour(go through) via W and shorten the path, then it should be taken.
  - This translates into the following equation:

    D[u][v]=min(D[u][v], D[u][w] + D[w][v])

- **Algorithm floyd-warshall(W)**

```
1. set n=row[w]
2. set $D^{(0)}$ =W
3. repeat  A. for k= 1 to n
      A. repeat 1. for i = 1 to n
            1. repeat for j=1 to n
               set $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
4. return D(n)
```

ANALYSIS : the algorithm run O(n^3) times

EDUCLASH Just Another Way To Learn

Assignments

- Find out the efficiency of Kruskal's , Prim's and Dijkstra's algorithm
- Write short notes on warshall's algorithm with an example

EDUCLASH Just Another Way To Learn