![alt text](https://s27.postimg.org/hxphs5hb7/cblogo.png)

# Glimpse of Graph

Formally, we define a graph as being composed by two sets, V and E, respectively denoting the set of vertices and the set of edges. We say that vertices are connected by edges, so an edge connects two vertices together. If we generically denote a graph as G, we usually denote it by its two sets, such that the following notation is very common:

G(V, E) - is the graph G composed by V vertices and E edges.

<br>

####Lets see how to traverse a graph : <br>

##### Traversing Graph \*\*BFS\*\* -> Breadth First Search

```

bfs(u):

queue<int> q;

visited[u]=true;

q.push\_back(u);

while(!q.empty())

u=q.front()

q.pop\_front()

for(i=0;i<tree[u].size();i++)

v = tree[u][i]

if(visited[v]=false)

q.push\_back(v)

visited[v]=true

```

##### Traversing Graph \*\*DFS\*\* -> Depth First Search

```

dfs(v):

visited(v)=true

for i = graph.begin() to graph.end()

if(visited(i)=false)m

dfs(i)

```

<br>

Shortest paths from source to all vertices in the given graph.

##### Dijikstra Algo

```

void dijkstra(int start)

{

int t1,i,u,v,cost=0;

// making such that it will give minimum value at first sight

priority\_queue< pii, vector< pii >, greater< pii > > Q;

memset(d,0x3f,sizeof(d));

d[start]=0; // from where i have to start

Q.push(pii(d[start],start));

while(!Q.empty())

{

u = Q.top().second; // node

cost = Q.top().first; // value of node

Q.pop();

if(d[u]<cost) continue; // already better value so neglect this one

for(i=0;i<graph[u].size();i++)

{

v = graph[u][i].first; // node

t1 = graph[u][i].second; // value

if(d[v]>(d[u]+t1))

{

d[v]=d[u]+t1;

Q.push(pii(d[v],v));

}

}

}

}

```

<br><br>

##### MST

```

Kruskal()

solve all edges in ascending order of their weight in an array e

ans = 0

for i = 1 to m

v = e.first

u = e.second

w = e.weight

if merge(v,u) // there will be no cycle

then ans += w

Time complexity : .

```

\*\*Note : If you want we can discuss Prim's, Burkwa's and all other algo with name we dont know how to spell :D \*\*

Okay so best point :

what is the difference between dijikstra and MST ?

:o

##### Bipartite Graph

Try implementing or think how can we implement with dfs and bfs both

<br><br>

#### \*\*Few Topics you have to implement by own:\*\*

1) Kosaraju's Algo and Tarjan's Algo

( Nothing simple SCC :p )

\*\* Note : If someone take algo name from now on.. simply smile ;) .. Ab kisi ne kitaab likh di hai toh hum kuch bol thodi skte hai :D \*\*

2) Longest Path in graph ( Cyclic and Acyclic )

3) Topo Sort

4) Printing all Topo's

5) All child sum

6) Parent finding

7) Print According to Finish time

8) Print all files in unknown folders ( recursive folder structure )

9) Maximum Spanning Tree

<br><br>

#### Few Applications

Simple Applications you can think by yourself :

<br>

1) Max Flow

( Dont think now you cannot implement this by own ;) .. Will discuss later )

2) Eulerian Tours

<br><br>

####Unknown Algo

Shortest path between all nodes to all nodes

```

Unknown\_algo()

d[v][u] = inf for each pair (v,u)

d[v][v] = 0 for each vertex v

for k = 1 to n

for i = 1 to n

for j = 1 to n

d[i][j] = min(d[i][j], d[i][k] + d[k][j])

Time complexity : O(n3).

```

<br><br>

####Disjoint Sets

( Important topic)

#####Union Find Algo

```

void initialize( int Arr[ ], int N)

{

for(int i = 0;i<N;i++)

Arr[ i ] = i ;

}

//returns true,if A and B are connected, else it will return false.

bool find( int Arr[ ], int A, int B)

{

if(Arr[ A ] == Arr[ B ])

return true;

else

return false;

}

//change all entries from Arr[ A ] to Arr[ B ].

void union(int Arr[ ], int N, int A, int B)

{

int TEMP = Arr[ A ];

for(int i = 0; i < N;i++)

{

if(Arr[ i ] == TEMP)

Arr[ i ] = Arr[ B ];

}

}

```

<br><br>

#####Path Compression

#####\*\*\* Not much effective most of the times\*\*\*

```

int root(int v){return par[v] < 0 ? v : (par[v] = root(par[v]));}

void merge(int x,int y){ // x and y are some tools (vertices)

if((x = root(x)) == (y = root(y)) return ;

if(par[y] < par[x]) // balancing the height of the tree

swap(x, y);

par[x] += par[y];

par[y] = x;

}

```

\*\*Note : We started by taking glimpse only :p \*\*