**Graph Intro, BFS, DFS**

**From where I am going to study and practice Graph,**

* CLRS
* GFG
* Interview bit
* Codechef
* Hackerrank
* Edx competitive programming course
* Graph theory notes

Note : - Here I am not going to write the whole theory and complete explanation. This repository is just for quick revision and to remind me what I have studied till now and provide the link of the original resource from where I have actually studied that topic. Therefore if I want to check the original resource then I can easily check that.

Note:- the order of this file is in that order in which I have studied them.

**From GFG,**

1. **Graph and its Representation**

Theory link: <https://www.geeksforgeeks.org/graph-and-its-representations/>

Problem link: <https://practice.geeksforgeeks.org/problems/print-adjacency-list/0>

Problem Solution:

#include<bits/stdc++.h>

using namespace std;

void add\_edge(vector<int> \*arr, int a, int b){ //pointer to array of vectors

arr[a].push\_back(b);

arr[b].push\_back(a);

}

void print\_list(vector<int> arr,int i){ //int vector

cout<<i;

if(arr.size()>=1){

cout<<"-> ";

for(int i=0;i<arr.size()-1;i++){

cout<<arr[i]<<"-> ";

}

cout<<arr[arr.size()-1];

}

cout<<endl;

}

int main()

{

//code

// This code works well in othere ide

long long t;

cin>>t;

while(t--){

int v,e,a,b;

cin>>v>>e;

vector<int> arr[v]; // array(of size v) of int vectors

for(int i=0;i<e;i++){

cin>>a>>b;

add\_edge(arr,a,b); // passing array of vectors

}

for(int i=0;i<v;i++){

print\_list(arr[i],i); // passing a vector row

}

}

return 0;

}

Method 2: use STL linked list : see my another solution

#include<bits/stdc++.h>

using namespace std;

void add\_edge(list<int> \*lst, int a, int b){ //pointer to array of lists

lst[a].push\_back(b);

lst[b].push\_back(a);

}

void print\_list(list<int> lst,int i){ //int list

cout<<i;

if(lst.size()>=1){

cout<<"-> ";

// for(int i=0;i<lst.size()-1;i++){

// cout<< lst[i]<<"-> "; this will not work because list don't have indexing

// }

// cout<< lst[lst.size()-1];

int c=0,s=lst.size();

for(auto item:lst) {

c++;

if(c<s)

cout<<item<<"-> ";

}

cout<<lst.back();

}

cout<<endl;

}

int main()

{

//code

// stl list implementation

long long t;

cin>>t;

while(t--){

int v,e,a,b;

cin>>v>>e;

list<int> lst[v]; // array(of size v) of int list

for(int i=0;i<e;i++){

cin>>a>>b;

add\_edge(lst,a,b); // passing array of lists

}

for(int i=0;i<v;i++){

print\_list(lst[i],i); // passing a list row

}

}

return 0;

}

Method 3: this will work for any kind of node string etc. and using hashmap.

1. **BFS**

Theory link: <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

Problem link: