Assignment:2 119cs0178

* Aim: To design a network topology of arbitrary number of nodes(Hypercube and Hybrid Topology) and also the adjacency list, matrix and degree of node.
* procedure:
* The hypercube is treated to be a loosely coupled system. This system is composed of N = 2n processors that are linked in an n-dimensional binary cube. Each processor denotes a node of the cube.
* A hybrid topology is a kind of network topology that is a combination of two or more network topologies, such as **star topology, bus topology**, and **ring topology**.
* We take the number of nodes by user input and then proceeded to create the graph.
* Then we print the adjacency list and DFS of the graph.
* Software and language : windows and c++.
* Code:
* Hypercube network :

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void showAdjList(vector<vector<int>> &g)

{

int i = 0;

for (auto v : g)

{

cout << i << "==> ";

for (auto x : v)

{

cout << x << " ";

}

cout << "\n";

i++;

}

}

void dfs(vector<vector<int>> &g, int c, vector<bool> &visited) {

if (visited[c]) return;

cout << c << " ";

visited[c] = true;

for (auto x : g[c])

{

if (!visited[x])

dfs(g, x, visited);

}

}

int main()

{

int n;

cout << "Enter the order of graph: ";

cin >> n;

vector<vector<int>> network(1);

if (n == 0)

{

cout << "Adjacency Matrix : \n";

showAdjList(network);

vector<bool> visited(n, false);

dfs(network, 0, visited);

}

int cnt = 1,k,i;

for (i = 1; i <= n; i++)

{

vector<vector<int>> network2 = network;

for (k = 0; k < network.size(); k++)

{

network2[k].push\_back(k + cnt);

network2.push\_back(vector<int>{k});

}

for (k = 0; k < network.size(); k++)

{

for (auto x : network[k])

{

network2[k + cnt].push\_back(x + cnt);

}

}

network = network2;

cnt = cnt \* 2;

}

cout << "Adjacency Matrix : \n";

showAdjList(network);

vector<bool> visited(n, false);

cout << "The DFS traversal: ";

dfs(network, 0, visited);

}

* Hybrid Network :

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void showAdjList(vector<vector<int>> &g)

{

int i = 0;

for (auto v : g)

{

cout << i << "==> ";

for (auto x : v)

{

cout << x << " ";

}

cout << "\n";

i++;

}

}

void dfs(vector<vector<int>> &g, int c, vector<bool> &visited) {

if (visited[c]) return;

cout << c << " ";

visited[c] = true;

for (auto x : g[c])

{

if (!visited[x])

dfs(g, x, visited);

}

}

int main()

{

int sn, rn, bn;

cout << "Enter the total no of nodes in ring network: ";

cin >> rn;

cout << "Enter the total no of nodes in bus network: ";

cin >> bn;

cout << "Enter the total no of nodes in star network: ";

cin >> sn;

vector<vector<int>> network(sn + rn + bn);

//RING NETWORK

for (int i = 1; i < rn; i++)

{

network[i].push\_back(i - 1);

network[i - 1].push\_back(i);

}

network[rn - 1].push\_back(0);

network[0].push\_back(rn - 1);

//BUS NETWORK

for (int i = rn; i < rn + bn; i++)

{

network[i].push\_back(i - 1);

network[i - 1].push\_back(i);

}

//STAR NETWORK

for (int i = rn + bn; i < sn + bn + rn; i++)

{

network[rn + bn - 1].push\_back(i);

network[i].push\_back(rn + bn - 1);

}

cout << "Adjacency Matrix : \n";

showAdjList(network);

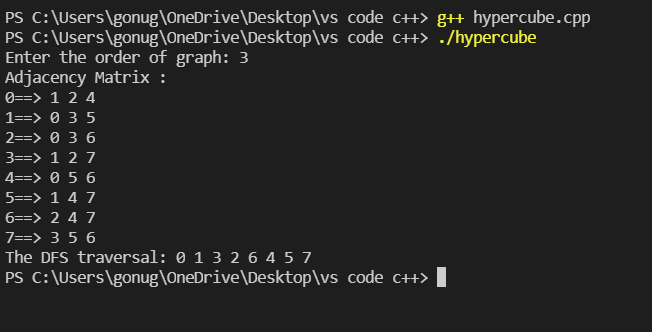
vector<bool> visited(sn + rn + bn, false);

cout << "The DFS traversal: ";

dfs(network, 0, visited);

}

* Input and Outputs:
* Hypercube Network:



* Hybrid Network:

