AIUB

CSC 1203: Programming Language 2[EEE]

Summer 2019-2020

BFS and DFS in C++

Supta Richard Philip

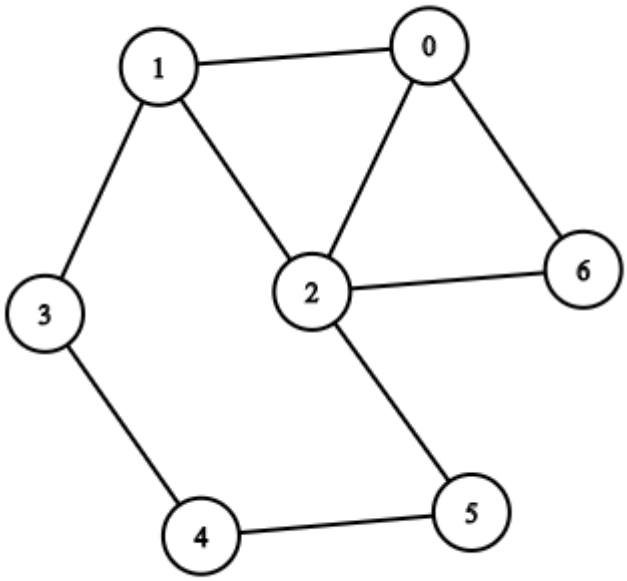
richard@aiub.edu

---

1. BFS

```
#include<bits/stdc++.h>
using namespace std;

#define MAX 100

vector <int> graph[MAX];
int dist[MAX];
bool visited[MAX];

void bfs(int source){

queue<int> Q;

visited[source]=1;
dist[source]=0;
Q.push(source);

while(!Q.empty()){

    int node = Q.front();
    Q.pop();

    for(int i=0;i<graph[node].size();i++){

        int next = graph[node][i];

        if(visited[next]==0){
            visited[next]=1;
            dist[next] = dist[node]+1;
            Q.push(next);
        }

    }
}

}

void printGraph(vector<int> graph[], int n){
    cout<<"The graph is: "<<endl;
```

```
    for(int i=0;i<n;i++){
        cout<<i<<" ->";
        for(int j=0;j<graph[i].size();j++){
            cout<<graph[i][j]<<" ";
        }
        cout<<endl;

    }

}

int main(){

int node, edges;

cin>>node>>edges;

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

        int u,v;
        cin>>u>>v;
        graph[u].push_back(v);
        graph[v].push_back(u);
}
 printGraph(graph,node);

int source;

cin>>source;

bfs(source);

cout<<"From node "<<source<<endl;

for(int i=0;i<node;i++){
    cout<<"Distance of "<<i <<" is "<<dist[i]<<endl;
}
return 0;
}
```



Graph Data for https://csacademy.com/app/graph_editor/

Node Count:

7

Graph Data: 0

1

2

3

4

5

6

0 1

0 6

0 2

1 2

1 3

3 4

4 5

2 5

2 6

input data for algorithm:

node edge

7 9

edges

0 1

0 6

0 2

1 2

1 3

3 4

4 5

2 5

2 6

intput source: 0

2. Input from file

```
#include<stdio.h>
#include<iostream>
using namespace std;
int main(){

    freopen("in.txt","r",stdin);
    int a,b;
    scanf("%d %d",&a,&b);

    printf("%d %d",a,b);


return 0;
}
```

in.txt

1 5

4. Graph input from file(Adjacency matrix)

```
#include<iostream>
using namespace std;
```

```cpp
int adj[10][10];

int main(){

freopen("input.txt","r",stdin);
//freopen("out.txt","w",stdout);

int node,edge;
//printf("Enter the number of node: ");
//scanf("%d",&node);
cin>>node;
//printf("Enter the number of edge: ");
//scanf("%d",&edge);
cin>>edge;

int u,v;
for(int i=0;i<edge;i++){
    //scanf("%d %d",&u,&v);
    cin>>u>>v;
    adj[u][v]=1;
    adj[v][u]=1;
}

for(int i=0;i<node;i++){
        for(int j=0;j<node;j++){
            //printf("%d ",adj[i][j]);
            cout<<adj[i][j]<<" ";

        }
    printf("\n");
}

printf("Adjacent of 1:\n");
for(int j=0;j<node;j++){
            if(adj[1][j]==1){

                cout<<j<<" ";
            }


        }

return 0;
}
```

input.txt

```
7 8

0 1

0 3

1 2
```

2 3

2 6

4 3

4 5

5 6

4. Queue

```
#include<bits/stdc++.h>
using namespace std;
int main(){

freopen("queue.txt","w",stdout);

queue<int> Q;
Q.push(10);
Q.push(20);
Q.push(30);
Q.push(40);

while(!Q.empty()){
    int a;
a=Q.front();
Q.pop();

    printf("%d ",a);
}

return 0;
}
```

4. Stack

```
#include<bits/stdc++.h>
using namespace std;

int main(){

    freopen("stack.txt","w",stdout);

stack<int> s;
s.push(10);
s.push(20);
s.push(30);
s.push(40);

while(!s.empty()){
    int a;
    a=s.top();
    s.pop();
```

```
    printf("%d ",a);

}

return 0;
}
```

4. DFS (Adjacency Matrix)

```
#include<iostream>
#include<stack>
using namespace std;

int adj[10][10];
int visited[10]={0};

void DFS(int source,int node){
stack<int> s;
s.push(source);
visited[source]=1;
while(!s.empty()){
    int v= s.top();
    s.pop();
    cout<<v<<" ";
    for(int j=0;j<node;j++){
            if(adj[v][j]==1){

                    if(visited[j]==0){
                        s.push(j);
                        visited[j]=1;
                    }
            }

        }
}

}

int main(){

freopen("input.txt","r",stdin);
//freopen("out.txt","w",stdout);

int node,edge;
//printf("Enter the number of node: ");
//scanf("%d",&node);
cin>>node;
//printf("Enter the number of edge: ");
//scanf("%d",&edge);
cin>>edge;

int u,v;
for(int i=0;i<edge;i++){
```

```
    //scanf("%d %d",&u,&v);
    cin>>u>>v;
    adj[u][v]=1;
    adj[v][u]=1;
}

for(int i=0;i<node;i++){
        for(int j=0;j<node;j++){
            //printf("%d ",adj[i][j]);
            cout<<adj[i][j]<<" ";

        }
    printf("\n");
}
printf("\n\nDFS\n");
DFS(0,node);

return 0;
}
```

input.txt

```
7 8

0 1

0 3

1 2

2 3

2 6

4 3

4 5

5 6
```

4. BFS(Adjacency Matrix)

```
#include<iostream>
#include<queue>
using namespace std;

int adj[10][10];
int visited[10]={0};
int dist[10]={0};

void BFS(int source,int node){

queue<int> q;
visited[source]=1;
q.push(source);
```

```
dist[source]=0;

while(!q.empty()){
    int v=q.front();
    q.pop();
    cout<<v<<" ";
    for(int j=0;j<node;j++){
            if(adj[v][j]==1){

                    if(visited[j]==0){
                        q.push(j);
                        visited[j]=1;
                        dist[j]=dist[v]+1;
                    }
            }
        }
}

}

int main(){


freopen("BFS.txt","r",stdin);
//freopen("out.txt","w",stdout);

int node,edge;
//printf("Enter the number of node: ");
//scanf("%d",&node);
cin>>node;
//printf("Enter the number of edge: ");
//scanf("%d",&edge);
cin>>edge;

int u,v;
for(int i=0;i<edge;i++){
    //scanf("%d %d",&u,&v);
    cin>>u>>v;
    adj[u][v]=1;
    adj[v][u]=1;
}

for(int i=0;i<node;i++){
        for(int j=0;j<node;j++){
            //printf("%d ",adj[i][j]);
            cout<<adj[i][j]<<" ";

        }
    printf("\n");
}
printf("\n\nBFS\n");
```

```
BFS(0,node);
cout<<"\ndistance from node 0: "<<endl;
for(int i=0;i<node;i++){

            cout<<"0 to "<<i<<" "<<dist[i]<<"\n ";
}

return 0;
}
```

BFS.txt

```
8 9

0 1

0 2

0 7

1 2

2 6

1 3

3 4

2 5

4 5
```

4. Adjaceny List using vector

```
#include<iostream>
#include<stdio.h>
#include<vector>
using namespace std;

int main(){

    freopen("input.txt","r",stdin);

    vector<int> vec[8];

    int node, edge;
    //cin>>node>>edge;
    scanf("%d %d", &node,&edge);


    int u,v;

    for(int i=0;i<edge;i++){
        cin>>u>>v;
        vec[u].push_back(v);
        vec[v].push_back(u);
```

```
    }


    for(int i=0;i<node;i++){
        cout<<i<<"->";
        for(int j=0;j<vec[i].size();j++){

            cout<<vec[i][j]<<" ";
        }
        cout<<endl;
    }

return 0;
}
```

input.txt

```
7 8

0 1

0 3

1 2

2 3

2 6

4 3

4 5

5 6
```

4. DFS Adjacency List using vector

```
#include<iostream>
#include<stdio.h>
#include<vector>
#include<stack>
using namespace std;

int visited[7]={0};
vector<int> vec[7];


void DFS(int source){
stack<int> S;
S.push(source);
visited[source]=1;

while(!S.empty()){
    int v = S.top();
    S.pop();
    cout<<v<<" ";
```

```
    for(int i=0;i<vec[v].size();i++){
        if(visited[vec[v][i]]==0){
                //cout<<vec[v][i]<<" ";
            S.push(vec[v][i]);
            visited[vec[v][i]]=1;
        }
    }
}
}

int main(){

    freopen("input.txt","r",stdin);



    int node, edge;
    //cin>>node>>edge;
    scanf("%d %d", &node,&edge);


    int u,v;

    for(int i=0;i<edge;i++){
        cin>>u>>v;
        vec[u].push_back(v);
        vec[v].push_back(u);
    }


    for(int i=0;i<node;i++){
        cout<<i<<"->";
        for(int j=0;j<vec[i].size();j++){

            cout<<vec[i][j]<<" ";
        }
        cout<<endl;
    }

    printf("\n\nDFS\n");
    DFS(0);



return 0;
}
```

input.txt

```
7 8

0 1
```

0 3

1 2

2 3

2 6

4 3

4 5

5 6

4. BFS Adjaceny List

```
#include<iostream>
#include<stdio.h>
#include<vector>
#include<queue>
using namespace std;

int visited[8]={0};
vector<int> vec[8];
int dist[8]={0};


void BFS(int source){
queue<int> q;
q.push(source);
visited[source]=1;
dist[source]=0;
while(!q.empty()){
    int v = q.front();
    q.pop();
    cout<<v<<" ";
    for(int i=0;i<vec[v].size();i++){
        if(visited[vec[v][i]]==0){
                //cout<<vec[v][i]<<" ";
            q.push(vec[v][i]);
            visited[vec[v][i]]=1;
            dist[vec[v][i]]=dist[v]+1;
        }
    }
}
}

int main(){

    freopen("BFS.txt","r",stdin);



    int node, edge;
    //cin>>node>>edge;
```

```
    scanf("%d %d", &node,&edge);


    int u,v;

    for(int i=0;i<edge;i++){
        cin>>u>>v;
        vec[u].push_back(v);
        vec[v].push_back(u);
    }


    for(int i=0;i<node;i++){
        cout<<i<<"->";
        for(int j=0;j<vec[i].size();j++){

            cout<<vec[i][j]<<" ";
        }
        cout<<endl;
    }

    printf("\n\nBFS\n");
    BFS(0);

cout<<"\ndistance from node 0: "<<endl;
for(int i=0;i<node;i++){

            cout<<"0 to "<<i<<" "<<dist[i]<<"\n ";
}


return 0;
}
```

BFS.txt

```
8 9

0 1

0 2

0 7

1 2

2 6

1 3

3 4

2 5

4 5
```

4. Excercise

1. ex1
2. ex2

References:

https://erlerobotics.gitbooks.io/erle-robotics-cpp-gitbook/content/object-oriented_programming__oop_and_inheritance/exercises_oop.html

https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm