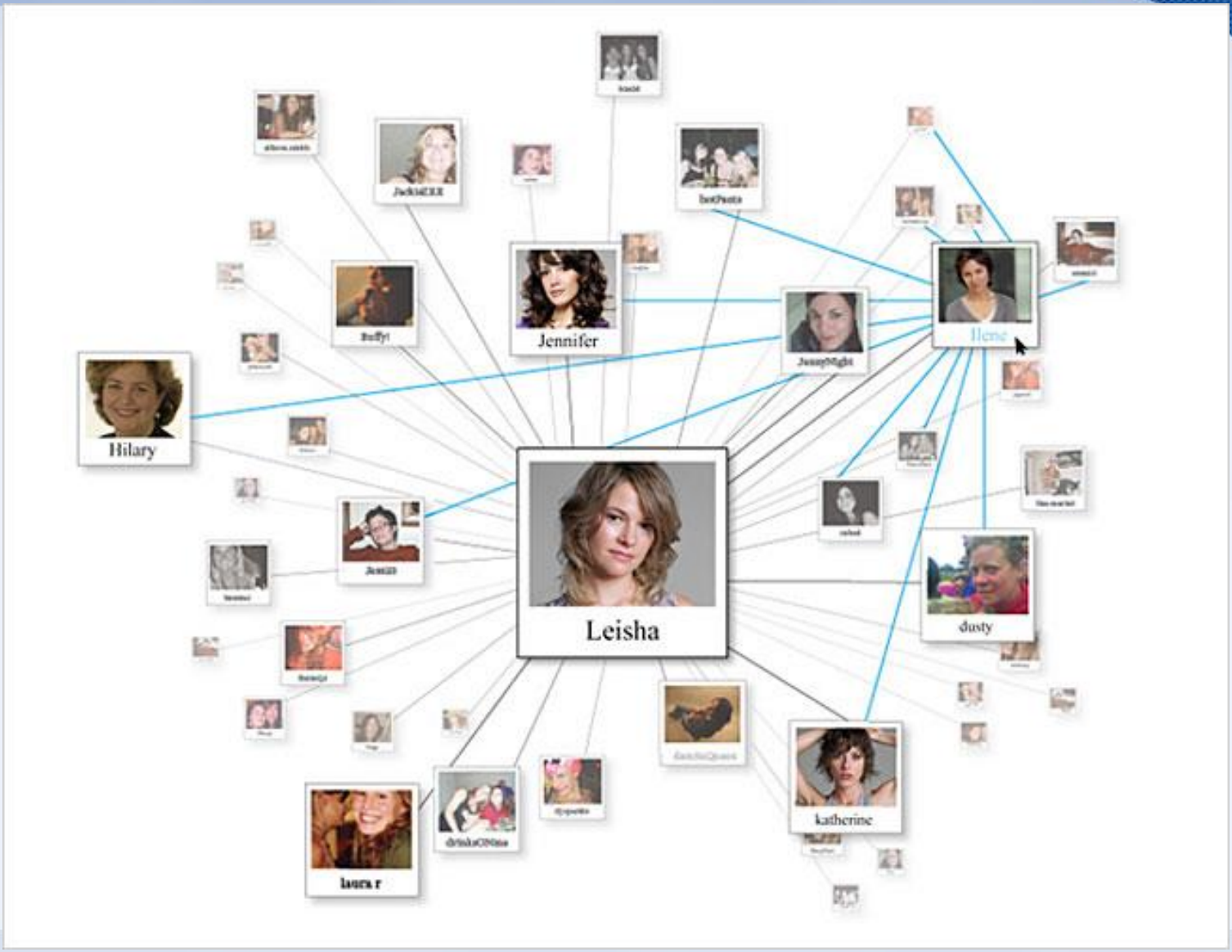
ACM DAMASCUS UNIVERCITY

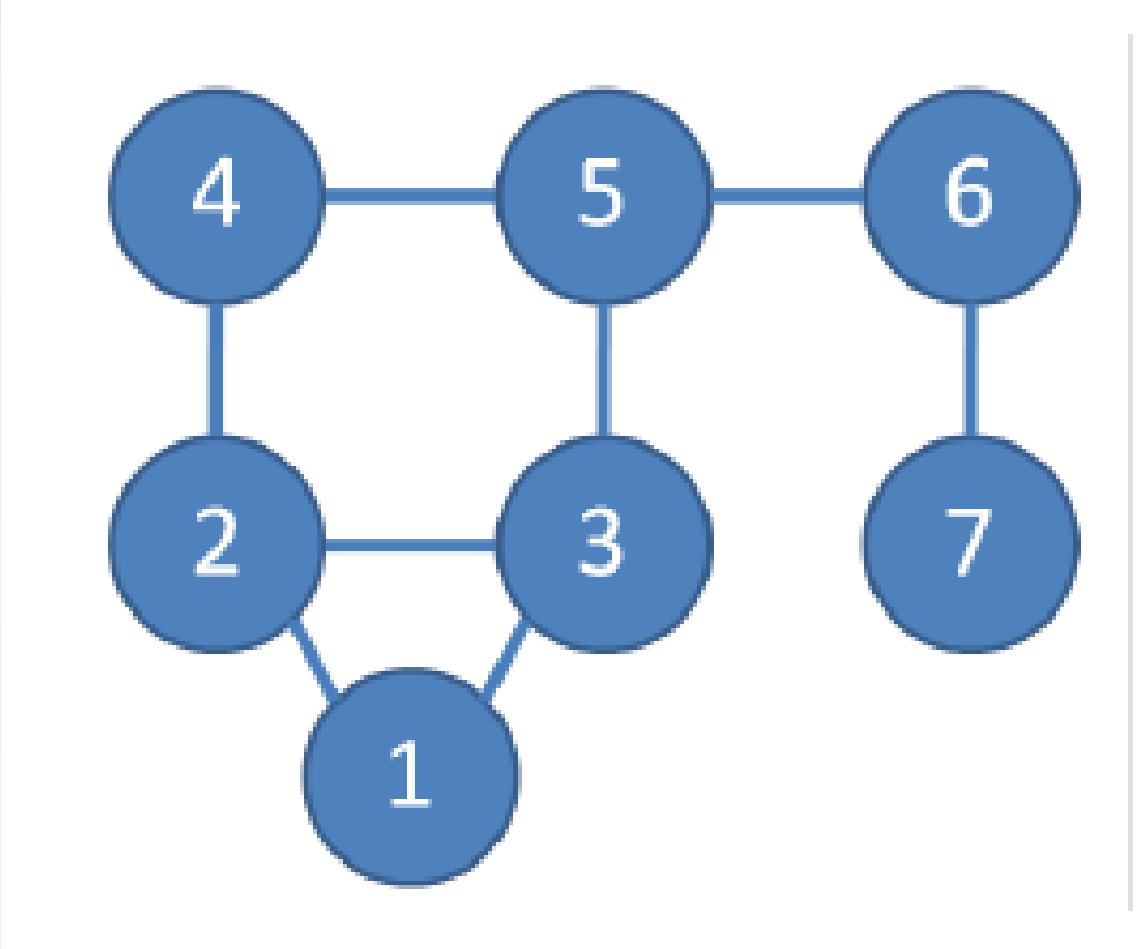
Feras AL-Kassar

Graph


Jennifer
Hilary
Leisha
dusty
katherine
laura r

Why Study Graph ?

- Lots of problems formulated and solved in terms of graphs
  - Shortest path problems
  - Network flow problems
  - Matching problems
  - 2-SAT problem
  - Graph coloring problem
  - Traveling Salesman Problem (TSP): *still unsolved!*
  - and many more…

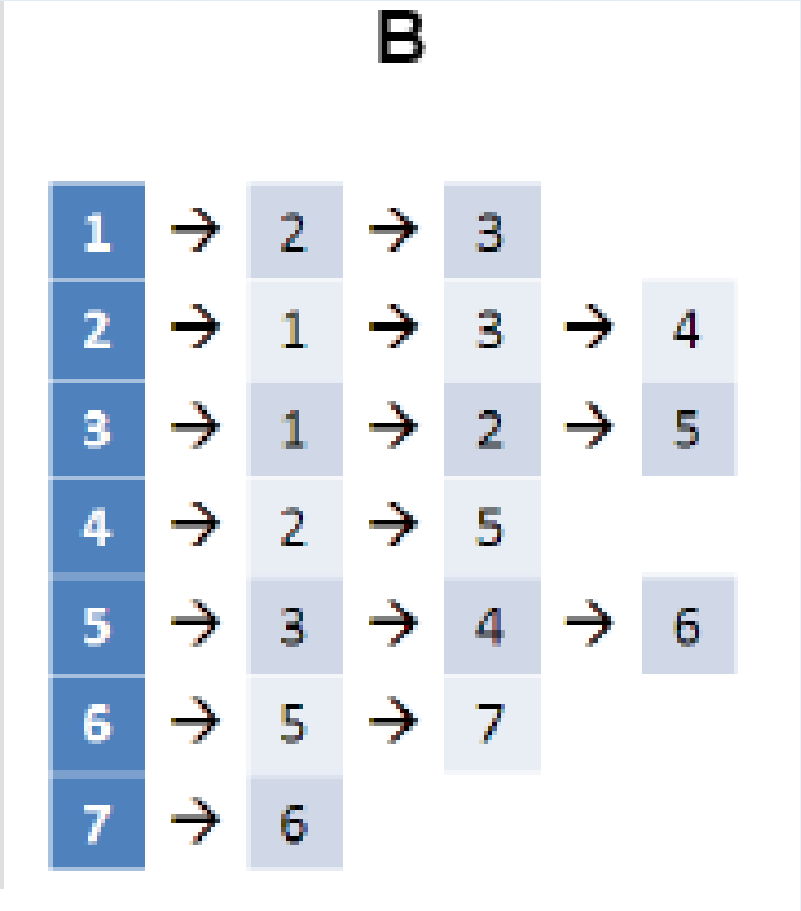

4
5
6
2
3
7
1

Structure 1

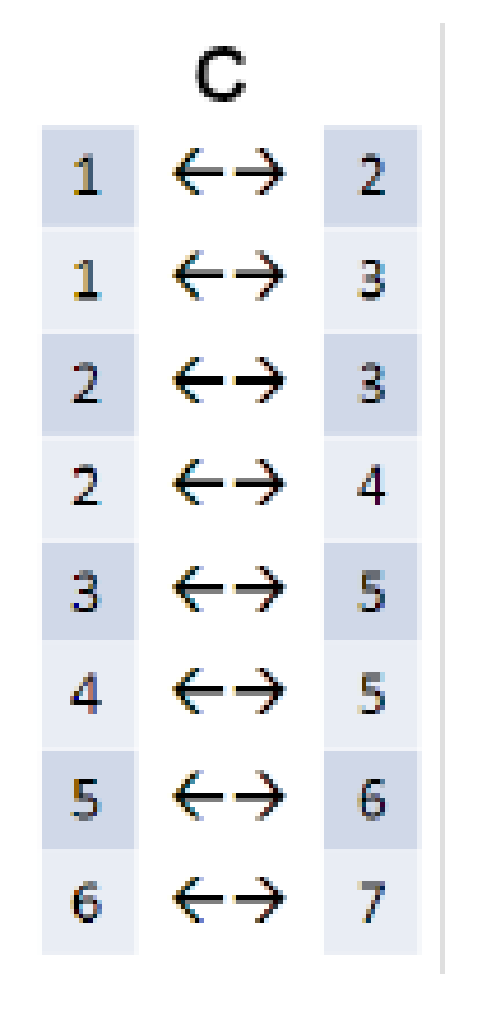
A

	1	2	3	4	5	6	7
1		1	1				
2	1		1	1			
3	1	1			1		
4		1			1		
5			1	1		1	
6					1		1
7						1	

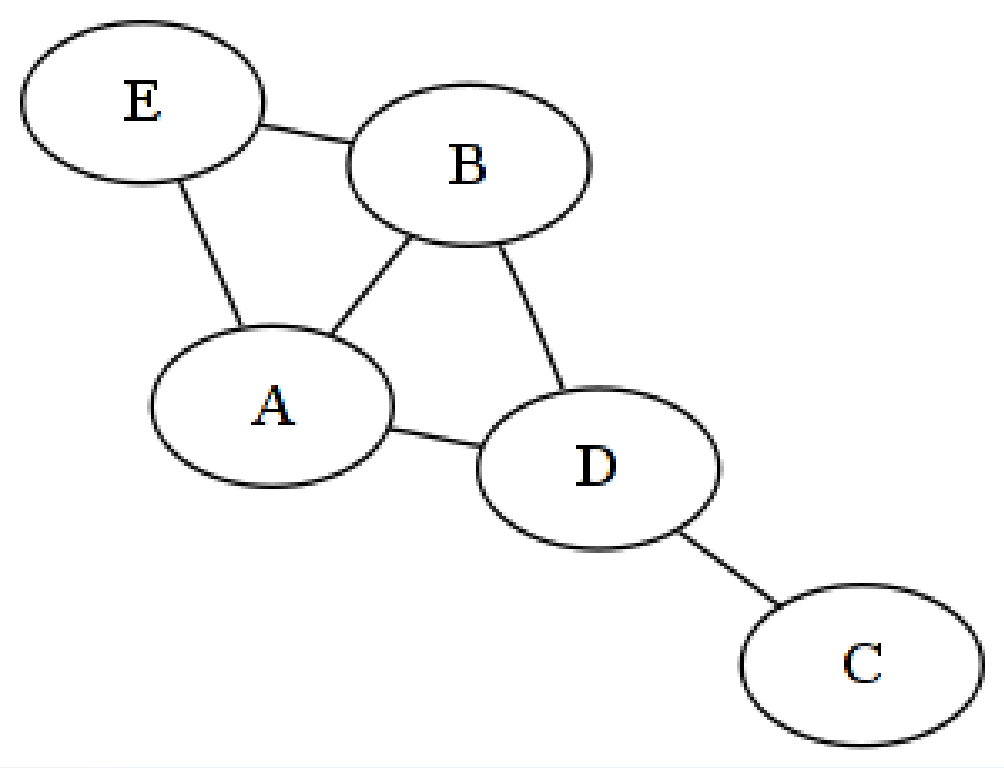
Structure 2



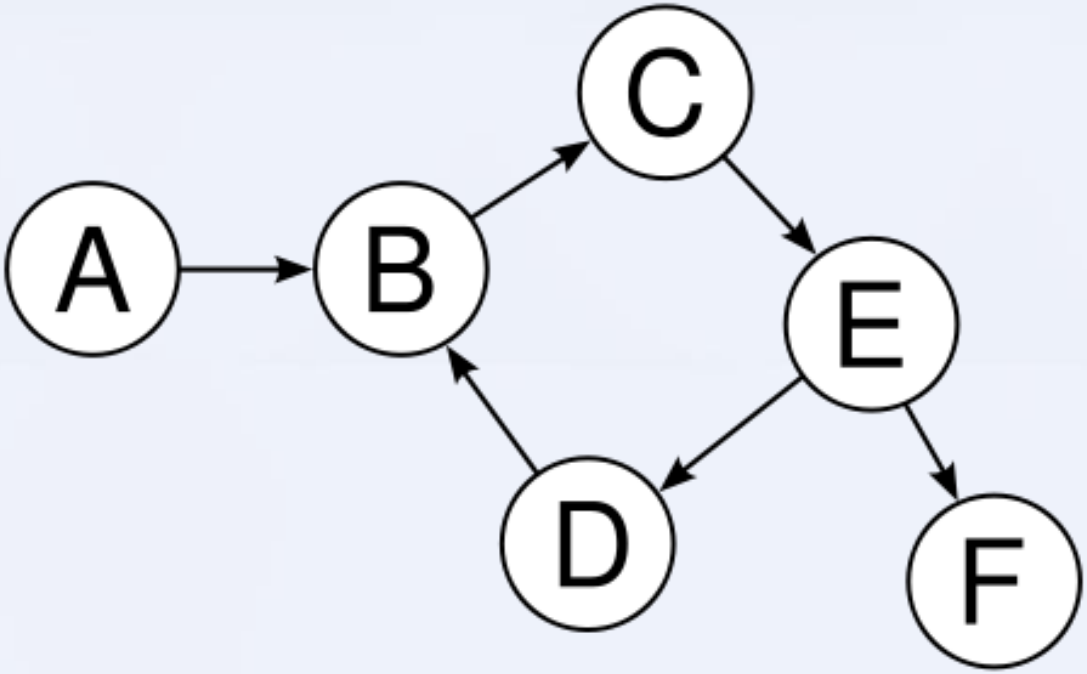
Structure 3



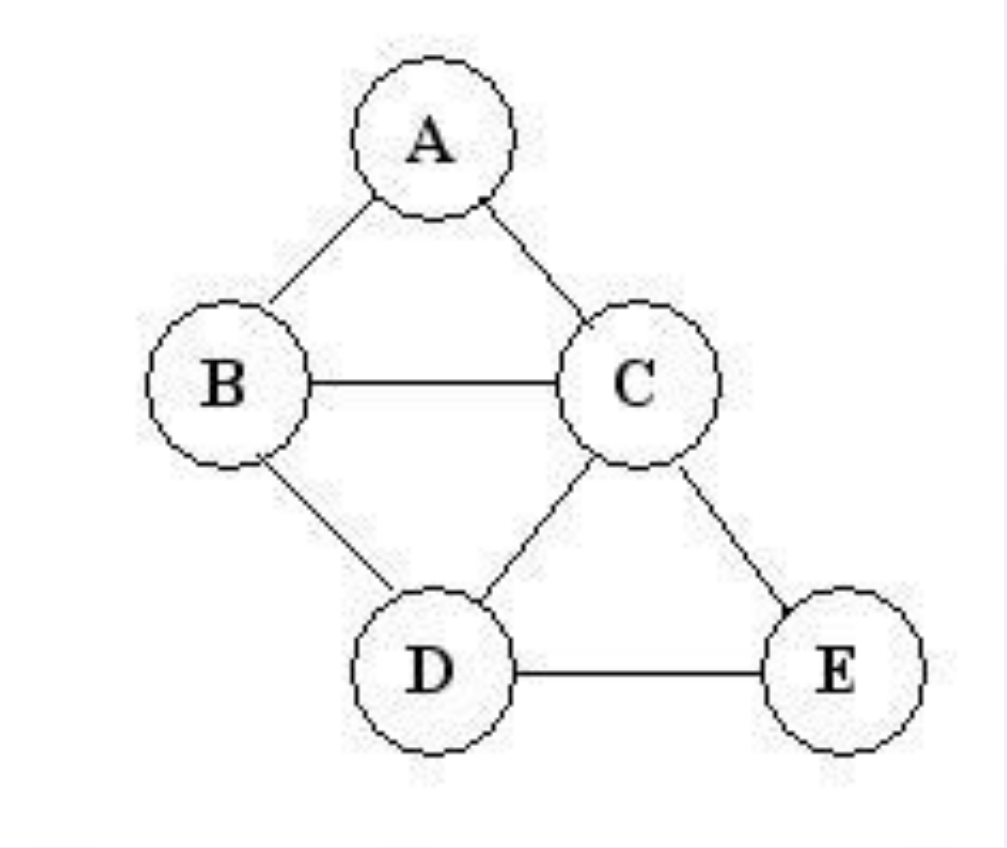
Un-Directed



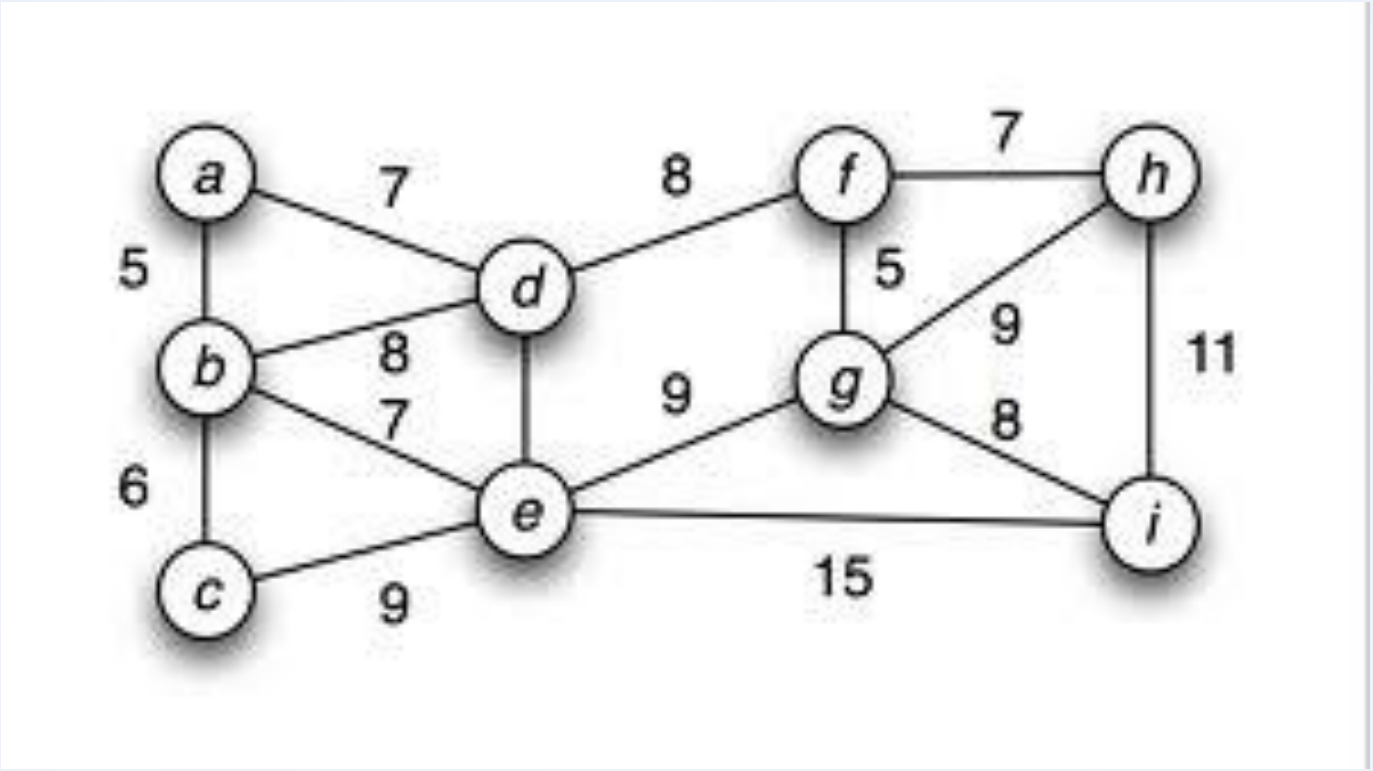
Directed



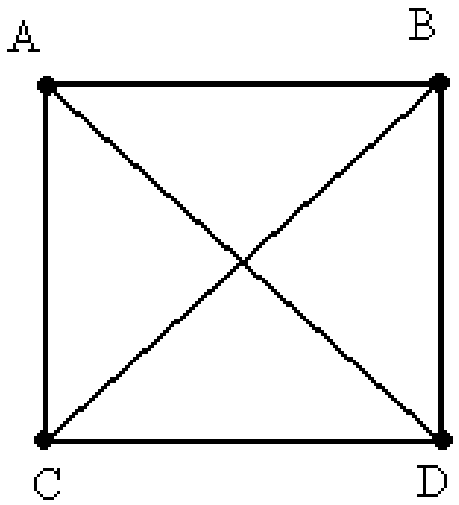
Un-weighted



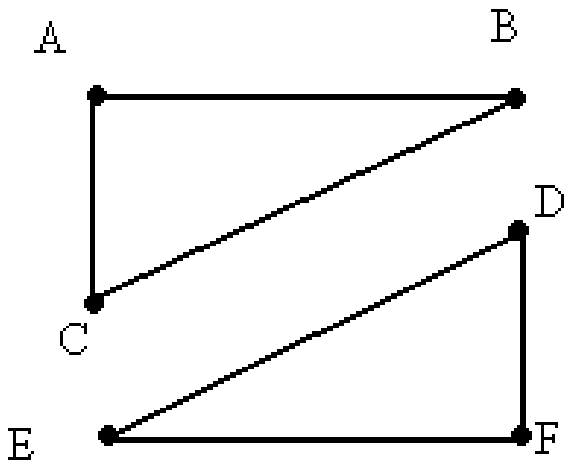
weighted



Connected / dis-connected



Connected graph



Disconnected graph

Input

N: number of vertices
M: number of edges

X Y: edge between X and Y
X Y Z: edge between X and Y weight Z

Structure 1 / un-Directed / un-weight

```
#include<iostream>
using namespace::std;
int main()
{
	int n,m;
	cin>>n>>m;
	int G[50][50]={0};
	for(int i=0;i<m;i++)
	{
		int x,y; cin>>x>>y;
		G[x][y]=1;
		G[y][x]=1;
	}
}
```

Structure 1 / Directed / un-weight

```
#include<iostream>
using namespace::std;
int main()
{
    int n,m;
    cin>>n>>m;
    int G[50][50]={0};
    for(int i=0;i<m;i++)
    {
        int x,y; cin>>x>>y;
        G[x][y]=1;
    }
}
```

Structure 1 / un-Directed / weight

```
#include<iostream>
using namespace::std;
int main()
{
	int n,m;
	cin>>n>>m;
	int G[50][50]={0};
	for(int i=0;i<m;i++)
	{
		int x,y; cin>>x>>y>>z;
		G[x][y]=z;
		G[y][x]=z;
	}
}
```

Structure 1 / Directed / weight

```
#include<iostream>
using namespace::std;
int main()
{
	int n,m;
	cin>>n>>m;
	int G[50][50]={0};
	for(int i=0;i<m;i++)
	{
		int x,y; cin>>x>>y>>z;
		G[x][y]=z;
	}
}
```

Structure 2 / un-Directed / un-weight

```cpp
#include<iostream>
#include<vector>
using namespace::std;
int main()
{
	int n,m;
	cin>>n>>m;
	vector<int> G[50];
	for(int i=0;i<m;i++)
	{
		int x,y; cin>>x>>y;
		G[x].push_back(y);
		G[y].push_back(x);
	}
}
```

Structure 2 / Directed / un-weight

```cpp
#include<iostream>
#include<vector>
using namespace::std;
int main()
{
	int n,m;
	cin>>n>>m;
	vector<int> G[50];
	for(int i=0;i<m;i++)
	{
		int x,y; cin>>x>>y;
		G[x].push_back(y);
	}
}
```

Structure 2 / un-Directed / weight

```cpp
#include<iostream>
#include<vector>
using namespace::std;
int main()
{
    int n,m;
    cin>>n>>m;
    vector<pair<int,int> > G[50];
    for(int i=0;i<m;i++)
    {
        int x,y,z; cin>>x>>y>>z;
        G[x].push_back(make_pair(y,z));
        G[y].push_back(make_pair(x,z));
    }
}
```

Structure 1 / Directed / weight

```
#include<iostream>
#include<vector>
using namespace::std;
int main()
{
    int n,m;
    cin>>n>>m;
    vector<pair<int,int> > G[50];
    for(int i=0;i<m;i++)
    {
        int x,y,z; cin>>x>>y>>z;
        G[x].push_back(make_pair(y,z));
    }
}
```

Note:

```
#include<iostream>
#include<vector>
using namespace::std;
int main()
{
     int n,m;
     cin>>n>>m;
     vector<pair<int,int> > G[50];
     for(int i=0;i<m;i++)
     {
          int x,y,z; cin>>x>>y>>z;
          G[x].push_back(make_pair(y,z));
     }
}
```

Structure 3 / un-Directed / un-weight

```
#include<iostream>
#include<queue>
using namespace::std;
int main()
{
     int n,m;
     cin>>n>>m;
     priority_queue<pair<int,int> >q;
     for(int i=0;i<m;i++)
     {
          int x,y,z; cin>>x>>y;
          q.push(make_pair(x,y));
     }
}
```

Structure 3 / Directed / un-weight

```
#include<iostream>
#include<queue>
using namespace::std;
int main()
{
     int n,m;
     cin>>n>>m;
     priority_queue<pair<int,int> >q;
     for(int i=0;i<m;i++)
     {
          int x,y,z; cin>>x>>y;
          q.push(make_pair(x,y));
     }
}
```

Structure 1 / un-Directed / weight

```
#include<iostream>
#include<queue>
using namespace::std;
int main()
{
     int n,m;
     cin>>n>>m;
     priority_queue<pair<int,pair<int,int> > >q;
     for(int i=0;i<m;i++)
     {
          int x,y,z; cin>>x>>y>>z;
          q.push(make_pair(z,make_pair(x,y)));
     }
}
```

Structure 1 / Directed / weight

```
#include<iostream>
#include<queue>
using namespace::std;
int main()
{
     int n,m;
     cin>>n>>m;
     priority_queue<pair<int,pair<int,int> > >q;
     for(int i=0;i<m;i++)
     {
          int x,y,z; cin>>x>>y>>z;
          q.push(make_pair(z,make_pair(x,y)));
     }
}
```

BFS

```
const int N=50;
int G[N][N];
int visited[N];
int n;

void BFS(int x)
{
    queue<int>q;
    q.push(x);
    while(!q.empty())
    {
        x = q.front(); q.pop();
        visited[x]=1;
        cout<<x<<endl;
        for(int i=0;i<n;i++)
            if(G[x][i] && !visited[i])
                q.push(i);
    }
}
```

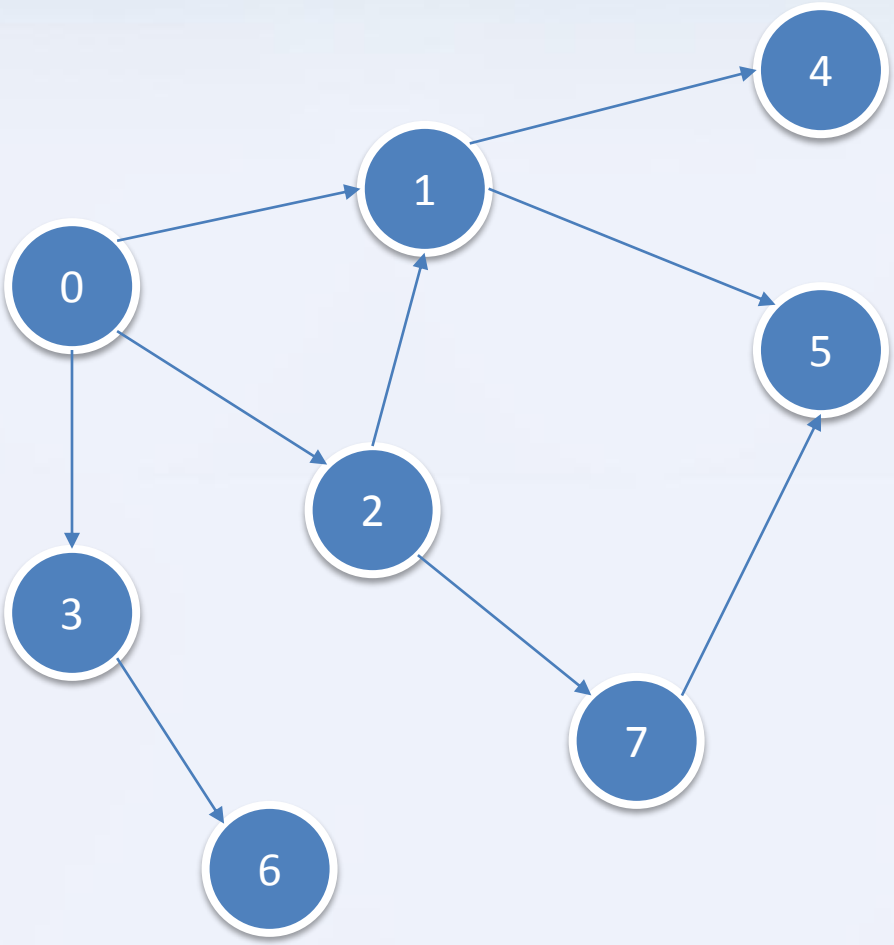
DFS

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

const int N=50;
int G[N][N];
int visited[N];
int n;

void DFS(int x)
{
    visited[x]=1;
    cout<<x<<endl;
    for(int i=0;i<n;i++)
        if(G[x][i] && !visited[i])
            DFS(i);
    visited[x]=0;
}
```

DFS

```
int main()
{
    freopen("in.txt","r",stdin);
    int m;
    cin>>n>>m;
    for(int i=0;i<m;i++)
    {
        int x,y; cin>>x>>y;
        G[x][y]=1;
    }
    DFS(0);
}
```

BFS



BFS

0

1

2

3

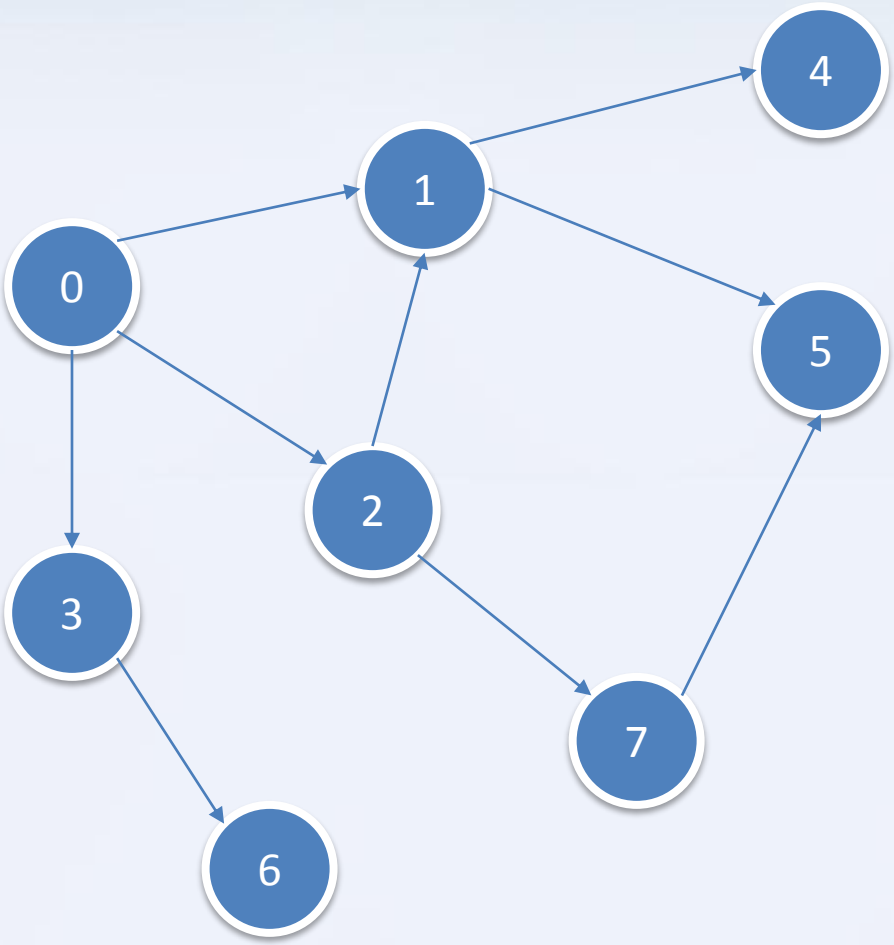
4

5

7

6

DFS



DFS

0

1

4

5

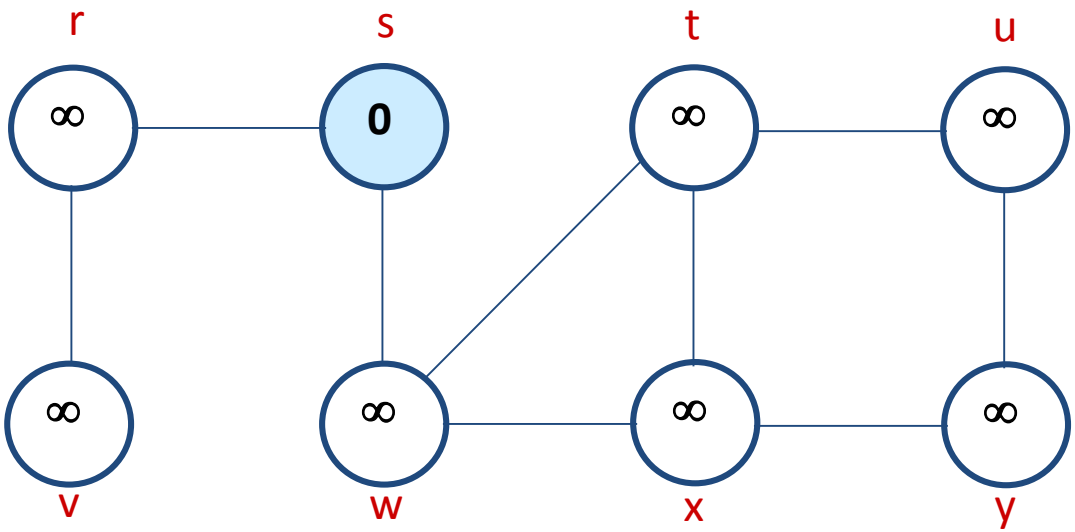
2

7

3

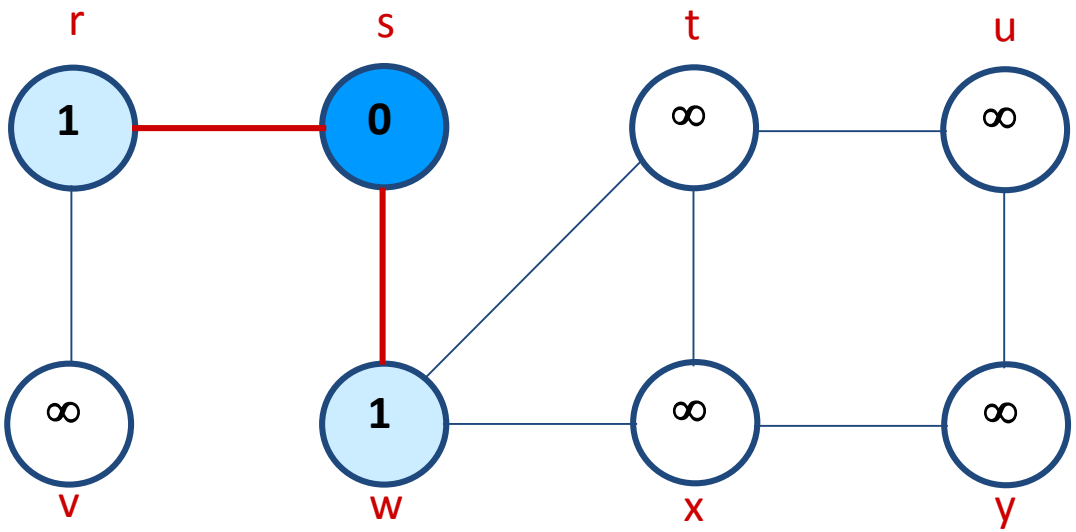
6

Example (BFS)



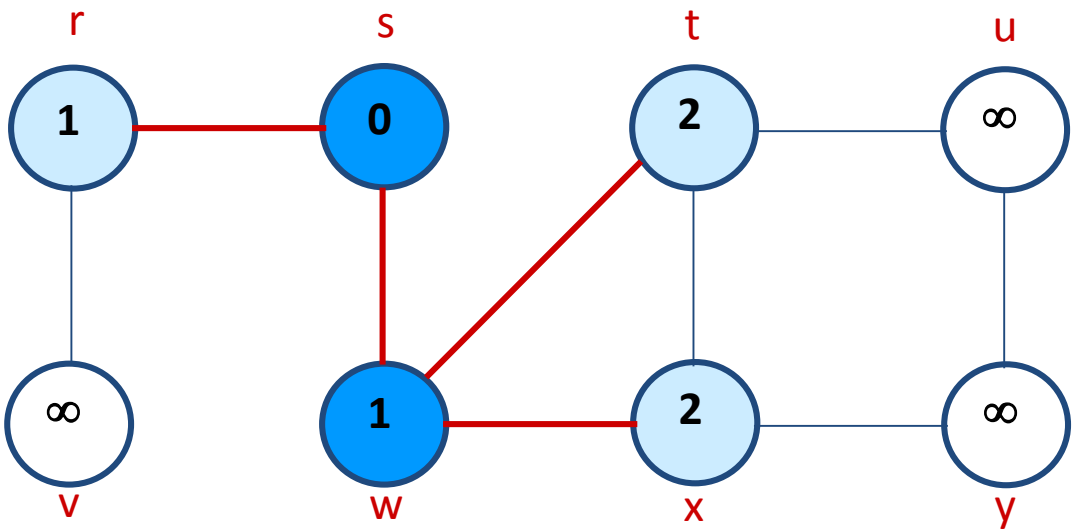
Q: s
0

Example (BFS)



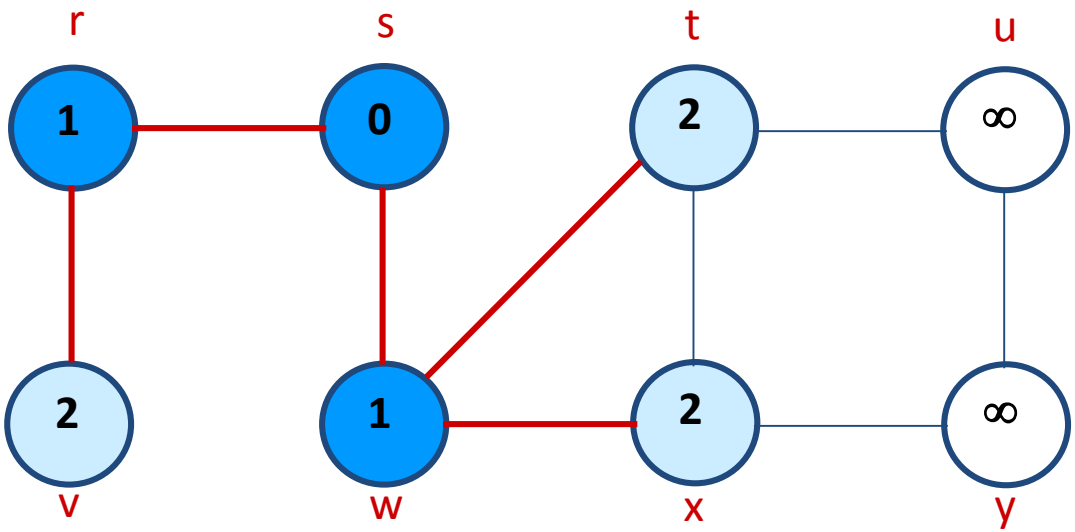
Q: w r
1 1

Example (BFS)



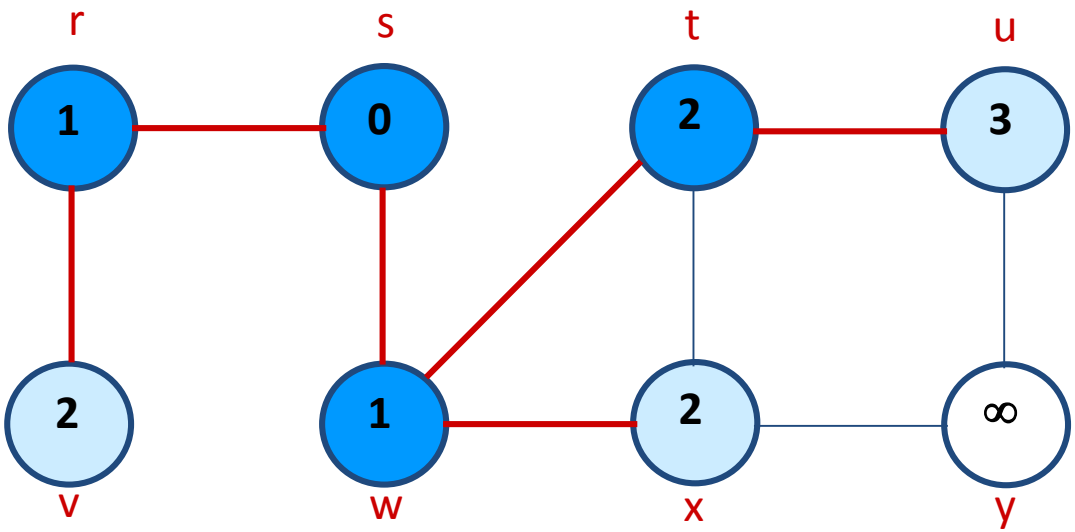
Q: r t x
1 2 2

Example (BFS)



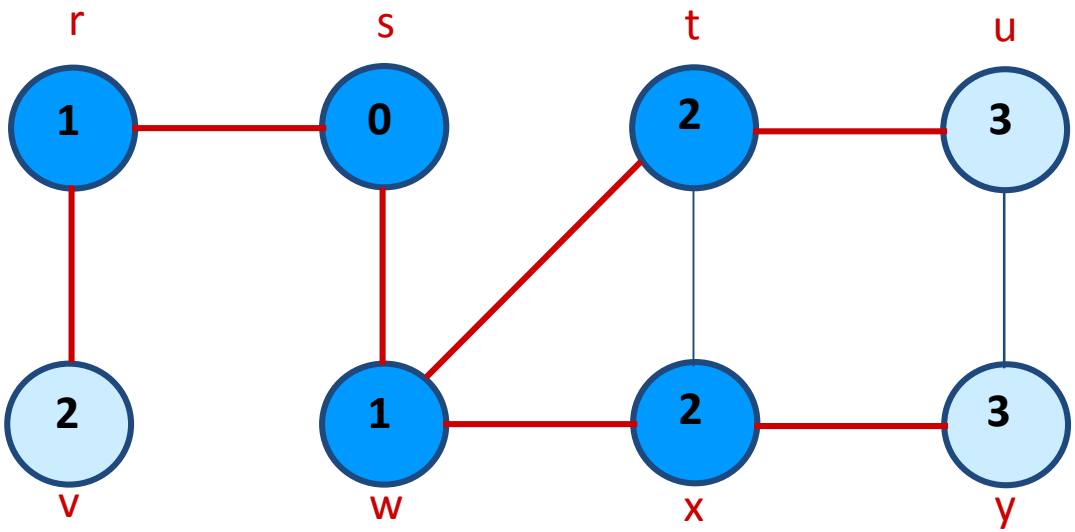
Q: t x v
2 2 2

Example (BFS)



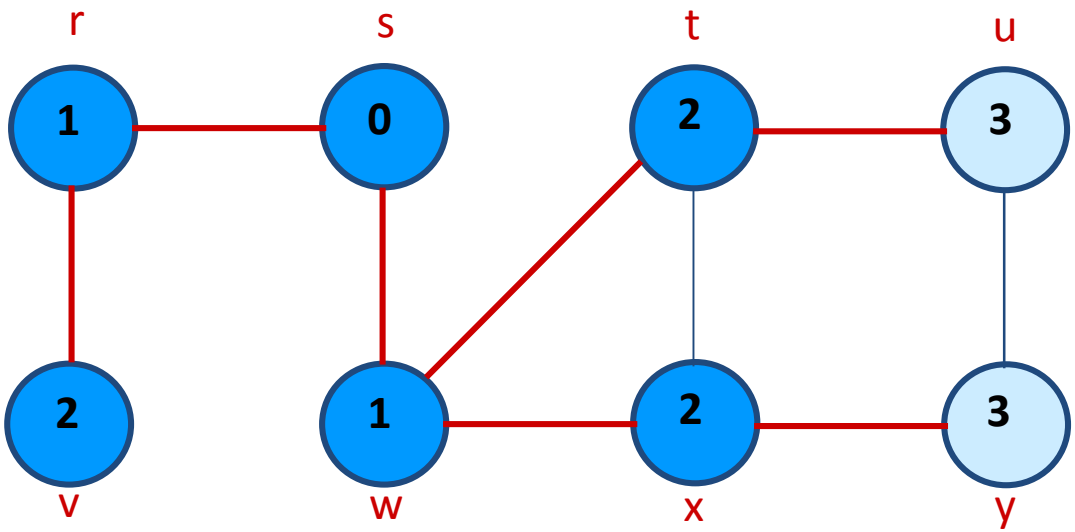
Q: x v u
2 2 3

Example (BFS)



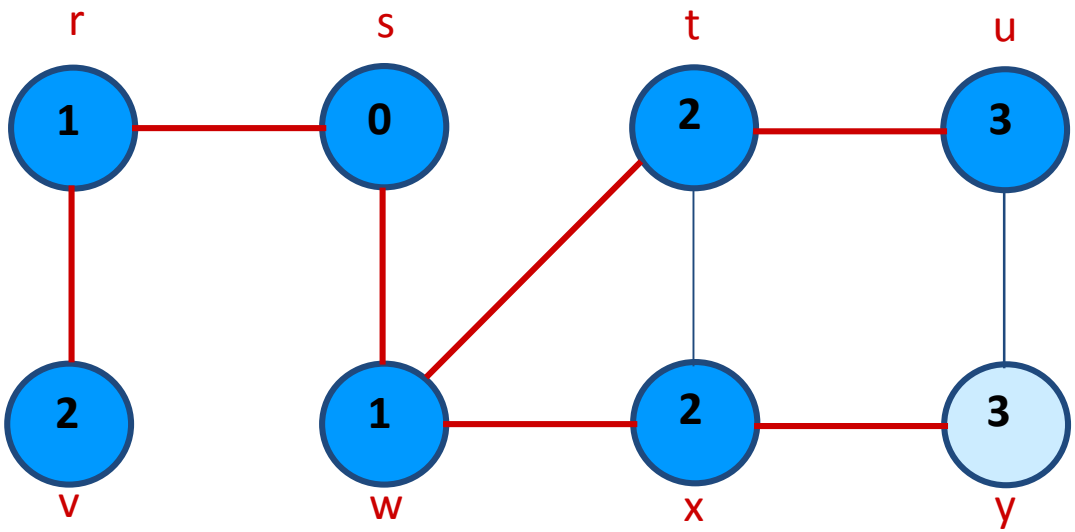
Q: v u y
2 3 3

Example (BFS)



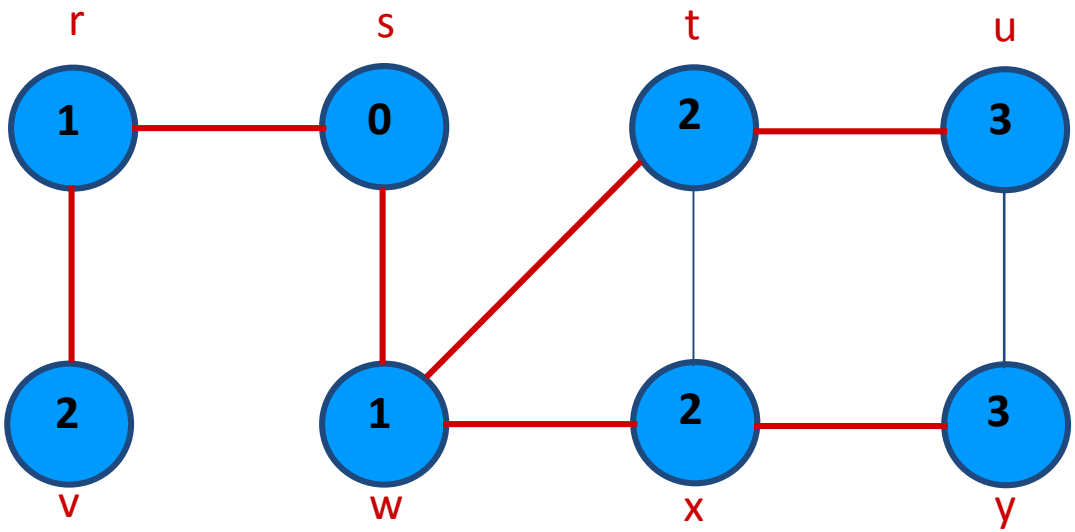
Q: u y
3 3

Example (BFS)



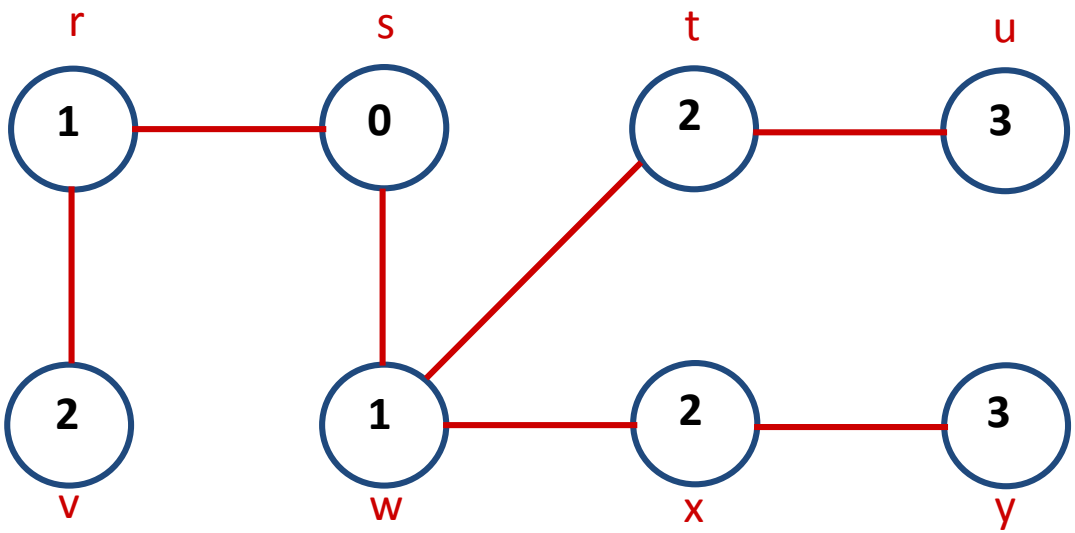
Q: y
3

Example (BFS)



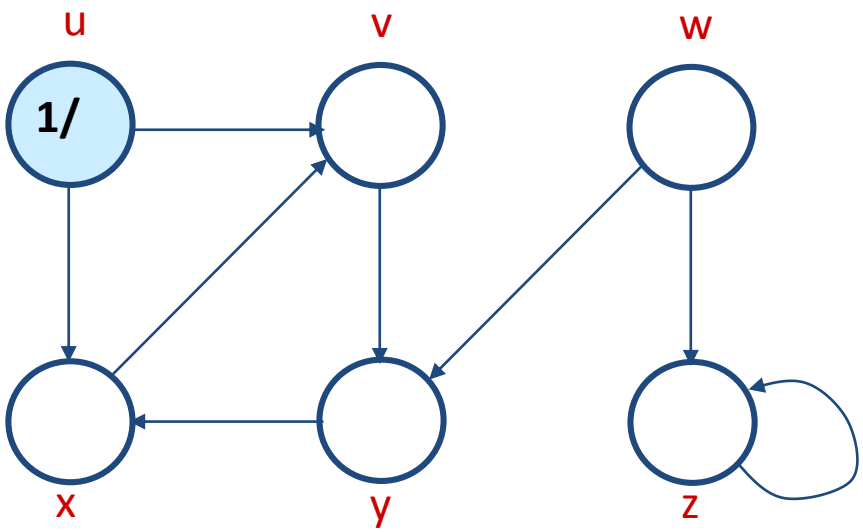
Q: ∅

Example (BFS)

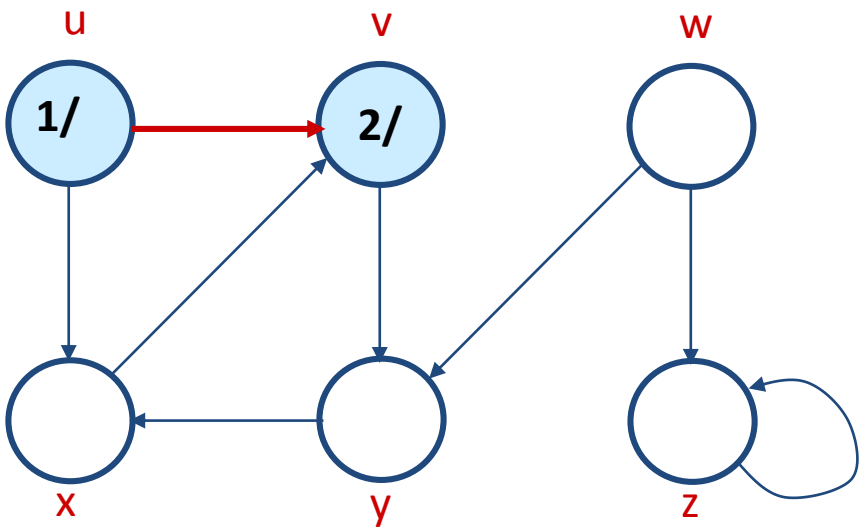


BF Tree

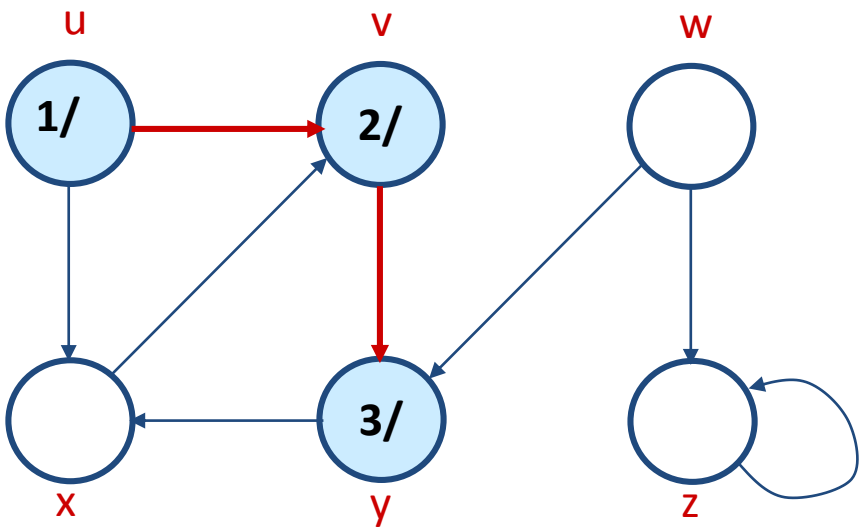
Example (DFS)



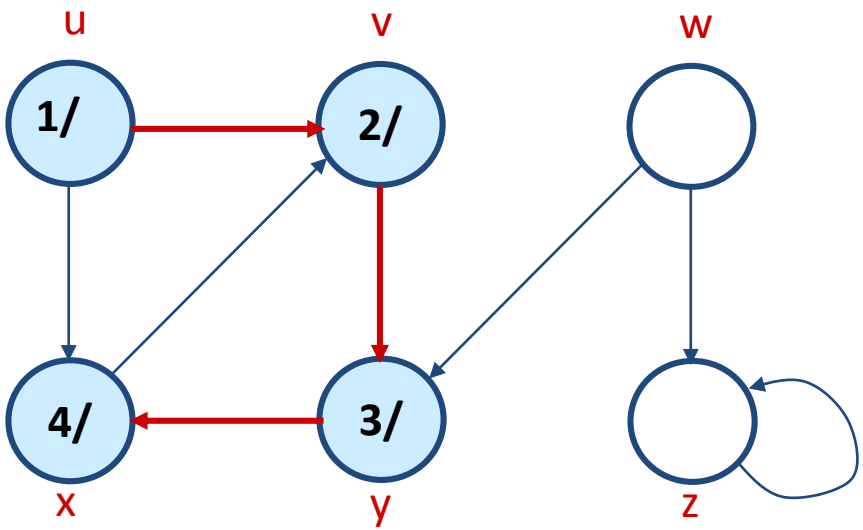
Example (DFS)



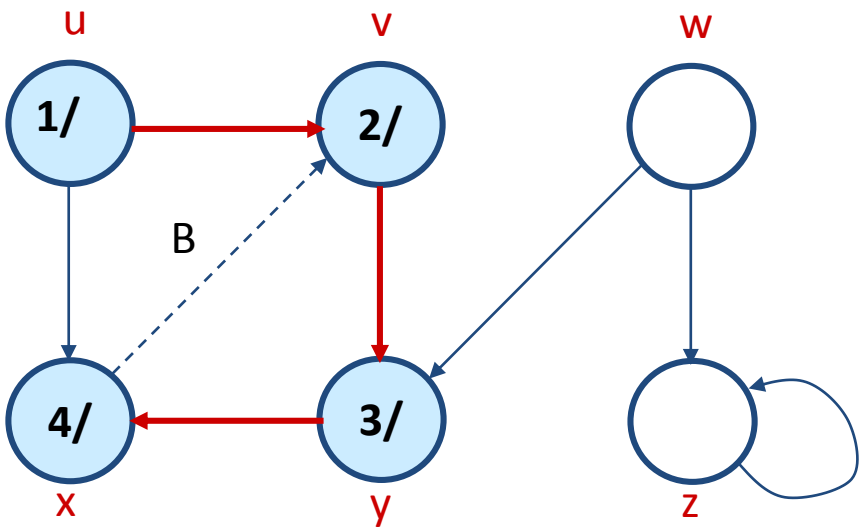
Example (DFS)



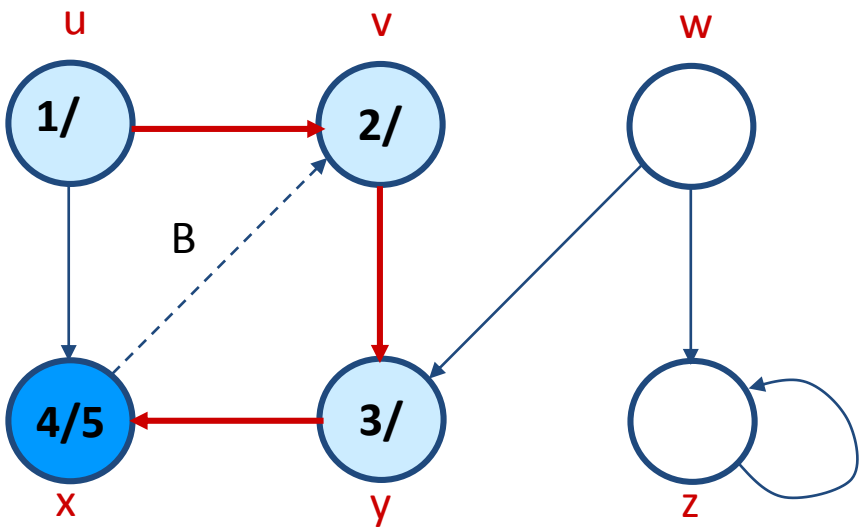
Example (DFS)



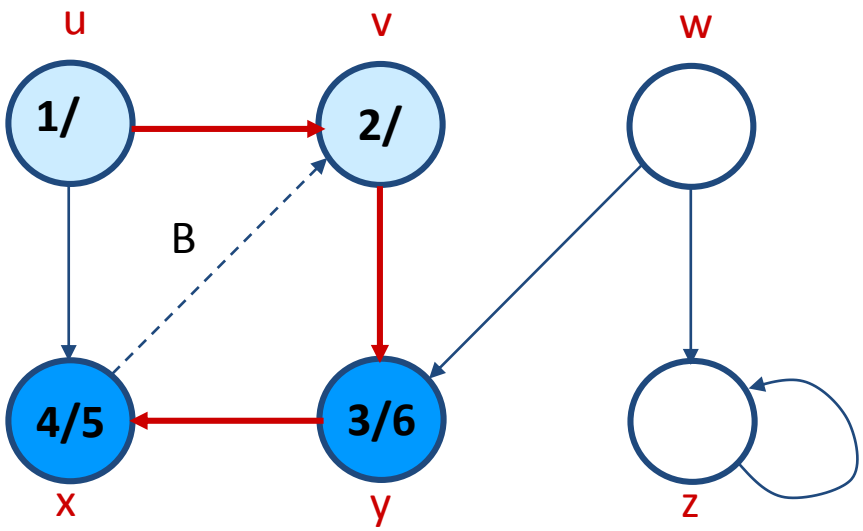
Example (DFS)



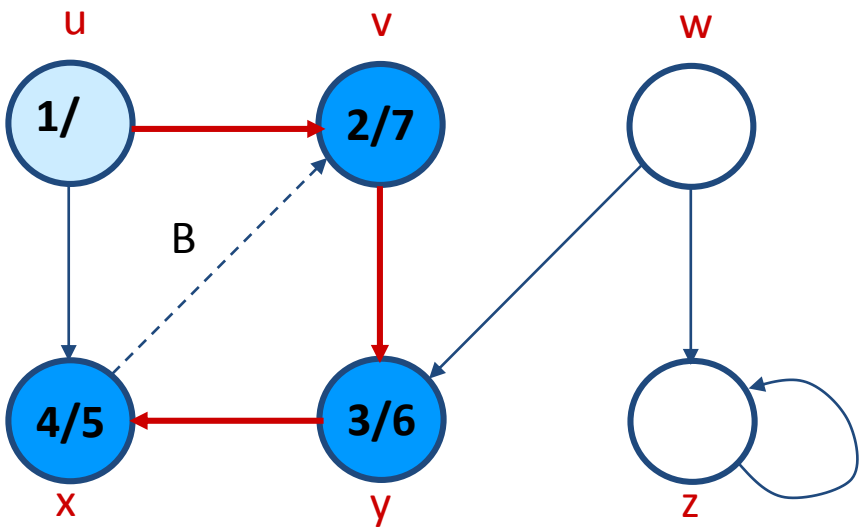
Example (DFS)



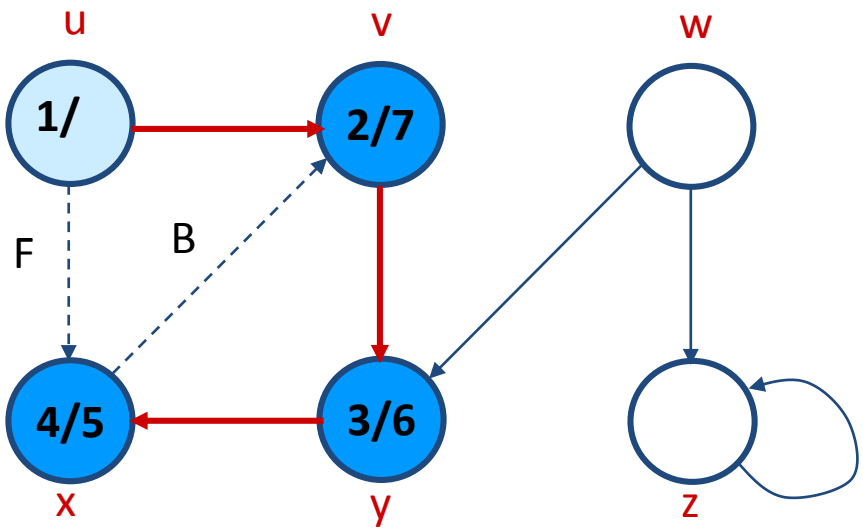
Example (DFS)



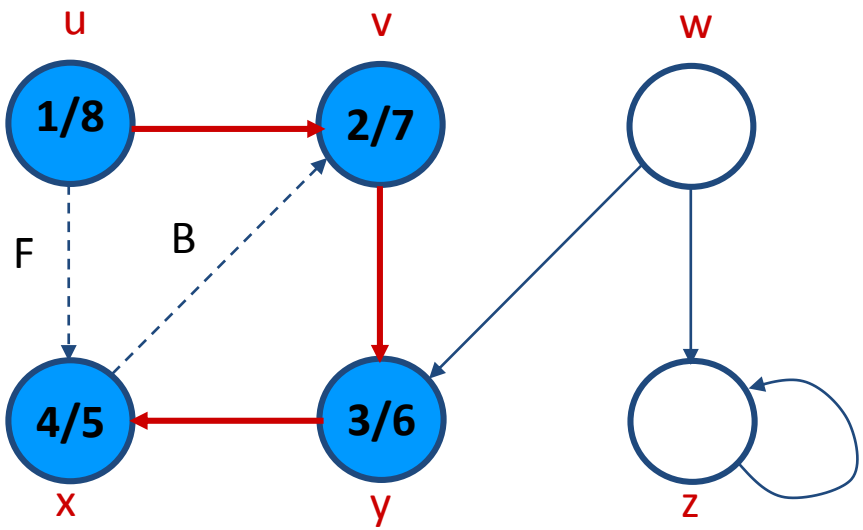
Example (DFS)



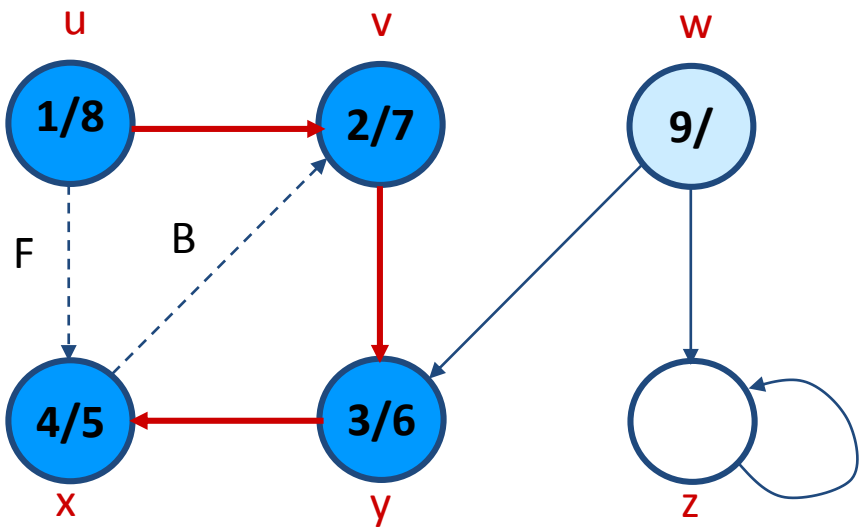
Example (DFS)



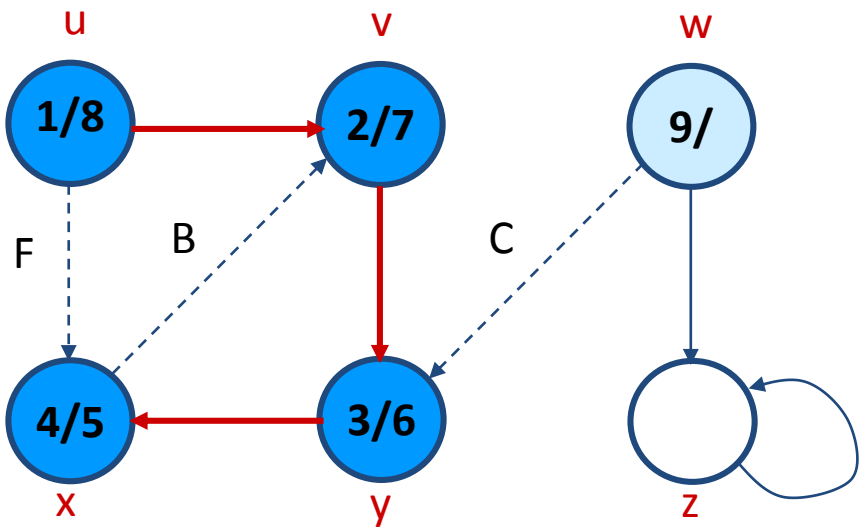
Example (DFS)



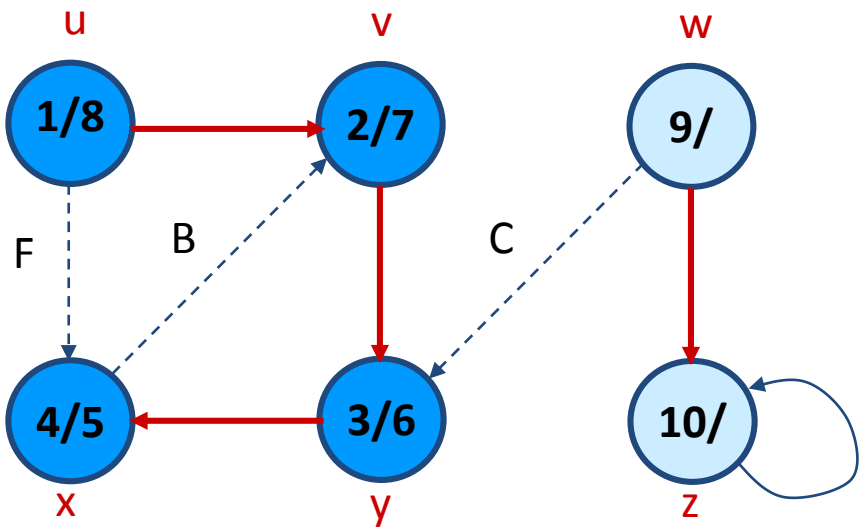
Example (DFS)



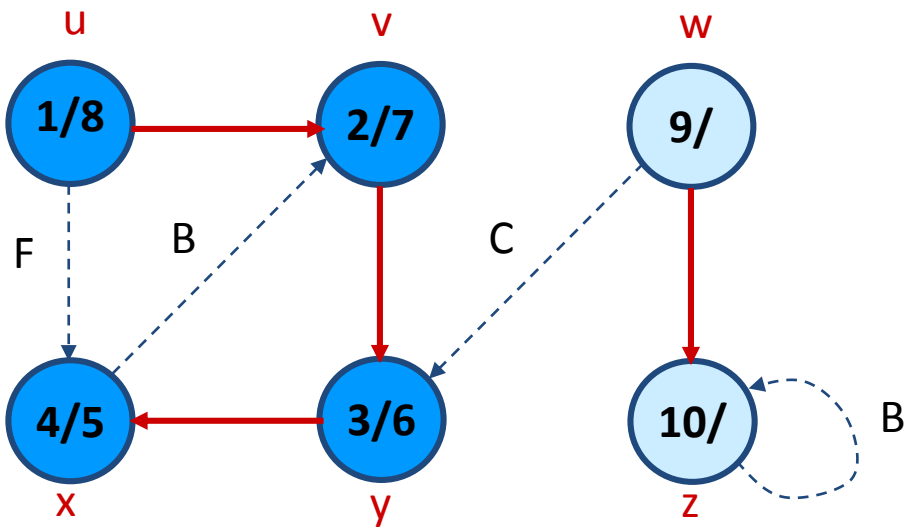
Example (DFS)



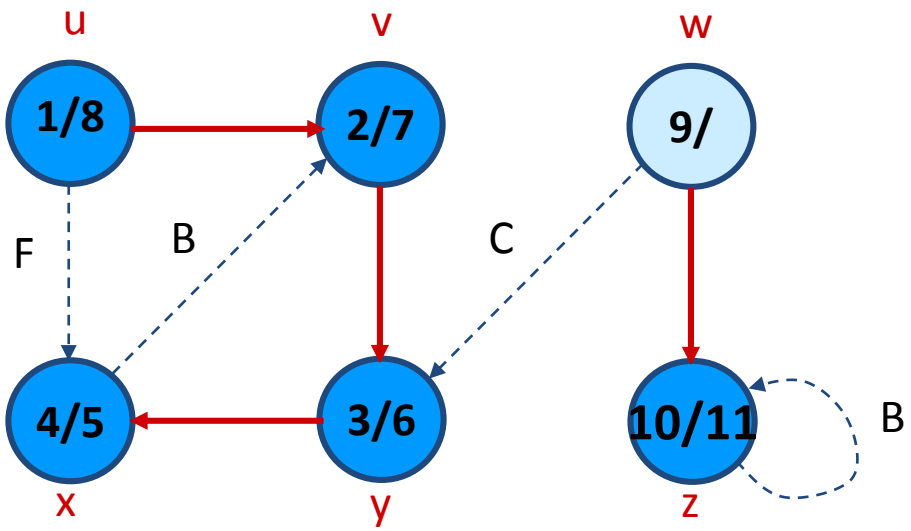
Example (DFS)



Example (DFS)



Example (DFS)



Example (DFS)