GRAPH BASIC

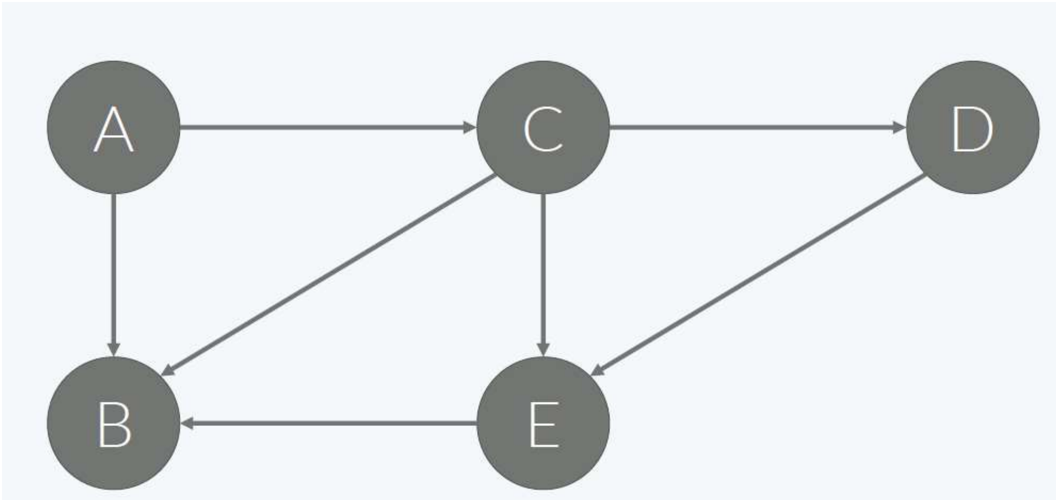
Path, Cycle

Path

- A→C→D→E→B
- A→B
- A→C→B
- A→C→E→B

Cycle

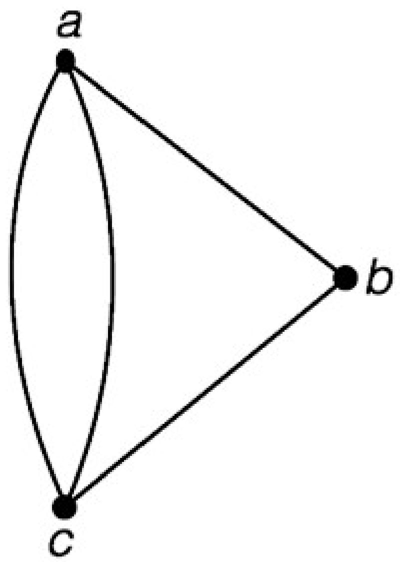
- A→C→B →A
- A→C→E→B→A
- A→C→D→E→B→A



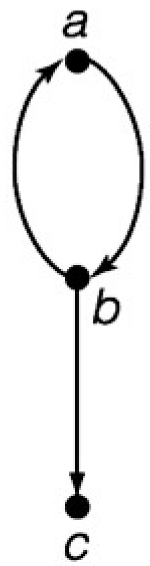
Directed, Undirected Graph

- A *graph* is a non-empty finite set *V* of elements called *vertices* together with a possibly empty set *E* of pairs of vertices called *edges.*

Undirected Graph
Bidirection Graph



directed Graph



Representation of graph

1.

G = (V, E)

V = {v1, v2, v3, v4, v5}

E = {e1, e2, e3, e4, e5, e6, e7, e8 }

= {(v1, v2), (v1, v3), (v1, v4), (v2, v3),

(v2, v5), (v3, v4), (v3, v5),(v4, v5) }

2. Multiple Edges

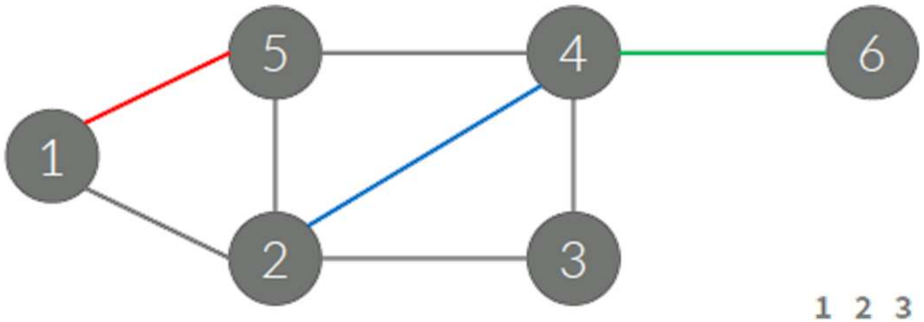
G = (V, E),

V = {1, 2, 3, 4, }

E = {(1, 2),(1, 3),(1, 4),(2, 3),(2, 3),(3, 4),(3, 4),(3, 4)}

Representation of graph(unweighted)

V = {1, 2, 3, 4, 5, 6}
E = {(1,2), (1,5), (2,5), (2,3), (3,4), (2,4), (4,5), (4,6)}

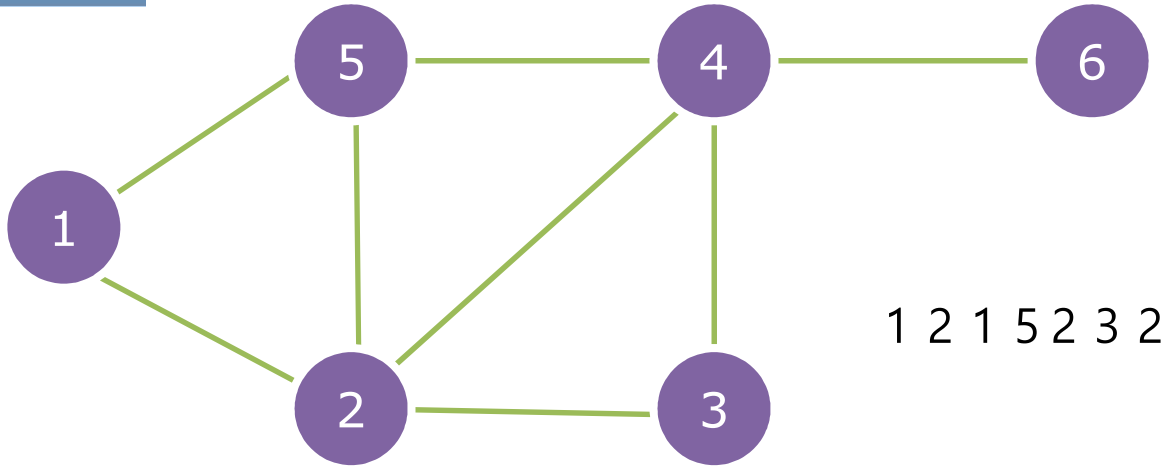


	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	1	1	0
3	0	1	0	1	0	0
4	0	1	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

```
6 8 //v= 6, E= 8
1 2
1 5
2 3
2 4
2 5
5 4
4 3
4 6
```

Representation of graph

6 8 //v= 6, E= 8



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

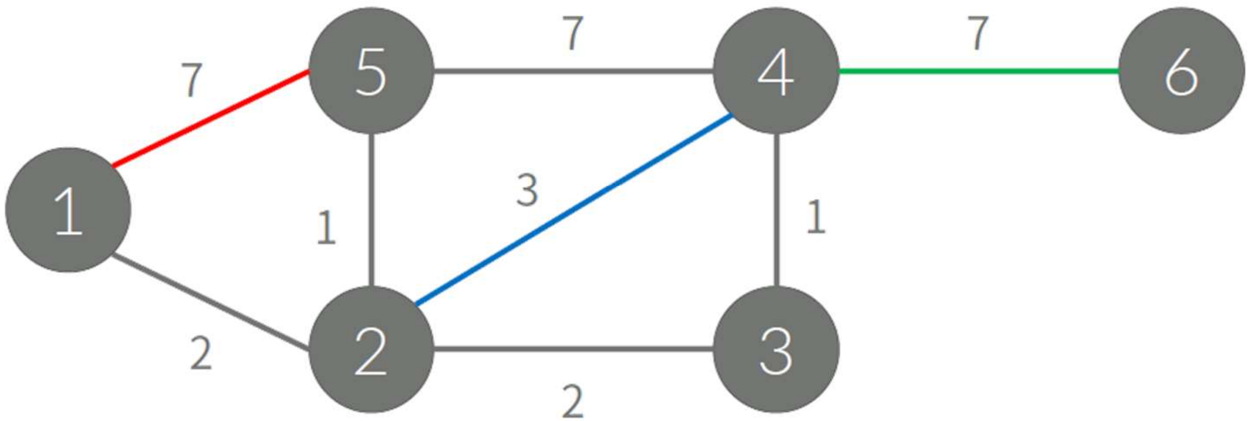
```
MyMap = [[0]*7 for i in range(7)]
Data =  list(map(int,input().split()))

for i in range(howmany):
    Start= Data[i*2]
    Stop = Data[i*2+1]
    MyMap[Start][Stop] = 1
    MyMap[Stop][Start] = 1
```

	0	1	2	3	4	5	6
0							
1			1			1	
2		1		1	1	1	
3			1		1		
4			1	1	1		1
5		1	1		1		
6					1		

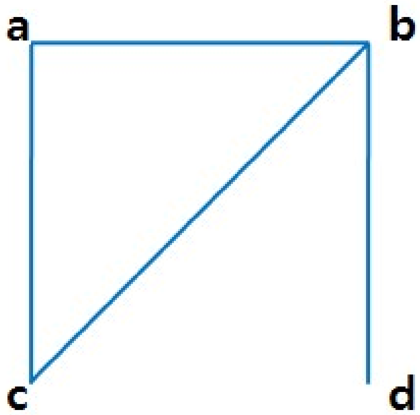
Representation of graph(weighted)

	1	2	3	4	5	6
1	0	2	0	0	7	0
2	2	0	2	3	1	0
3	0	2	0	1	0	0
4	0	3	1	0	7	7
5	7	1	0	7	0	0
6	0	0	0	7	0	0



Map[start][stop] = Map[stop][start] = cost;

DeGree

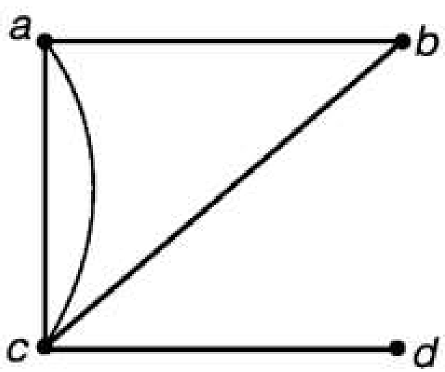


deg(a) =

deg(b) =

deg(c) =

deg(d) =



deg(a) =

deg(b) =

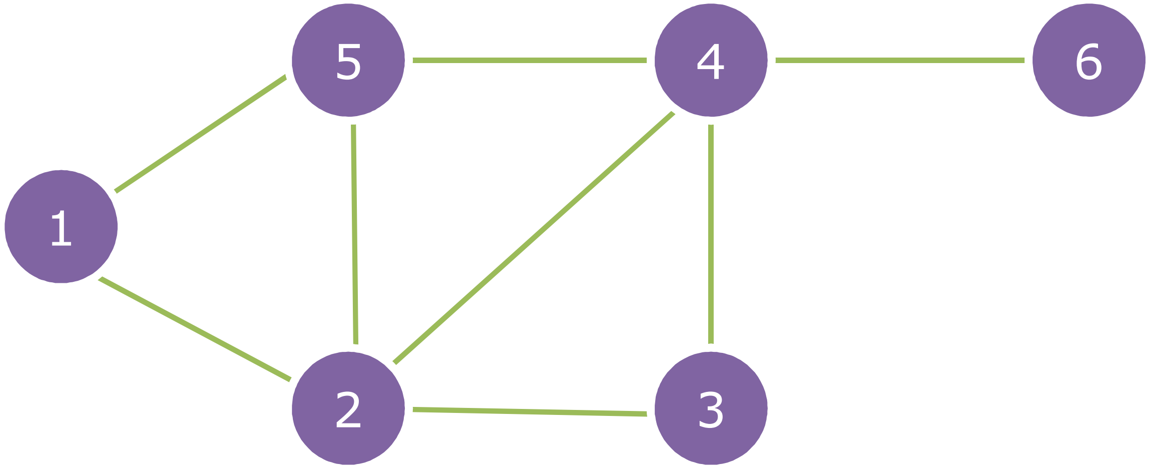
deg(c) =

deg(d) =

The sum of degrees =

The sum of edges =

Degree : Undirected Graph



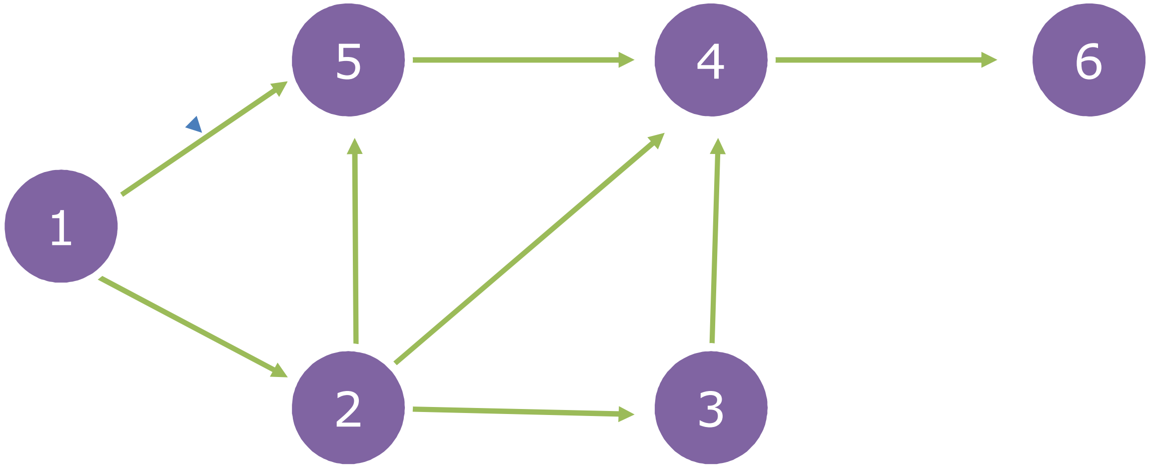
6 8 //v= 6, E= 8
1 2
1 5
2 3
2 4
2 5
5 4
4 3
4 6

Degree[Start] += 1
Degree[Stop]+=1

Degree

0	1	2	3	4	5	6
0	2	4	2	4	3	1

Degree : directed Graph



InDegree

0	1	2	3	4	5	6
0	0	1	1	3	2	1

OutDegree

0	1	2	3	4	5	6
0	2	3	1	1	1	0

Exercise 1
Minimum Sum

- n*n numbers are given. ($1 \leq n \leq 10$)
- Select one number in each row and one in each column that does not overlap.
- (i.e., we will take a total of n numbers, and each number is a value of less than 100).
- Find the minimum sum when you find the sum of n
- Input
  - n is entered in the first line. N integers are entered from the next line to the n+1 line.
- Output
  - Print out the minimum sum obtained.
- Input / Output

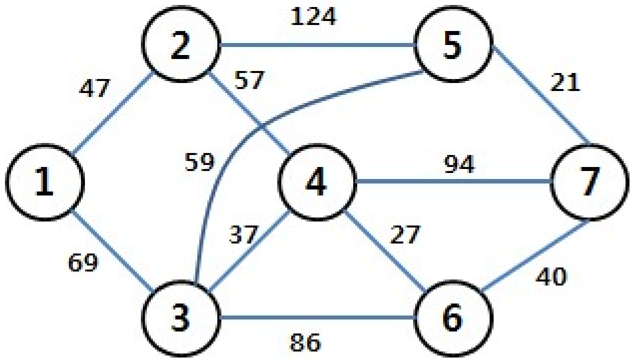
Input	Output
3 1 5 3 2 4 7 5 3 5	7

Exercise 2
On the way to meet the boyfriend

- Shalala wants to go to university A to meet her boyfriend. The number of vertex including departure and arrival is n. And the number of Edges linking the vertex is m. If Shalala's location is vertex1, and A University is vertex n, write a program that asks for the minimum cost of Shalala's visit to her boyfriend. ($n \leq 10$, $m \leq 30$)
- And the cost of going from one vertex to another vertex is an integer of not more than 200 and if there is a way from one vertex to another, the same method and cost can be reversed.

Exercise 2
On the way to meet the boyfriend

- The following graph shows an example (there may be multiple edges to Vertex a-> Vertex b, There may be an edge going back to the same vertex.



- Minimum Cost Path: 1→3→5→7,
- Minimum Cost: 69+59+21=149

Exercise 2
On the way to meet the boyfriend

- Input
  - In the first line, the number of vertices n and the number of edges m are entered, separated by spaces. The two vertices are numbered and weighted over the following line to m.
- Output
  - print out the minimum cost of getting to the university. If not, print out "-1".
- Input / Output

Input:

```
7 11
1 2 47
1 3 69
2 4 57
2 5 124
3 4 37
3 5 59
3 6 86
4 6 27
4 7 94
5 7 21
6 7 40
```

Output:

```
149
```