In [1]:
```
import networkx as nx
# The following line initializes two empty directed graph objects
G1=nx.DiGraph()
G2=nx.DiGraph()
# An empty undirected graph object can be initialized using the command
# G=nx.Graph()
```

In [2]:
```
G1.add_node(1)
G1.add_node(2)
G1.add_node(3)
G1.add_node(4)
G1.add_node(5)
G1.add_node(6)
G1.nodes()
```

Out[2]:
```
NodeView((1, 2, 3, 4, 5, 6))
```

In [3]:
```
list_nodes = [1, 2, 3, 4, 5, 6]
G2.add_nodes_from(list_nodes)
G2.nodes()
```

Out[3]:
```
NodeView((1, 2, 3, 4, 5, 6))
```

In [8]:
```
G1.add_edge(1, 2, weight = 2.0)
G1.add_edge(1,3,weight = 240)
#G1.edge[1][3]['weight'] = 4.0
G1.add_edge(2, 3, weight = 1.0)
G1.add_edge(2, 4, weight = 4.0)
G1.add_edge(2, 5, weight = 2.0)
G1.add_edge(3, 5,weight=3)
#G1.edge[3][5]['weight'] = 3.0
G1.add_edge(4, 6, weight = 2.0)
G1.add_edge(5, 4, weight = 3.0)
G1.add_edge(5, 6, weight = 2.0)
G1.edges()
```

Out[8]:
```
OutEdgeView([(1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (4, 6), (5, 4),
(5, 6)])
```

In [6]:
```
list_arcs = [(1,2,2.0) , (1,3,4.0) , (2,3,1.0) , (2,4,4.0) , (2,5,2.0) , (3,5,3.(
G2.add_weighted_edges_from(list_arcs)
G2.edges()
```

Out[6]:
```
OutEdgeView([(1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (4, 6), (5, 4),
(5, 6)])
```

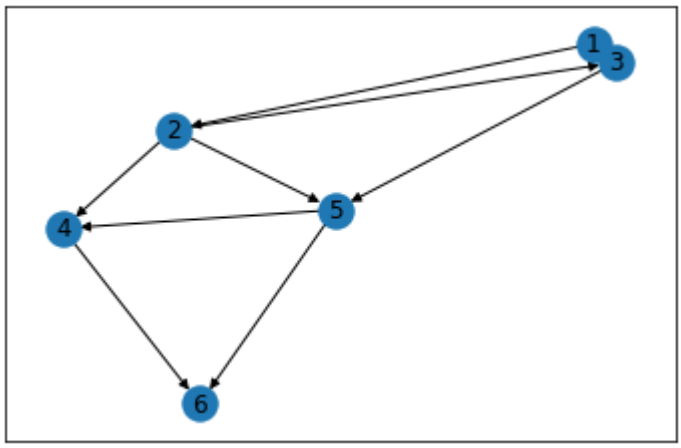
In [9]:
```
sp = nx.dijkstra_path(G1,source = 1, target = 6)
print(sp)
```

```
[1, 2, 5, 6]
```

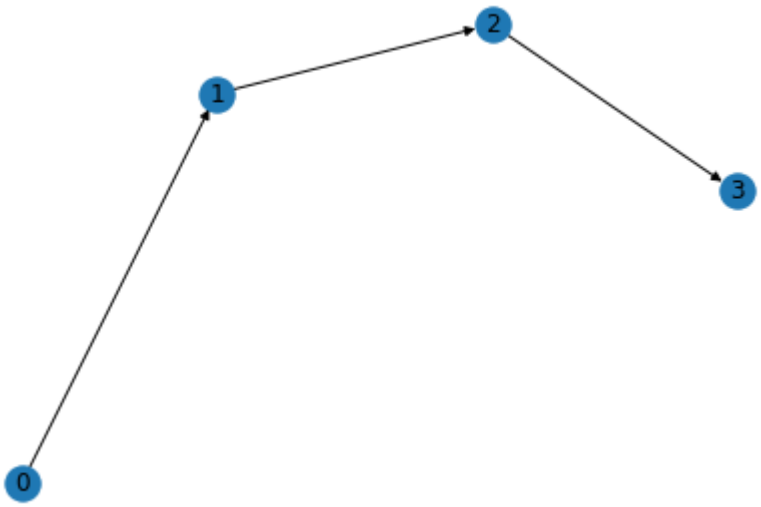
In [10]:
```
print(nx.shortest_path(G1,source = 1, target = 6))
```

```
[1, 2, 4, 6]
```

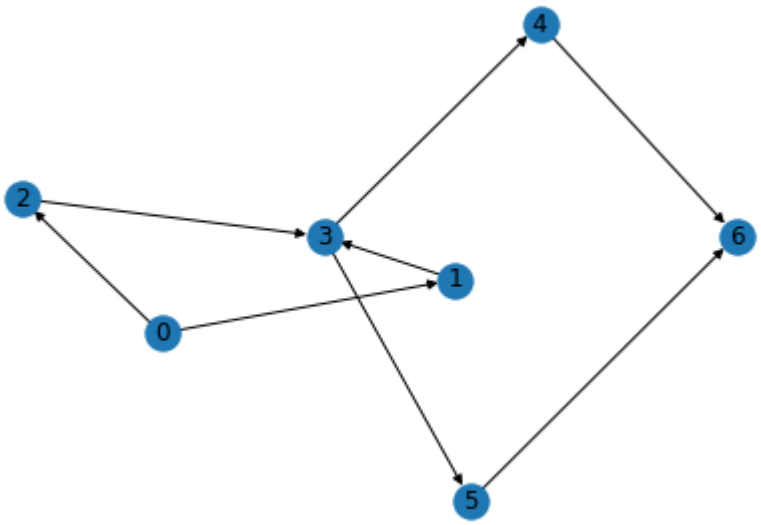
In [21]:
```
nx.draw_networkx(G1)
```



In [185]:
```
G = nx.DiGraph()

G.add_nodes_from([0, 1, 2, 3])

G.add_weighted_edges_from([
    (0, 1, 4),
    (1, 2, 3),
    (2, 3, 6),

], weight='capacity')

plt.figure()
nx.draw(G, with_labels=True)
plt.show()
```

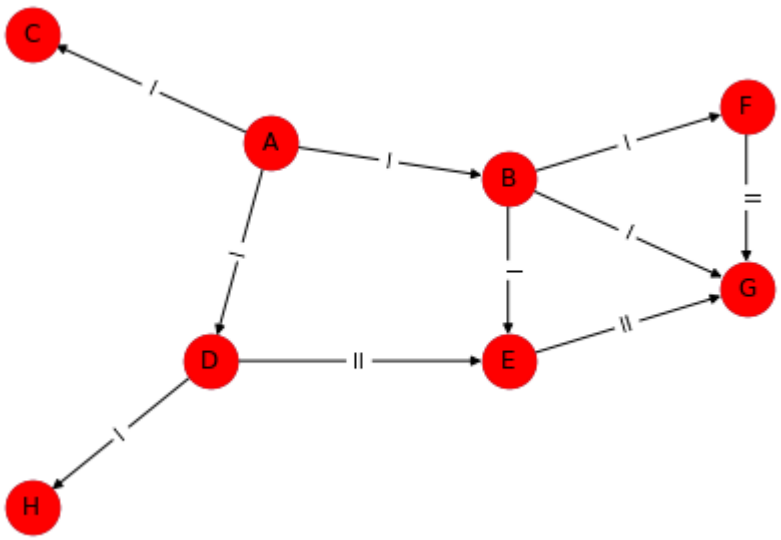


In [186]:
```
G = nx.DiGraph()

G.add_nodes_from([0, 1, 2, 3, 4, 5, 6])
G.add_edges_from([
    (0, 1),
    (0, 2),
    (1, 3),
    (2, 3),
    (3, 4),
    (3, 5),
    (4, 6),
    (5, 6),
])

plt.figure()
nx.draw(G, with_labels=True)
plt.show()
```

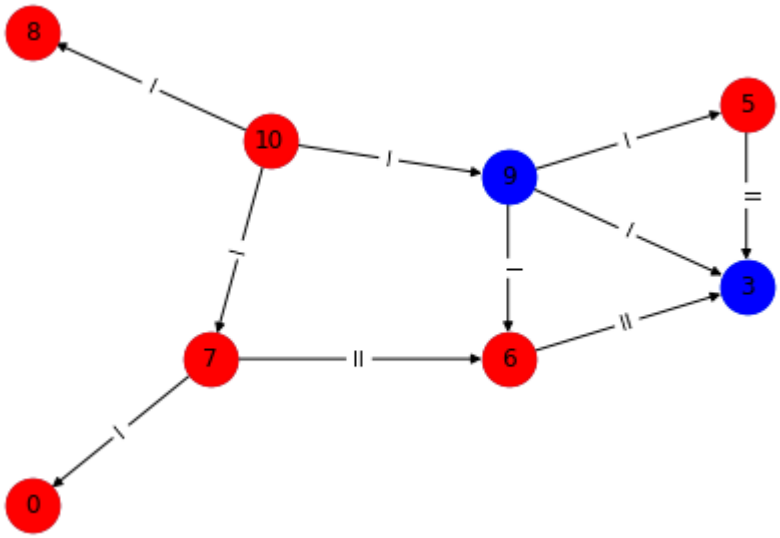


In [187]:
```
import matplotlib.pyplot as plt
g2=nx.Graph()
g2.add_edge('A','B',weight=0.9)
g2.add_node('C')
#nx.draw(g2)

pos = nx.spring_layout(g2)  # compute graph layout
nx.draw(g2, pos, node_size=700)  # draw nodes and edges
nx.draw_networkx_labels(g2, pos)

labels = nx.get_edge_attributes(g2, 'weight')
nx.draw_networkx_edge_labels(g2, pos, edge_labels=labels)
#nx.draw_networkx_nodes(g2, pos,node_labels=labels2)
#nx.draw_networkx_edge_labels(g2, pos,edge_labels=labels2)
#nx.draw_networkx_labels(G, pos)
#nx.draw_networkx_edges(G, pos, width=1.0)

plt.show(g2)
```

In [213]:
```
import networkx as nx

G=nx.DiGraph()
# Add nodes by specifying their positions
G.add_node('10', pos=(2, 10), label='A',color='r')
G.add_node('9',  pos=(4, 9), label='B',color='r')
G.add_node('8',  pos=(0, 13), label='C',color='r')
G.add_node('7',  pos=(1.5, 4), label='D',color='r')
G.add_node('6',  pos=(4, 4), label='E',color='r')
G.add_node('5',  pos=(6, 11), label='F',color='r')
G.add_node('3',  pos=(6, 6), label='G',color='r')
G.add_node('0',  pos=(0, 0), label='H',color='r')
# Add edges by defining weight and label
G.add_edge('10','9',weight=1, label='I')
G.add_edge('10','8',weight=1, label='I')
G.add_edge('10','7',weight=1, label='I')
G.add_edge('9','3', weight=1, label='I')
G.add_edge('9','6',weight=1, label='I')
G.add_edge('9','5',weight=1, label='I')
G.add_edge('7','0',weight=1, label='I')
G.add_edge('7','6',weight=0, label='II')
G.add_edge('6','3',weight=0, label='II')
G.add_edge('5','3',weight=0, label='II')

node_position=nx.get_node_attributes(G,'pos')
node_label=nx.get_node_attributes(G,'label')
edge_label=nx.get_edge_attributes(G,'label')
node_color=nx.get_node_attributes(G,'color')

#print("Node label=",node_label)
#print("Pos label=",node_position)
#print("Edge label=",edge_label)

#nx.draw_networkx_nodes(G,pos,node_color='r')

nx.draw(G,node_position,node_size=700,with_labels=False)  # draw nodes and edges
nx.draw_networkx_labels(G, node_position,node_label)
nx.draw_networkx_edge_labels(G,node_position,edge_label)
nx.draw_networkx_nodes(G, node_position,node_color='r', node_size=700)
plt.show(G)
```


C
A
B
F
G
D
E
H

In [258]:
```
import networkx as nx

G=nx.DiGraph()
# Add nodes by specifying their positions
G.add_node('10', pos=(2, 10), label='10',color='r')
G.add_node('9',  pos=(4, 9), label='9',color='b')
G.add_node('8',  pos=(0, 13), label='8',color='r')
G.add_node('7',  pos=(1.5, 4), label='7',color='r')
G.add_node('6',  pos=(4, 4), label='6',color='r')
G.add_node('5',  pos=(6, 11), label='5',color='r')
G.add_node('3',  pos=(6, 6), label='3',color='b')
G.add_node('0',  pos=(0, 0), label='0',color='r')
# Add edges by defining weight and label
G.add_edge('10','9',weight=1, label='I')
G.add_edge('10','8',weight=1, label='I')
G.add_edge('10','7',weight=1, label='I')
G.add_edge('9','3', weight=1, label='I')
G.add_edge('9','6',weight=1, label='I')
G.add_edge('9','5',weight=1, label='I')
G.add_edge('7','0',weight=1, label='I')
G.add_edge('7','6',weight=0, label='II')
G.add_edge('6','3',weight=0, label='II')
G.add_edge('5','3',weight=0, label='II')

node_position=nx.get_node_attributes(G,'pos')
node_label=nx.get_node_attributes(G,'label')
edge_label=nx.get_edge_attributes(G,'label')
node_color=nx.get_node_attributes(G,'color')

#print("Node label=",node_label)
#print("Pos label=",node_position)
#print("Edge label=",edge_label)

nx.draw(G,node_position,node_size=700,with_labels=False)  # draw nodes and edges
nx.draw_networkx_labels(G, node_position,node_label)
nx.draw_networkx_edge_labels(G,node_position,edge_label)

blue_nodes=[n for n,d in G.nodes(data=True) if d['color']=='b']
red_nodes=[n for n,d in G.nodes(data=True) if d['color']=='r']

nx.draw_networkx_nodes(G,node_position,nodelist=blue_nodes,node_color='b', node_s
nx.draw_networkx_nodes(G,node_position,nodelist=red_nodes,node_color='r', node_si
#nx.draw_networkx_nodes(G,node_position,nodelist=red_nodes,node_color='r')

plt.show(G)
print(G.nodes)
```




```
['10', '9', '8', '7', '6', '5', '3', '0']
```

Find th path

In [205]:
```
nx.shortest_path(G, '10', '3')
```

Out[205]:
```
['10', '9', '3']
```

In [206]:
```
 list(nx.all_shortest_paths(G, '10','3'))
```

Out[206]:
```
[['10', '9', '3']]
```

In [207]:
```
print([p for p in nx.all_shortest_paths(G,source='10',target='3')])
```

```
[['10', '9', '3']]
```

Find all paths

In [212]:
```
p = []
for path in nx.all_simple_paths(G, source='10', target='3'):
    p.append(path)

p
```

Out[212]:
```
[['10', '9', '3'],
 ['10', '9', '6', '3'],
 ['10', '9', '5', '3'],
 ['10', '7', '6', '3']]
```

Neighbors

In [215]:
```
[n for n in G.neighbors('10')]
```

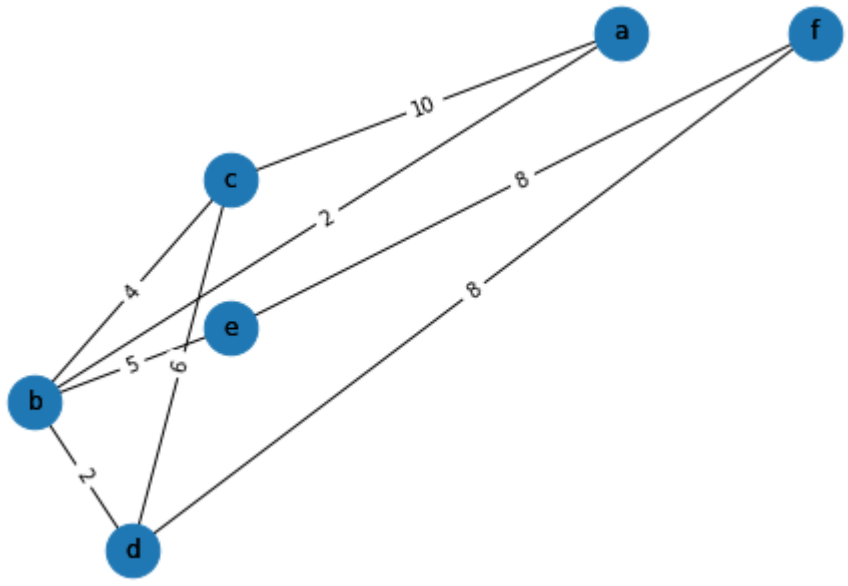
Out[215]:
```
['9', '8', '7']
```

In [ ]:
```

```

In [216]:
```
def k_shortest_path(G, source, target, k):
    def path_cost(G, path):
        return sum([G[path[i]][path[i+1]]['weight'] for i in range(len(path)-1)]
    return sorted([(path_cost(G,p), p) for p in nx.shortest_simple_paths(G, sour
```

In [262]:
```
import networkx as nx

G = nx.Graph()

G.add_node('a',pos=(8, 10))
G.add_node('b',pos=(2, 5))
G.add_node('c',pos=(4, 8))
G.add_node('d',pos=(3, 3))
G.add_node('e',pos=(4, 6))
G.add_node('f',pos=(10, 10))


G.add_edge('a', 'b', weight=2)
G.add_edge('b', 'c', weight=4)
G.add_edge('a', 'c', weight=10)
G.add_edge('c', 'd', weight=6)
G.add_edge('b', 'd', weight=2)
G.add_edge('b', 'e', weight=5)
G.add_edge('e', 'f', weight=8)
G.add_edge('d', 'f', weight=8)
edge_weight=nx.get_edge_attributes(G,'weight')
nod_pos=nx.get_node_attributes(G,'pos')

nx.draw(G,nod_pos,node_size=700,with_labels=True)
nx.draw_networkx_labels(G, nod_pos)
nx.draw_networkx_edge_labels(G,nod_pos,edge_weight)
print(edge_weight)
```

```
{('a', 'b'): 2, ('a', 'c'): 10, ('b', 'c'): 4, ('b', 'd'): 2, ('b', 'e'): 5,
('c', 'd'): 6, ('d', 'f'): 8, ('e', 'f'): 8}
```



In [266]:
```
k_shortest_path(G, 'a', 'f', 4)
#nx.k_shortest_path(G, 'a', 'f')
```

Out[266]:
```
(12, ['a', 'b', 'd', 'f'])
```

In [ ]: