Shortest path in weighted graphs

find the sp between every edge:

> floyd-warshal algorithm (dynamic programming)

find the sp between a source and other nodes:

> dijkstra algorithm (greedy)

> bellman-ford algorithm (dynamic programming)

> sssp (fast dijkstra in DAG)

floyd-warshal implementation

Timecomplexity => O(n^3)

In [5]:
```
def floyd(adj_matrix, nodes):
    dp = [[0 for x in range(nodes)] for y in range (nodes)]
    for i in range (nodes):
        for j in range(nodes):
            dp[i][j] = adj_matrix[i][j]

    for k in range (nodes):
        for i in range (nodes):
            for j in range (nodes):
                dp[i][j] = min (dp[i][j], dp[i][k] + dp[k][j])

    return dp
```

bellman-ford implementation

Timecomplexity => O(N*E)

In [14]:
```
def bellman_ford (graph, nodes, source):
    infi = 1000009
    dp = [infi for x in range(nodes)]
    dp[source] = 0

    for i in range (nodes):
        for u, v, w in graph:
            if dp[u] != infi and dp[u] + w < dp[v]:
                        dp[v] = dp[u] + w

    return dp
```

In [15]:
```
def input_graph():
    nodes = int(input())
    edges = int(input())
    graph = []
    for i in range (edges):
        u, v, w = map(int, input().split())
        graph.append((u, v, w))
    return bellman_ford(graph, nodes, 0)
```

In [16]:
```
input_graph()
```

```
6
7
0 1 2
0 2 3
1 2 1
2 3 1
2 4 1
3 4 2
2 5 100
```

Out[16]:
```
[0, 2, 3, 4, 4, 103]
```