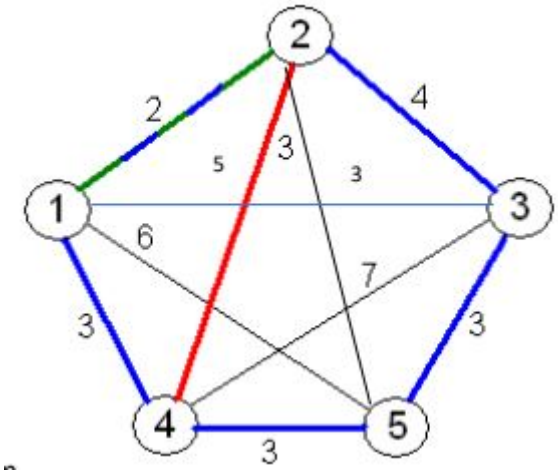
TSP Using Hill Climbing Algorithm



In [1]:
```
tsp = [
    [0,2,3,3,6],
    [2,0,4,5,3],
    [3,4,0,7,3],
    [3,5,7,0,3],
    [6,3,3,3,0]
]
```

To calculate path length/cost

In [2]:
```
def pathCost(path):
    cost = 0
    for i in range(len(path)):
        cost += tsp[path[i - 1]][path[i]]
    return cost
```

To get Neighbours

- done by swapping positions

In [3]:
```
def getNeighbours(route):
    neighbours = []
    for i in range(len(route)):
        for j in range(i + 1, len(route)):
            temp = route.copy()
            temp[i] = route[j]
            temp[j] = route[i]
            neighbours.append(temp)
    return neighbours
```

To find best neighbour

In [4]:
```
def getBestNeighbour(neighbours):
    bestNeighbour = neighbours[0]
    minCost = pathCost(bestNeighbour)
    for neighbour in neighbours:
        cost = pathCost(neighbour)
        if cost < minCost:
            minCost = cost
            bestNeighbour = neighbour
    return bestNeighbour, minCost
```

To determine where to start from

In [5]:
```
import random
n = int(input("Enter choice for initial route: \n1)Custom\n2)Random\n:>"))
initial_route = []
if(n==1):
    for i in range(len(tsp)):
        initial_route.append(int(input(" "))-1)
else:
    temp = [i for i in range(len(tsp))]
    for i in range(len(tsp)):
        randomCity = temp[random.randint(0, len(temp)-1)]
        initial_route.append(randomCity)
        temp.remove(randomCity)
```

```
Enter choice for initial route:
1)Custom
2)Random
:>2
```

In [6]:
```
best_route = initial_route
best_cost = pathCost(best_route)
print("Initial path : {0} and it's cost: {1}".format([i+1 for i in best_route],best_cost))
```

```
Initial path : [5, 1, 4, 2, 3] and it's cost: 21
```

Hill climbing alog

In [7]:
```
neighbours = getNeighbours(best_route)
bestNeighbour, bestNeighbourCost = getBestNeighbour(neighbours)
while bestNeighbourCost < best_cost:
    best_route = bestNeighbour
    best_cost = bestNeighbourCost
    neighbours = getNeighbours(best_route)
    bestNeighbour, bestNeighbourCost = getBestNeighbour(neighbours)
print("Best local route: {0} and it's cost: {1}".format([i+1 for i in best_route], best_cost))
```

```
Best local route: [2, 1, 4, 5, 3] and it's cost: 15
```