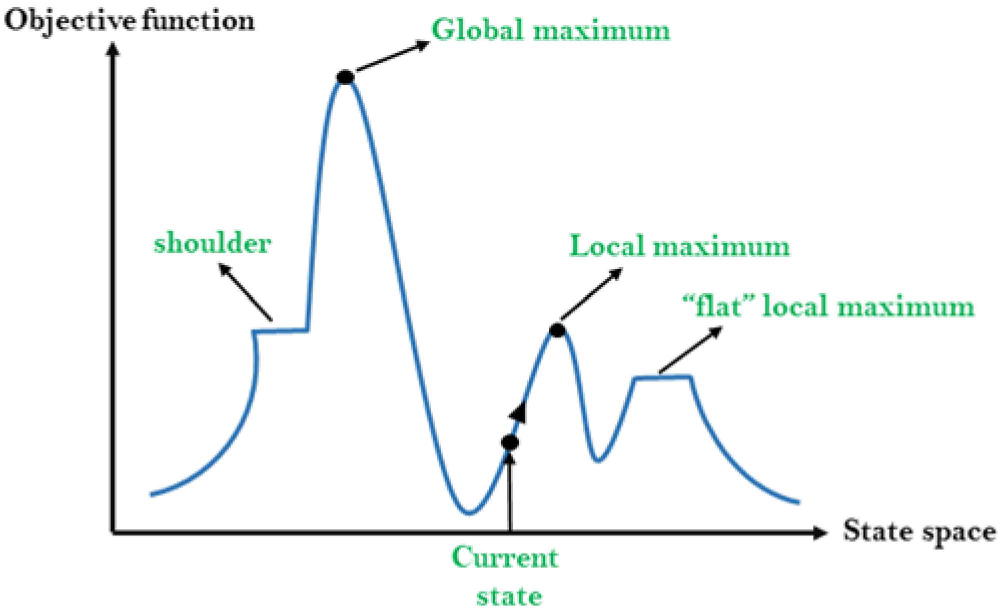
Hill-Climbing

- Problem :

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal



```
import random

def randomSolution(tsp):
    cities = list(range(len(tsp)))
    # 0 , 1, 2, 3
    print("cities",cities)
    solution = []

    for i in range(len(tsp)):
        randomCity = cities[random.randint(0, len(cities) - 1)]
        # print(cities[random.randint(0, len(cities) - 1)])
        # 0 1 3 2 ,  0 3 2 1
        solution.append(randomCity)
        cities.remove(randomCity)
    # print("solution",solution)
    return solution

def routeLength(tsp, solution):
    routeLength = 0
    for i in range(len(solution)):
        print("query",tsp[solution[i - 1]][solution[i]])
        routeLength += tsp[solution[i - 1]][solution[i]] #[0 , 300 , 200 , 500]
        # print("routeLength",routeLength)
    # print("route",routeLength)
```

```
        # print("route",routeLength)
        return routeLength

    def getNeighbours(solution):
        neighbours = []

        return neighbours

    def getBestNeighbour(tsp, neighbours):

        return bestNeighbour, bestRouteLength

    def hillClimbing(tsp):
        currentSolution = randomSolution(tsp)
        print("currentSolution",currentSolution)
        currentRouteLength = routeLength(tsp, currentSolution)
        print("currentRouteLength",currentRouteLength)
        neighbours = getNeighbours(currentSolution) # list
        bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

        while bestNeighbourRouteLength < currentRouteLength:
            currentSolution = bestNeighbour
            currentRouteLength = bestNeighbourRouteLength
            neighbours = getNeighbours(currentSolution)
            bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

        return currentSolution, currentRouteLength
    def problemGenerator(nCities):
        tsp = []
        for i in range(nCities):
            distances = []
            for j in range(nCities):
                if j == i:
                    distances.append(0)
                elif j < i:
                    distances.append(tsp[j][i])
                else:
                    distances.append(random.randint(10, 100))
            tsp.append(distances)
        print(tsp)
        return tsp
    def main():
        tsp = [
            [0, 400, 500, 300],
            [400, 0, 300, 500],
            [500, 300, 0, 400],
            [300, 500, 400, 0]
        ]

        print(hillClimbing(tsp))
        # tsp = problemGenerator(10)
        # for i in range(10):
```

```
    #     print(hillClimbing(tsp))
if __name__ == "__main__":
    main()
```

```
def problemGenerator(nCities):
    tsp = []
    for i in range(nCities):
        distances = []
        for j in range(nCities):
            if j == i:
                distances.append(0)
            elif j < i:
                distances.append(tsp[j][i])
            else:
                distances.append(random.randint(10, 1000))
        tsp.append(distances)
    return tsp
def main():
    tsp = problemGenerator(10)
    for i in range(10):
        print(hillClimbing(tsp))
```

Task

Perform the following functions to complete Travelling sales man problem.

```
def getNeighbours(solution):
    neighbours = []

    return neighbours

def getBestNeighbour(tsp, neighbours):

    return bestNeighbour, bestRouteLength
```

● ✕