Optimierung B

Aufgabe 1

Function processPrim

Use normal Prim algorithms. Use dictionary to save the edge

```
------------aufgabe 1 -----------------
minimale spannebaum cost: 238.0
time: 0.00302791595459s
```

Aufgabe 2

Function processDijkstra

Use normal Dijkstra algorithms without acceleration

```
------------aufgabe 2 -----------------
cost of Dijkstra from 2 to 50 is 18.0
['36', '2']
time: 0.000775098800659s
```

Aufgabe 3

Function processHeuristric

we are given original graph and a point set (unvisited set)

visited set is empty.

Use the Dijkstra as the distance between any two points in the set. The algorithms as following:

1. Start with arbitrary point, add this point to visited set (like in Prim),
2. Use Dijkstra to calculate the distance between any pair of visited set and unvisited set. The idea is just like in Prim, change the following two things:
    a. Any two points in visited set and unvisited set, respectively, can be reachable by each other
    b. The distance (heuristic) of two points is the shortest path distance calculate by Dijkstra.
3. Calculate the nearest point to any element of the visited set
4. Add this point to visited set and add the cost, and remove this point from unvisited set. Add all the point through the shortest path to new graph, which means that the points the

5. Repeat 2-4 until the unvisited set tern to empty.

```
print('------------aufgabe 3 ------------------')
print('#############for algorithm as prim############')
nodeSet = graph.nodes
graph.ProcessHeuristic(nodeSet)


print('#############for algorithm as Dijkstra#######')
nodeSet = ['2','50']
graph.ProcessHeuristic(nodeSet)
```

```
------------aufgabe 1 -----------------
minimale spannebaum cost: 238.0
time: 0.00302791595459s
------------aufgabe 2 -----------------
cost of Dijkstra from 2 to 50 is 18.0
['36', '2']
time: 0.000775098800659s
------------aufgabe 3 -----------------
#############for algorithm as prim#############
cost: 238.0
#############for algorithm as Dijkstra#######
cost: 18.0
```

It's very clear that when we input all the points as our unvisited set, the result will be the same as 'minimale spannebaum' calculated before. And if we input two points here, the algorithms will back to Dijkstra algorithms.

Aufgabe 4

```
------------aufgabe 4 a-----------------
minimale spannebaum cost: 238.0
time: 0.0022120475769s
minimale spannebaum cost: 337.0
time: 0.0147788524628s
minimale spannebaum cost: 255.0
time: 0.255584955215s
minimale spannebaum cost: 6481.0
time: 0.133061885834s
minimale spannebaum cost: 8763.0
time: 1.52745389938s
------------aufgabe 4 b-----------------
cost: 85.0
------------aufgabe 4 c-----------------
cost: 223.0
time: 60.4063251019s
```