Full desk checking for Dijkstra(graph,origin) function:

While nodes = true?	**Min_ node**	**For node in nodes**	**Visited**	**Path**	**Node in visited?**	**Min_node is none?**	**Visited[node] < visited[min_node]?**	**Nodes.remove(min_node)**	**Nodes after**	**Current_weight = visited[min_node]**	**For edge in graph.edges[min_node]**	**Weight = current_weight + graph.distaces[(min_node, edge)]. Error?**	**Edge not in visited or weight < visited[edge]?**	**Visited now**	**Path now**
Yes, nodes = {'F', 'A', 'G', 'D', 'B', 'E', 'C'}	None	Node = 'F'	{'A':0}	{}	No, proceed to new loop										
	None	Node = 'A'	{'A':0}	{}	Yes	Yes, min_node = 'A'	Visited['A'] < visited ['A'] = 0 < 0? no								
	A	Node = 'G'	{'A':0}	{}	No, proceed to new loop										
	A	Node = 'D'	{'A':0}	{}	No, proceed to new loop										
	A	Node = 'B'	{'A':0}	{}	No, proceed to new loop										
	A	Node = 'E'	{'A':0}	{}	No, proceed to new loop										
	A	Node = 'C'	{'A':0}	{}	No, proceed to new loop			Nodes.remove('A')	nodes = {'F', 'G', 'D', 'B', 'E', 'C'}	Current_weight = 0	Graph.edges['A'] ='B','C' Edge = 'B'	Weight = 0 + graph.distances['A','B'] = 0 + 10 = 10	Yes. Edge no in visited Visited['B'] = 10 Path['B] = 'A'	{'A': 0, 'B': 10}	{'B':'A'}
											Edge = 'C'	Weight = 0 + graph.distances['A','C'] = 0 + 20 =20	Yes, edge not in visited Visited['C'] = 20 Path['C'] = 'A'	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}
Yes, nodes = {'F', 'G', 'D', 'B', 'E', 'C'}	None	Node = 'F'	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	No, proceed to new loop										
	None	Node = 'G'	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	No, proceed to new loop										
	None	Node = 'D'	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	No, proceed to new loop										

	None	Node = ‘B’	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	Yes	Yes, Min_node = ‘B’	Visited[‘B’] < visited [‘B’] = 10 < 10? no								
	B	Node = ‘E’	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	No, proceed to new loop										
	B	Node = ‘C’	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}	Yes	No	Visited[‘C’] < visited[‘B’] = 20 < 10? no	Nodes.remove(‘B’)	nodes = {'F', 'G','D', 'E', 'C'}	Current_weight = 10	Graph.edges[‘B’] =‘A’,‘D’,‘E’ Edge = ‘A’	Weight = 10 + graph.distances[‘B’,‘A’] = error	No. A is in visited	{'A': 0, 'B': 10, 'C': 20}	{'B': 'A', 'C': 'A'}
											Edge = ‘D’	Weight = 10 + graph.distances[‘B’,‘D’] = 10 + 15 = 25	Yes, edge not in visited Visited[‘D’] = 25 Path[‘D’] = ‘B’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25}	{'B': 'A', 'C': 'A', ‘D’:‘B’}
											Edge = ‘E’	Weight = 10 + Graph.distances[‘B’,‘E’] = 10 + 50 = 60	Yes, edge not in visited Visited[‘E’]= 60 Path[‘E’] = ‘B’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}
Yes, nodes = {'F', 'G','D', 'E',‘C’}	None	Node = ‘F’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	No, proceed to new loop										
	None	Node = ‘G’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	No, proceed to new loop										
	None	Node = ‘D’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	Yes	Yes, min_node = ‘D’	Visited[‘D’] < visited [‘D’] = 25 < 25? no								
	D	Node = ‘E’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	Yes	no	Visited[‘E’] < visited[‘D’] = 60 < 25? no								
	D	Node = ‘C’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	Yes	No	Visited[‘C’] < visited[[‘D] = 20 < 25? Yes	Nodes.remove(‘C’)	nodes = {'F', 'G','D', 'E'}	Current_weight = 20	Graph.edges[‘C’] =‘A’,‘D’ Edge = ‘A’	Weight = 20 + Graph.distances[‘C’,‘A’] = error	No, A is in visited	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}
											Edge = ‘D’	Weight = 20 + Graph.distances[‘C’,‘D’] = 20 + 30 = 50	No, D is in visited, 50 > 25	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}
Yes, nodes = {'F', 'G','D', 'E'}	None	Node = ‘F’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:60}	{'B': 'A', 'C': 'A', ‘D’:‘B’, ‘E’:‘B’}	No, proceed to new loop										
	None	Node = ‘G’	{'A': 0, 'B': 10,	{'B': 'A', 'C': 'A',	No, proceed										

			'C': 20, 'D':25, 'E':60}	'D':'B', 'E':'B'}	to new loop										
	None	Node = 'D'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'B'}	Yes	Yes, min_node = 'D'	Visited['D'] < visited ['D'] = 25 < 25? no								
	D	Node = 'E'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'B'}	Yes	no	Visited['E'] < visited['D] = 60 < 25? no	Nodes.remove('D')	nodes = {'F', 'G', 'E'}	Current_weight = 25	Graph.edges['D'] ='B','C','E' Edge = 'B'	Weight = 25 + Graph.distances['D','B'] = error	No, B is in visited	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'B'}
											Edge = 'C'	Weight = 25 + Graph.distances['D','C'] = error	No, C is in visited	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'B'}
											Edge = 'E'	Weight = 25 + Graph.distances['D','E'] = 25 + 30 = 55	Yes, weight < visited['E'] = 55 < 60 Visited['E']= 55 Path['E']= 'D'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}
Yes, nodes = {'F', 'G', 'E'}	None	Node = 'F'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}	No, proceed to new loop										
	None	Node = 'G'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}	No, proceed to new loop										
	None	Node = 'E'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}	Yes	Yes, min_node = 'E'	Visited['E'] < visited ['E'] = 55 < 55? no	Nodes.remove('E')	nodes = {'F', 'G'}	Current_weight = 55	Graph.edges['E'] ='B','D','F' Edge = 'B'	Weight = 55 + Graph.distances['E','B''] = error	No, B is in visited	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}
											Edge = 'D'	Weight = 55 + Graph.distances['E','D''] = error	No, D is in visited	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D'}
											Edge = 'F'	Weight = 55 + Graph.distances['E','F''] = 55 + 5 = 60	Yes, edge not in visited Visited['F'] = 60 Path['F] = 'E'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55, 'F':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D, 'F':'E'}
Yes, nodes = {'F', 'G',}	None	Node = 'F'	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55, 'F':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D, 'F':'E'}	Yes	Yes, min_node = 'F'	Visited['F'] < visited ['F'] = 60 < 60? no	Nodes.removes('F')	nodes = {'G'}	Current_weight = 60	Graph.edges['F] ='E','G' Edge = 'E'	Weight = 60 + Graph.distances['F','E'] = error	No, E is in visited	{'A': 0, 'B': 10, 'C': 20, 'D':25, 'E':55, 'F':60}	{'B': 'A', 'C': 'A', 'D':'B', 'E':'D, 'F':'E'}
											Edge = 'G'	Weight = 60 + Graph.distances['F','G'] = 60 + 2 = 62	Yes, edge not in visited	{'A': 0, 'B': 10, 'C': 20,	{'B': 'A', 'C': 'A', 'D':'B',

													Visited[‘G’] = 62 Path[‘G’] = ‘F’	‘D’:25, ‘E’:55, ‘F’:60, ‘G’:62}	‘E’:’D, ‘F’:’E’, ‘G’:’F’}
Yes, nodes = {'G'}	None	Node = ‘G’	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:55, ‘F’:60, ‘G’:62}	{'B': 'A', 'C': 'A', ‘D’:’B’, ‘E’:’D, ‘F’:’E’, ‘G’:’F’}	Yes	Yes, min_node = ‘G’	Visited[‘G’] < visited [‘G’] = 62 < 62? no	Nodes.removes(‘G’)	nodes = {}	Current_weight= 62	Graph.edges[‘G’] = ‘F’ Edge = ‘F’	Weight = 62 + Graph.distances[‘G’,’F’] = error	No, F is in visited	{'A': 0, 'B': 10, 'C': 20, ‘D’:25, ‘E’:55, ‘F’:60, ‘G’:62}	{'B': 'A', 'C': 'A', ‘D’:’B’, ‘E’:’D, ‘F’:’E’, ‘G’:’F’}