Directed Graph implementation

**Language** : Python

**Version**: 3

**Data Structures**: collections, list, stack, queue

Steps to execute program

1. This program provides below 8 options for graph implementation.

* Create graph
* DFS traversal
* BFS traversal
* Shortest path using dijkstra’s algorithm
* Detect cycle in graph
* Check graph is bipartite or not
* Check graph is tree or not

1. Press 1 to start creating directed graph.

User must provide number of vertices and edges in graph.

Then provide edges between vertices with their respective distances

e.g 0,1,2 where 0 and 1 are nodes and 2 is distance between them

Please enter only numeric values for nodes and distances.

1. Once graph is created user may use options from above list.
2. **DFS traversal:**  Press 2 for DFS traversal of graph.

Type node from which DFS needs to be traversed.

1. **BFS traversal:**  Press 3 for BFS traversal of graph.

Type node from which BFS needs to be traversed.

1. **Shortest path** : Press 4 to find shortest path between source node and each node.

Type source node and press enter.

1. **Detect Cycle** : Press 5 to check if graph has cycle in it or not.
2. **Check graph is bipartite :** Press 6 to check is graph is bipartite or not.
3. **Check graph is tree** : Press 7 to check if graph is tree or not.
4. **Exit**: Press 8 to exit the console

For storing graph,edges I have used default dictionary list.

Functions created :

* **AddEdge**: This function is used to add edges to graph
* **DFS** : This function is used to traverse the graph using DFS traversal with the help of queue and stack data structure.
* **BFS**() : This function is used to traverse the graph using BFS traversal with the help of queue data structure.
* **Dijkstra**() : This function calculates the shortest path and distance between source node and each node in graph and also prints shortest distances and predecessor of each node.
* **isCyclic**() : This function is used to check whether there is any cycle present or not in graph.
* **isTree**() : This function is used to check whether graph is tree or not.
* **isBipartite**(): This function is used to check bipartiteness of graph.