# Class GraphUtil

java.lang.Object
    GraphUtil

---

```
public class GraphUtil
extends java.lang.Object
```

## *Constructor Summary*

**Constructors**

| Constructor and Description |
| --- |
| `GraphUtil()` |

## *Method Summary*

**All Methods** **Static Methods** **Concrete Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| static void | `bfs(Graph g, Vertex vsrc)`<br>Lists (prints) the vertices (labels) visited by the Breadth-First-Search traversal algorithm on the graph g from the vsrc vertex. |
| static void | `dfs(Graph g, Vertex vsrc)`<br>Lists (prints) the vertices (labels) visited by the Depth-First-Search traversal algorithm on the graph g from the vsrc vertex. |
| static void | `dijkstra(Graph g, Vertex vsrc)`<br>Applies the Dijkstra's shortest path algorithm to the graph g from the vsrc vertex. |
| static void | `floyedWarshall(Graph g)`<br>applies Floyd-Warshall algorithm to compute the all-pairs shortest paths algorithm to graph g; also prints the paths from every vertex to every other vertex by calling printAllPairsShortestPaths |

| | |
|---|---|
| static void | **kruskal**(Graph g)<br>applies Kruskal's algorithm to find the minim spanning tree in graph g and prints the total cost and the edges in the mst in the order they are added to it. |
| static void | **printAllPairsShortestPaths**(Graph g, double[][] dist, Vertex[][] pred)<br>Using the given matrices, reconstructs and prints the shortest path from every vertex to every other vertex in g. |
| static void | **printDijkstraPaths**(Graph g, java.lang.String src, java.lang.String dst)<br>Prints the path from src to dst assuming Dijkstra's shortest path algorithm had been applied and vertices have proper information (cost and predecessor of the path, if exists). |
| static void | **testGraph**(java.lang.String gName)<br>Test the given graph by applying BFS, DFS and Dijkstra's algorithm |
| static void | **testGraphs**(int min, int max)<br>Test the given graphs by applying BFS, DFS and Dijkstra's algorithm |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## *Constructor Detail*

### GraphUtil

```
public GraphUtil()
```

## *Method Detail*

### bfs

```
public static void bfs(Graph g,
                       Vertex vsrc)
```

Lists (prints) the vertices (labels) visited by the Breadth-First-Search traversal algorithm on the graph g from the vsrc vertex.

**Parameters:**

`g - the graph`

`vsrc - the source vertex`

### dfs

```
public static void dfs(Graph g,
                       Vertex vsrc)
```

Lists (prints) the vertices (labels) visited by the Depth-First-Search traversal algorithm on the graph g from the vsrc vertex.

**Parameters:**

`g - the graph`

`vsrc - the source vertex`

### dijkstra

```
public static void dijkstra(Graph g,
                            Vertex vsrc)
```

Applies the Dijkstra's shortest path algorithm to the graph g from the vsrc vertex. Prints the paths from src to each vertex in the graph g by calling printDijkstraPaths.

**Parameters:**

`g - the graph`

`vsrc - the source vertex`

### printDijkstraPaths

```
public static void printDijkstraPaths(Graph g,
                                      java.lang.String src,
                                      java.lang.String dst)
```

Prints the path from src to dst assuming Dijkstra's shortest path algorithm had been applied and vertices have proper information (cost and predecessor of the path, if exists).

**Parameters:**

`g - the graph`

`src - the source vertex`

`dst - the destination vertex`

### floyedWarshall

```
public static void floyedWarshall(Graph g)
```

applies Floyd-Warshall algorithm to compute the all-pairs shortest paths algorithm to graph g; also prints the paths from every vertex to every other vertex by calling printAllPairsShortestPaths

**Parameters:**

`g - the target graph`

### printAllPairsShortestPaths

```
public static void printAllPairsShortestPaths(Graph g,
                                              double[][] dist,
                                              Vertex[][] pred)
```

Using the given matrices, reconstructs and prints the shortest path from every vertex to every other vertex in g.

**Parameters:**

`g - the target graph`

`dist - the distance matrix`

`pred - the predecessor matrix`

### kruskal

```
public static void kruskal(Graph g)
```

applies Kruskal's algorithm to find the minim spanning tree in graph g and prints the total cost and the edges in the mst in the order they are added to it.

**Parameters:**

`g - the target graph`

### testGraph

```
public static void testGraph(java.lang.String gName)
```

Test the given graph by applying BFS, DFS and Dijkstra's algorithm

**Parameters:**

`gName - the name of the graph`

### testGraphs

```
public static void testGraphs(int min,
                              int max)
```

Test the given graphs by applying BFS, DFS and Dijkstra's algorithm

**Parameters:**

`min - the min index`

`max - the max index`