|  |  |
| --- | --- |
| **BFS** | **DFS** |
| **BFS** Stands for “**Breadth First Search**”. | **DFS** stands for “**Depth First Search**”. |
| BFS starts traversal from the root node and then explore the search in the level by level manner i.e. as close as possible from the root node. | DFS starts the traversal from the root node and explore the search as far as possible from the root node i.e. depth wise. |
| Breadth First Search can be done with the help of **queue** i.e. **FIFO** implementation. | Depth First Search can be done with the help of **Stack** i.e. **LIFO** implementations. |
| This algorithm works in single stage. The visited vertices are removed from the queue and then displayed at once. | This algorithm works in two stages – in the first stage the visited vertices are pushed onto the stack and later on when there is no vertex further to visit those are popped-off. |
| BFS is **slower** than DFS. | DFS is more **faster** than BFS. |
| BFS requires **more** memory compare to DFS. | DFS require **less** memory compare to BFS. |
| **Applications of BFS** > To find Shortest path > Single Source & All pairs shortest paths > In Spanning tree > In Connectivity | **Applications of DFS** > Useful in Cycle detection > In Connectivity testing > Finding a path between V and W in the graph. > useful in finding spanning trees & forest. |
| BFS is useful in finding shortest path. BFS can be used to find the shortest distance between some starting node and the remaining nodes of the graph. | DFS in not so useful in finding shortest path. It is used to perform a traversal of a general graph and the idea of DFS is to make a path as long as possible, and then go back (**backtrack**) to add branches also as long as possible. |
| **Example :**  [BFS traversal](http://freefeast.info/wp-content/uploads/2014/04/BFS-traversal.jpg) | **Example :** [DFS Traversal](http://freefeast.info/wp-content/uploads/2014/04/DFS-Traversal.jpg) |