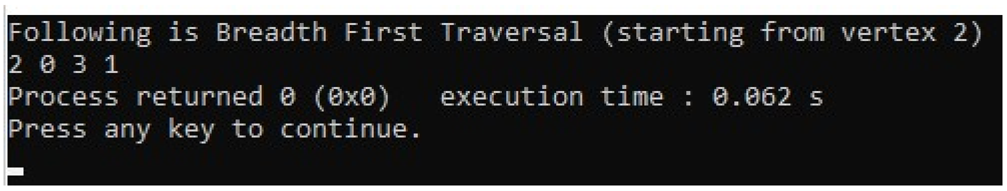
WAP TO TRAVERSE A GRAPH USING BFS METHOD

```
#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>


#define MAX_VERTICES 50

typedef struct Graph_t {

  int V;

  bool adj[MAX_VERTICES][MAX_VERTICES];

} Graph;

Graph* Graph_create(int V)

{

  Graph* g = malloc(sizeof(Graph));

  g->V = V;


  for (int i = 0; i < V; i++) {

    for (int j = 0; j < V; j++) {

      g->adj[i][j] = false;

    }

  }


  return g;

}

void Graph_destroy(Graph* g) { free(g); }

void Graph_addEdge(Graph* g, int v, int w)

{
  g->adj[v][w] = true;
}


void Graph_BFS(Graph* g, int s)
{


  bool visited[MAX_VERTICES];
  for (int i = 0; i < g->V; i++) {
    visited[i] = false;
  }


  int queue[MAX_VERTICES];
  int front = 0, rear = 0;


  visited[s] = true;
  queue[rear++] = s;


  while (front != rear) {
    s = queue[front++];
    printf("%d ", s);
    for (int adjacent = 0; adjacent < g->V;
         adjacent++) {
      if (g->adj[s][adjacent] && !visited[adjacent]) {
          visited[adjacent] = true;
          queue[rear++] = adjacent;
      }
    }
  }
}
int main()
{
  Graph* g = Graph_create(4);
  Graph_addEdge(g, 0, 1);
  Graph_addEdge(g, 0, 2);
  Graph_addEdge(g, 1, 2);
  Graph_addEdge(g, 2, 0);
  Graph_addEdge(g, 2, 3);
  Graph_addEdge(g, 3, 3);

  printf("Following is Breadth First Traversal ""(starting from vertex 2) \n");
  Graph_BFS(g, 2);
  Graph_destroy(g);
  return 0;
}
```

OUTPUT:



```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```