

program tentang algoritma Algoritma Breadth First Search (BFS) dalam bahasa C

Source Code:

```
#include <stdio.h>
#include <stdlib.h>

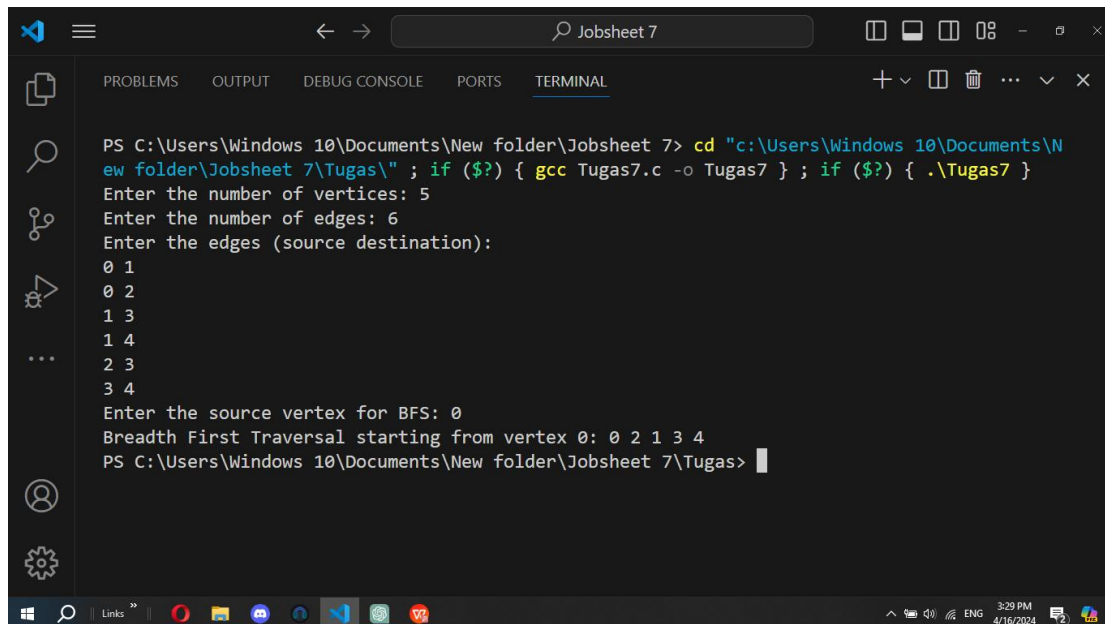
#define MAX_VERTICES 100
// Definisi struktur simpul/graf
typedef struct Node
{
    int data;
    struct Node *next;
} Node;
// Fungsi untuk membuat simpul baru
Node *createNode(int data)
{
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
// Fungsi untuk menambahkan tepi antara dua simpul
void addEdge(Node *adjacencyList[], int src, int dest)
{
    Node *newNode = createNode(dest);
    newNode->next = adjacencyList[src];
    adjacencyList[src] = newNode;
}
// Fungsi untuk mencetak jalur BFS dari simpul sumber
void BFS(Node *adjacencyList[], int vertices, int source)
{
    int visited[MAX_VERTICES] = {0}; // Array untuk menyimpan status
    kunjungan simpul
    int queue[MAX_VERTICES];          // Queue untuk menyimpan simpul
    yang akan dikunjungi
    int front = 0, rear = 0;          // Indeks depan dan belakang untuk
    queue
    visited[source] = 1;              // Menandai simpul sumber sebagai sudah
    dikunjungi
    queue[rear++] = source; // Menambahkan simpul sumber ke dalam queue
    printf("Breadth First Traversal starting from vertex %d: ", source);
    while (front < rear)
    {
        int current = queue[front++]; // Mengambil simpul dari depan
        queue
        printf("%d ", current);
```

```

        // Melintasi semua simpul yang terhubung dengan simpul saat ini
        dan belum dikunjungi
        for (Node *temp = adjacencyList[current]; temp != NULL; temp =
temp->next)
        {
            int neighbor = temp->data;
            if (!visited[neighbor])
            {
                visited[neighbor] = 1;    // Menandai simpul tetangga
sebagai sudah dikunjungi
                queue[rear++] = neighbor; // Menambahkan simpul
tetangga ke dalam queue
            }
        }
    }
}
int main()
{
    int vertices, edges, i, src, dest;
    printf("Enter the number of vertices: ");
    scanf("%d", &vertices);
    Node *adjacencyList[MAX_VERTICES] = {NULL}; // Array untuk
merepresentasikan graf menggunakan linked list
    printf("Enter the number of edges: ");
    scanf("%d", &edges);
    printf("Enter the edges (source destination): \n");
    for (i = 0; i < edges; ++i)
    {
        scanf("%d%d", &src, &dest);
        addEdge(adjacencyList, src, dest);
    }
    printf("Enter the source vertex for BFS: ");
    scanf("%d", &src);
    BFS(adjacencyList, vertices, src);
    return 0;
}

```

Screenshot Output:



```
PS C:\Users\Windows 10\Documents\New folder\Jobsheet 7> cd "c:\Users\Windows 10\Documents\New folder\Jobsheet 7\Tugas\" ; if ($?) { gcc Tugas7.c -o Tugas7 } ; if ($?) { .\Tugas7 }
Enter the number of vertices: 5
Enter the number of edges: 6
Enter the edges (source destination):
0 1
0 2
1 3
1 4
2 3
3 4
Enter the source vertex for BFS: 0
Breadth First Traversal starting from vertex 0: 0 2 1 3 4
PS C:\Users\Windows 10\Documents\New folder\Jobsheet 7\Tugas>
```

Penjelasan tentang Algoritma BFS:

Breadth First Search (BFS) adalah salah satu algoritma yang digunakan untuk melintasi atau mencari semua simpul dalam graf atau struktur data yang berbentuk pohon, secara berurutan dari simpul awal (sumber) ke simpul yang bertetangga dengannya.

Pada dasarnya, algoritma BFS menggunakan pendekatan 'level by level'. Ini berarti ia mulai dari simpul awal (sumber), menelusuri semua simpul yang bertetangga dengannya terlebih dahulu, kemudian simpul-simpul tersebut akan menjadi sumber untuk melintasi simpul-simpul yang terhubung dengannya, dan begitu seterusnya.

Prinsip Queue dalam Algoritma BFS:

Queue digunakan dalam algoritma BFS untuk menyimpan simpul-simpul yang akan dikunjungi selanjutnya. Ketika sebuah simpul dikunjungi, simpul-simpul yang bertetangga dengannya ditambahkan ke dalam queue. Dengan menggunakan queue, simpul-simpul dikunjungi secara berurutan sesuai dengan urutan masuknya ke dalam queue, sehingga algoritma BFS memastikan bahwa simpul-simpul ditemukan secara 'level by level'. Dengan kata lain, simpul yang lebih dekat dengan sumber akan dikunjungi terlebih dahulu sebelum melanjutkan ke simpul-simpul yang lebih jauh.