**TUGAS PRATIKUM**

**STRUKTUR DATA**

**“JOBSHEET 7”**

**Dosen Pengampu : Randi Proska Sandra, S.Pd., M.Sc**



**Disusun Oleh :**

**Radhia Aulia Nisa**

**23343049**

**PROGRAM STUDI INFORMATIKA**

**DEPARTEMEN ELEKTRONIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS NEGERI PADANG**

**2024**

**TUGAS**

1. Buatlah program tentang algoritma Algoritma Breadth First Search (BFS) dalam bahasa C. Lalu, jelaskan terkait algoritma tersebut dan bagaimana prinsip queue digunakan dalam algoritma tersebut!
2. Buatlah tugas anda dalam dokumen .PDF lengkap dengan source code dan penjelasan
3. Dokumen .PDF dan source code di upload ke github bersama hasil percobaan dan Latihan.

**JAWAB :**

1. SOURCE CODE PROGRAM

//created by Radhia Aulia Nisa 23343049

#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTICES 100

// Mendefinisikan struktur untuk simpul dalam graf

typedef struct Node {

int vertex;

struct Node\* next;

} Node;

// Mendefinisikan struktur untuk representasi graf menggunakan linked list

typedef struct Graph {

int numVertices;

Node\* adjLists[MAX\_VERTICES];

int visited[MAX\_VERTICES];

} Graph;

// Fungsi untuk membuat simpul baru dalam graf

Node\* createNode(int v) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->vertex = v;

newNode->next = NULL;

return newNode;

}

// Fungsi untuk membuat graf kosong dengan sejumlah verteks

Graph\* createGraph(int vertices) {

Graph\* graph = (Graph\*)malloc(sizeof(Graph));

graph->numVertices = vertices;

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

graph->adjLists[i] = NULL;

graph->visited[i] = 0;

}

return graph;

}

// Fungsi untuk menambahkan tepian ke graf

void addEdge(Graph\* graph, int src, int dest) {

// Menambahkan tepian dari src ke dest

Node\* newNode = createNode(dest);

newNode->next = graph->adjLists[src];

graph->adjLists[src] = newNode;

// Karena graf tidak berarah, tambahkan tepian dari dest ke src juga

newNode = createNode(src);

newNode->next = graph->adjLists[dest];

graph->adjLists[dest] = newNode;

}

// Fungsi untuk mencetak hasil pencarian jalur

void printPath(int path[], int v) {

if (path[v] == -1) {

printf("%d ", v);

return;

}

printPath(path, path[v]);

printf("%d ", v);

}

// Algoritma BFS untuk mencari jalur dari simpul awal ke simpul tujuan

void BFS(Graph\* graph, int startVertex, int endVertex) {

int queue[MAX\_VERTICES];

int front = 0, rear = 0;

int visited[MAX\_VERTICES];

int path[MAX\_VERTICES];

for (int i = 0; i < graph->numVertices; i++) {

visited[i] = 0;

path[i] = -1;

}

visited[startVertex] = 1;

queue[rear++] = startVertex;

while (front < rear) {

int currentVertex = queue[front++];

Node\* temp = graph->adjLists[currentVertex];

while (temp) {

int adjVertex = temp->vertex;

if (!visited[adjVertex]) {

visited[adjVertex] = 1;

queue[rear++] = adjVertex;

path[adjVertex] = currentVertex;

}

temp = temp->next;

}

}

printf("Jalur dari %d ke %d: ", startVertex, endVertex);

printPath(path, endVertex);

}

int main() {

Graph\* graph = createGraph(6);

addEdge(graph, 0, 1);

addEdge(graph, 0, 2);

addEdge(graph, 1, 3);

addEdge(graph, 1, 4);

addEdge(graph, 2, 4);

addEdge(graph, 3, 4);

addEdge(graph, 3, 5);

addEdge(graph, 4, 5);

int startVertex = 0;

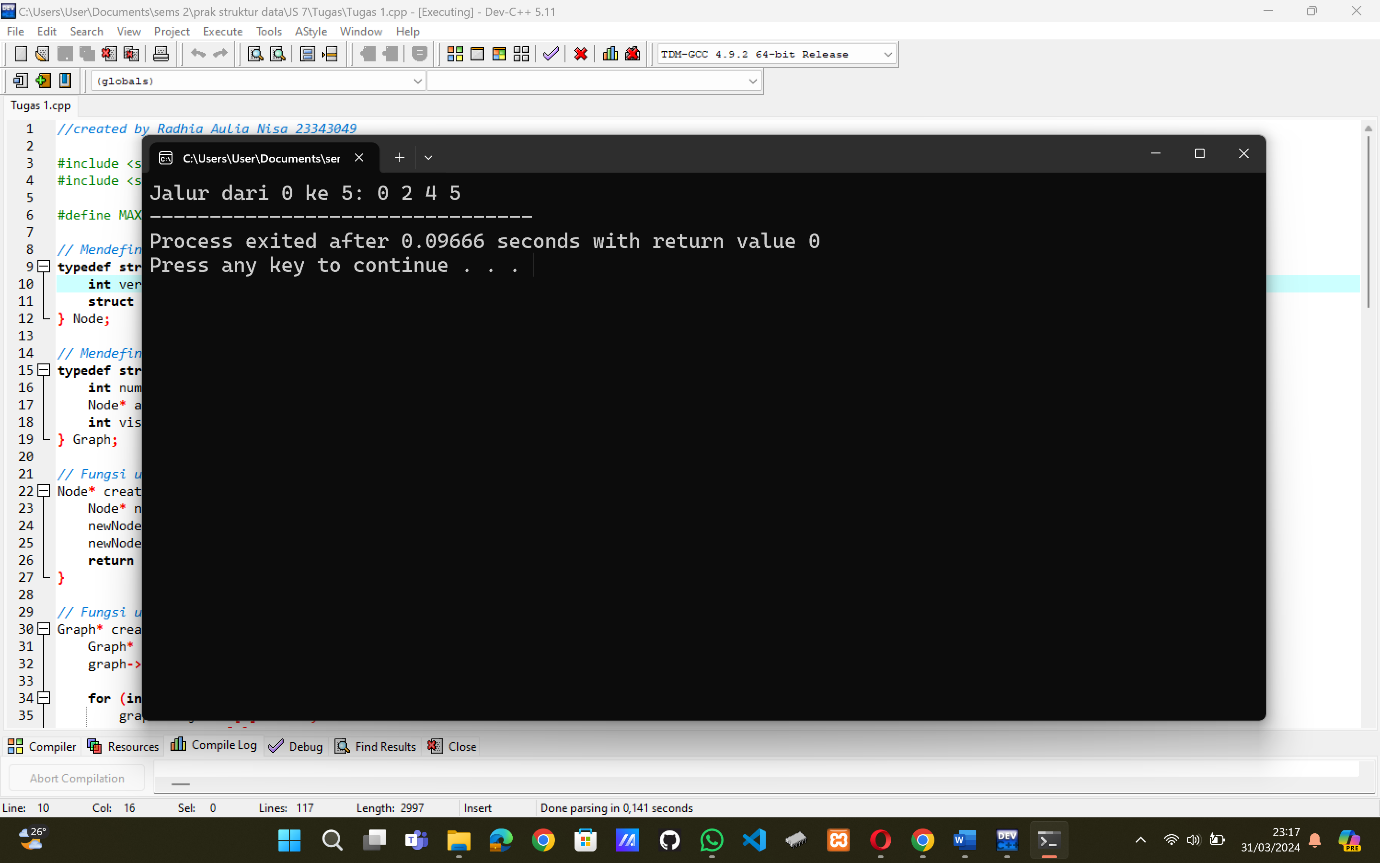
int endVertex = 5;

BFS(graph, startVertex, endVertex);

return 0;

}

1. SCREENSHOT PROGRAM



1. PENJELASAN

Algoritma Breadth First Search (BFS) merupakan algoritma traversal graf yang mengeksplorasi atau menelusuri semua simpul dalam graf pada kedalaman saat ini (level n) sebelum berpindah ke simpul pada tingkat kedalaman berikutnya (level n+1). BFS dimulai dari titik tertentu dan mengunjungi semua simpul yang bertetangga terlebih dahulu sebelum berpindah ke tetangga tingkat berikutnya. BFS biasanya digunakan dalam algoritma pencarian jalur, komponen terhubung, dan shorter path problem dalam grafik.

Algoritma Breadth First Search (BFS) adalah salah satu metode pencarian jalur yang umum digunakan. Ini menerapkan teknik traversal graf secara melebar, artinya itu menjelajahi simpul-simpul graf secara berurutan pada setiap tingkat jarak dari simpul awal. Tujuan utamanya adalah menemukan jalur dari simpul awal ke simpul tujuan dalam suatu graf.

BFS biasanya digunakan pada graf statis, yang sudah terbentuk sebelum pencarian dilakukan. Graf ini direpresentasikan dalam struktur data seperti graf dengan simpul-simpul yang saling terhubung, matriks adjacency yang merepresentasikan hubungan antara setiap pasangan simpul dengan nilai boolean, atau struktur data peta yang memetakan simpul ke daftar simpul-simpul tetangganya.

Selama pencarian, BFS membangun solusi dengan membangkitkan status-status dalam bentuk simpul pohon. Ini dilakukan dengan menerapkan langkah-langkah atau operator legal ke simpul-simpul lainnya dalam jalur pencarian. Jalur dari simpul awal ke simpul tujuan akan mengandung serangkaian langkah atau operator yang membentuk solusi untuk permasalahan yang diberikan. Solusi ini dapat dinilai sebagai benar atau salah tergantung pada apakah solusi tersebut memenuhi kriteria yang diinginkan.

Cara kerja algoritma Breadth First Search yaitu masukkan simpul ujung ke dalam sebuah antrean kemudian ambil simpul dari awal antrean. Lakukan pengecekan apakah simpul awal merupakan solusi. Jika simpul merupakan solusi pencarian selesai dan hasil dikembalikan. Jika simpul bukan merupakan solusi, masukan seluruh simpul yang bertetangga dengan simpul tersebut. Apabila semua simpul sudah dicek dan antrean kosong, pencarian selesai dengan mengembalikan hasil solusi tidak ditemukan. Pencarian diulang dari simpul awal antrean.

Tujuan Algoritma BFS :

1. Efisiensi Pencarian Jalur Terpendek

Algoritma BFS adalah salah satu alat terbaik untuk mencari jalur terpendek antara dua titik dalam graf. Dalam pemodelan transportasi, navigasi, atau jaringan komunikasi, BFS membantu dalam menemukan rute tercepat, menghemat waktu dan sumber daya.

2. Identifikasi Komponen Terhubung

BFS digunakan untuk mengidentifikasi komponen terhubung dalam graf. Ini berguna dalam analisis jaringan sosial, pemodelan jaringan komputer, dan bahkan dalam pemahaman hubungan antara objek dalam berbagai konteks.

3. Solusi untuk Teka-Teki dan Permainan

Dalam permainan teka-teki dan masalah kombinatorial, BFS membantu menemukan semua solusi yang mungkin. Ini memberikan pemain atau penyelesaian masalah gambaran lengkap tentang berbagai pilihan yang dapat diambil dan membantu dalam mengambil keputusan yang tepat.

1. DAFTAR PUSTAKA

Munir, Rinaldi. 2021. Graf (Bag.1). [Online] Tersedia dalam <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf>. Diakses pada tanggal 31 Maret 2024.

Munir, Rinaldi. 2021. Breadt/Depth First Search (BFS/DFS) Ba. [Online] Tersedia dalam <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf>. Diakses pada 28 Maret 2024.

GeeksForGeeks. Breadth First Search or BFS for a Graph. Tersedia dalam <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>. Diakses pada 28 Maret 2024.

Tamara, Defi. 2021. BFS (Breadth First Search) : Pengertian, Kekurangan, Kelebihan, dan Contohnya. Tersedia dalam <https://medium.com/@defytamara2610/bfs-breadth-first-search-pengertian-kekurangan-kelebihan-dan-contohnya-775a7d808fbc>. Diakses pada 29 Maret 2024.

Annisa. 2023. Algoritma BFS (Breadth First Search) : Pengertian, Fungsi, dan Cara Kerja. Tersedia dalam <https://fikti.umsu.ac.id/algoritma-bfs-breadth-first-search-pengertian-fungsi-dan-cara-kerja/>. Diakses pada 29 Maret 2024.