TUGAS

PRATIKUM STRUKTUR DATA

“JOBSHEET 7”

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

Disusun Oleh :

Setya Carina Rianti

23343054

PROGRAM STUDI INFORMATIKA

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.

- **Sourcecode**

```
//Setya Carina Rianti 23343054

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

#define MAX 100

// Struktur untuk representasi graf
struct Graph {
    int numVertices;
    int adjMatrix[MAX][MAX];
};

// Struktur untuk antrian
struct Queue {
    int items[MAX];
    int front;
    int rear;
};

// Fungsi untuk membuat graf
struct Graph* createGraph(int vertices) {
```

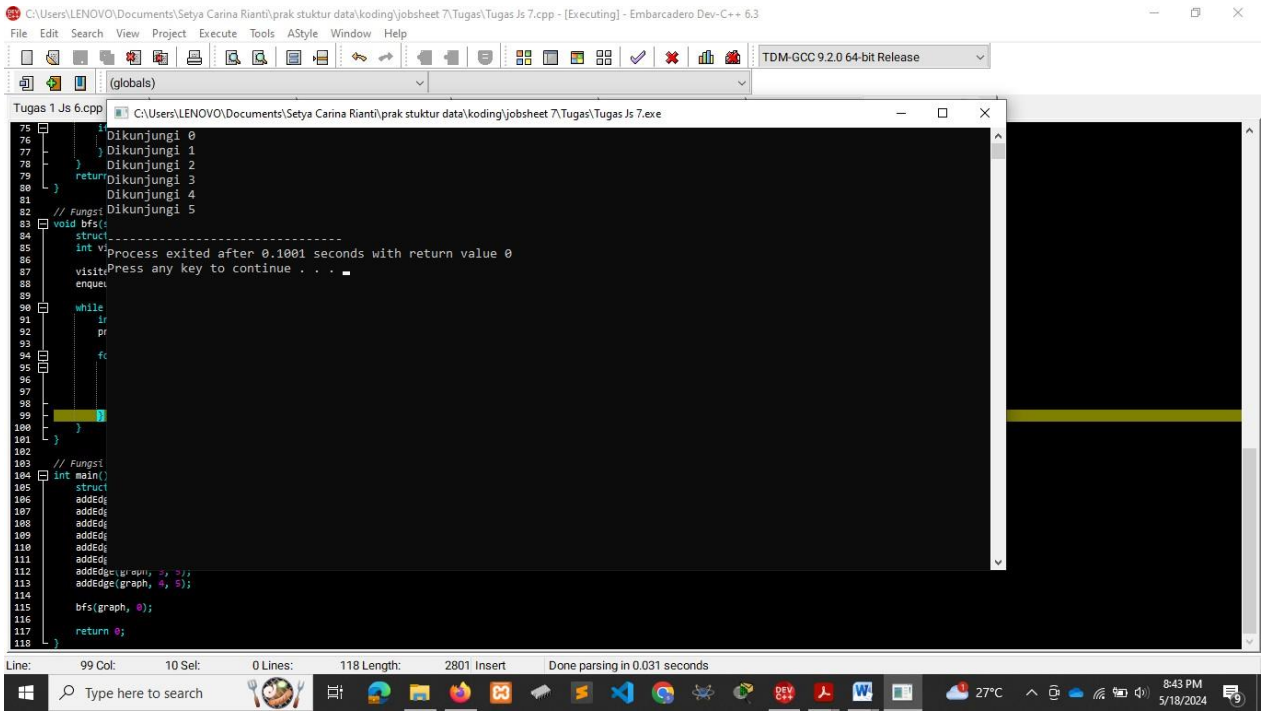
```
    struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
    graph->numVertices = vertices;

    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            graph->adjMatrix[i][j] = 0;
        }
    }

    return graph;
}

// Fungsi untuk menambahkan edge pada graf
void addEdge(struct Graph* graph, int src, int dest) {
    graph->adjMatrix[src][dest] = 1;
    graph->adjMatrix[dest][src] = 1;
}

// Fungsi untuk membuat antrian
struct Queue* createQueue() {
    struct Queue* queue = (struct Queue*)malloc(sizeof(struct Queue));
    queue->front = -1;
    queue->rear = -1;
    return queue;
}

// Fungsi untuk mengecek apakah antrian kosong
int isEmpty(struct Queue* queue) {
    return queue->rear == -1;
}
```

```
// Fungsi untuk menambahkan elemen ke dalam antrian
void enqueue(struct Queue* queue, int value) {
    if (queue->rear == MAX - 1) {
        printf("\nAntrian penuh\n");
    } else {
        if (queue->front == -1)
            queue->front = 0;
        queue->rear++;
        queue->items[queue->rear] = value;
    }
}

// Fungsi untuk menghapus elemen dari antrian
int dequeue(struct Queue* queue) {
    int item;
    if (isEmpty(queue)) {
        printf("Antrian kosong\n");
        item = -1;
    } else {
        item = queue->items[queue->front];
        queue->front++;
        if (queue->front > queue->rear) {
            queue->front = queue->rear = -1;
        }
    }
    return item;
}

// Fungsi BFS
```

```
void bfs(struct Graph* graph, int startVertex) {
    struct Queue* queue = createQueue();
    int visited[MAX] = {0};

    visited[startVertex] = 1;
    enqueue(queue, startVertex);

    while (!isEmpty(queue)) {
        int currentVertex = dequeue(queue);
        printf("Dikunjungi %d\n", currentVertex);

        for (int i = 0; i < graph->numVertices; i++) {
            if (graph->adjMatrix[currentVertex][i] == 1 &&
!visited[i]) {
                visited[i] = 1;
                enqueue(queue, i);
            }
        }
    }
}

// Fungsi utama
int main() {
    struct Graph* graph = createGraph(6);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 2, 4);
    addEdge(graph, 3, 4);
    addEdge(graph, 3, 5);
```

```
    addEdge(graph, 4, 5);

    bfs(graph, 0);

    return 0;
}
```

- **Screenshot**



- **Penjelasan**

Breadth-First Search (BFS) adalah algoritma untuk melakukan penelusuran atau pencarian dalam struktur data graf atau pohon. BFS mengeksplorasi simpul-simpul (node) pada tingkat (level) yang sama sebelum bergerak ke tingkat berikutnya. Dengan kata lain, BFS mengunjungi semua simpul pada satu tingkat sebelum berpindah ke tingkat berikutnya.

Prinsip Kerja BFS

1. **Inisialisasi**:
   - Mulai dari simpul awal (root atau start node).
   - Tandai simpul ini sebagai dikunjungi.
   - Masukkan simpul ini ke dalam antrian (queue).

2. **Proses Penelusuran**:
   - Selama antrian tidak kosong:
     - Ambil simpul dari depan antrian.
     - Kunjungi dan proses simpul tersebut.
     - Masukkan semua tetangga (neighbor) dari simpul yang diambil yang belum dikunjungi ke dalam antrian dan tandai mereka sebagai dikunjungi.

Contoh Penggunaan BFS

- **Penelusuran Graf**: Menemukan semua simpul yang dapat dicapai dari simpul awal.
- **Pencarian Jalur Terpendek**: Dalam graf tak berbobot, BFS dapat digunakan untuk menemukan jalur terpendek antara dua simpul.
- **Masalah Puzzle**: Seperti menyelesaikan permainan teka-teki atau permainan papan yang memerlukan penjelajahan semua kemungkinan langkah.

Struktur Program

1. **Definisi Struktur Graph**
   - Struktur `Graph` merepresentasikan graf dengan menggunakan matriks ketetanggaan (adjacency matrix).
   - `numVertices` menyimpan jumlah simpul (vertex) dalam graf.
   - `adjMatrix` adalah matriks 2D yang menyimpan hubungan antar simpul (1 jika ada edge, 0 jika tidak ada).

2. **Definisi Struktur Queue**
   - Struktur `Queue` digunakan untuk menyimpan simpul-simpul yang akan dieksplorasi.
   - `items` adalah array yang menyimpan elemen-elemen antrian.
   - `front` dan `rear` menyimpan indeks depan dan belakang antrian.
3. **Fungsi untuk Membuat Graph**
   - `createGraph` menginisialisasi graf dengan jumlah simpul tertentu dan mengatur semua nilai dalam `adjMatrix` ke 0 (tidak ada edge).

4. **Fungsi untuk Menambahkan Edge pada Graph**
   - `addEdge` menambahkan edge antara dua simpul dalam graf dengan mengatur nilai yang sesuai dalam `adjMatrix` menjadi 1.

5. **Fungsi untuk Membuat Queue**
   - `createQueue` menginisialisasi antrian dengan `front` dan `rear` diatur ke -1 (menandakan antrian kosong).

6. **Fungsi untuk Mengecek Apakah Queue Kosong**
   - `isEmpty` mengembalikan 1 (true) jika `rear` adalah -1, yang berarti antrian kosong.

7. **Fungsi untuk Menambahkan Elemen ke dalam Queue**
   - `enqueue` menambahkan elemen ke dalam antrian. Jika antrian penuh, akan menampilkan pesan "Antrian penuh".

8. **Fungsi untuk Menghapus Elemen dari Queue**
   - `dequeue` menghapus elemen dari antrian dan mengembalikannya. Jika antrian kosong, akan menampilkan pesan "Antrian kosong".

9. **Fungsi BFS**
   - `bfs` melakukan traversal BFS pada graf mulai dari simpul `startVertex`.
   - Array `visited` digunakan untuk melacak simpul yang sudah dikunjungi.
   - Simpul awal dimasukkan ke dalam antrian dan ditandai sebagai dikunjungi.
   - Selama antrian tidak kosong, simpul di depan antrian diambil dan tetangganya yang belum dikunjungi dimasukkan ke dalam antrian dan ditandai sebagai dikunjungi.
   - Setiap simpul yang dikunjungi akan menampilkan pesan "Dikunjungi X", di mana X adalah nomor simpul yang dikunjungi.

10. **Fungsi Utama**
    - `main` menginisialisasi graf dengan 6 simpul dan menambahkan beberapa edge.
    - Memanggil fungsi `bfs` untuk memulai traversal BFS dari simpul 0.

Penjelasan Fungsi BFS

1. **Inisialisasi**
   - Buat antrian kosong dengan memanggil `createQueue`.
   - Buat array `visited` untuk melacak simpul yang sudah dikunjungi, inisialisasi semua elemen ke 0.
2. **Proses BFS**
   - Tandai simpul awal (`startVertex`) sebagai dikunjungi (`visited[startVertex] = 1`) dan masukkan ke dalam antrian (`enqueue(queue, startVertex)`).
   - Selama antrian tidak kosong, lakukan langkah berikut:
   - Ambil simpul di depan antrian (`currentVertex = dequeue(queue)`).
   - Tampilkan pesan bahwa simpul tersebut dikunjungi (`printf("Dikunjungi %d\n", currentVertex)`).
   - Periksa semua tetangga dari simpul yang diambil. Jika tetangga tersebut belum dikunjungi dan ada edge antara `currentVertex` dan tetangga, tandai tetangga sebagai dikunjungi dan masukkan ke dalam antrian.
3. **Output**
   - Setiap simpul yang dikunjungi akan menampilkan pesan "Dikunjungi X", menunjukkan bahwa simpul tersebut telah diproses oleh BFS.

Kesimpulan

Dengan implementasi ini, algoritma BFS mengeksekusi traversal graf secara level-per-level menggunakan struktur data antrian, dan menampilkan urutan simpul yang dikunjungi dalam Bahasa Indonesia. Algoritma ini efisien untuk mencari jalur terpendek dalam graf tidak berbobot dan untuk menjelajahi semua simpul yang dapat dijangkau dari simpul awal.

- **Daftar Pustaka**

  - GeeksforGeeks. (2023). Breadth First Search or BFS for a Graph. Diakses dari https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
  - Khan Academy. (2023). Breadth-First Search Algorithm. Diakses dari https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/breadth-first-search-algorithm
  - Tutorialspoint. (2023). C Program for Breadth First Search. Diakses dari https://www.tutorialspoint.com/c-program-for-breadth-first-search