Laporan JobSheet 7

Dosen Pengampu : Randi Proska Sandra, M.Sc

Kode Kelas : **:** 202323430158

Disusun Oleh :

Hanna Fadilah

23343068

PROGRAM STUDI INFORMATIKA (NK)

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2024

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!

   Source code

```
//created by 23343068_Hanna Fadilah
#include <stdio.h>
#include <stdbool.h>

#define MAX 100

// Struktur untuk representasi graf
struct Graph {
    int V; // Jumlah simpul (vertices)
    int adj[MAX][MAX]; // Matriks adjacency
};

// Inisialisasi graf
void initGraph(struct Graph* G, int V) {
    G->V = V;
    for (int i = 0; i < V; ++i) {
        for (int j = 0; j < V; ++j) {
            G->adj[i][j] = 0;
        }
    }
}

// Menambahkan edge ke graf
void addEdge(struct Graph* G, int src, int dest) {
    G->adj[src][dest] = 1;
    // Untuk graf tak berarah, tambahkan juga edge dari dest
ke src
    G->adj[dest][src] = 1;
}

// BFS traversal dari simpul s dalam graf
void BFS(struct Graph* G, int s) {
    bool visited[MAX] = { false }; // Array untuk menandai
simpul yang sudah dikunjungi
    int queue[MAX]; // Queue untuk menyimpan simpul yang akan
dikunjungi
    int front = -1, rear = -1; // Indeks untuk front dan rear
queue

    visited[s] = true; // Tandai simpul awal sebagai sudah
dikunjungi
    queue[++rear] = s; // Masukkan simpul awal ke queue

    while (front != rear) { // Selama queue belum kosong
        int v = queue[++front]; // Ambil simpul dari queue dan
tandai sebagai dikunjungi
```

```
        printf("%d ", v);

        // Lakukan BFS untuk semua simpul yang bertetangga
dengan simpul v
        for (int i = 0; i < G->V; ++i) {
            if (G->adj[v][i] && !visited[i]) {
                visited[i] = true;
                queue[++rear] = i;
            }
        }
    }
}

int main() {
    struct Graph G;
    int V = 6; // Jumlah simpul
    initGraph(&G, V);

    // Tambahkan edge ke graf
    addEdge(&G, 0, 1);
    addEdge(&G, 0, 2);
    addEdge(&G, 1, 3);
    addEdge(&G, 1, 4);
    addEdge(&G, 2, 4);
    addEdge(&G, 3, 5);
    addEdge(&G, 4, 5);

    printf("Hasil BFS traversal (dimulai dari simpul 0): ");
    BFS(&G, 0); // Lakukan BFS traversal dimulai dari simpul 0

    return 0;
}
```

Penjelasan Algoritma BFS

Algoritma Breadth First Search (BFS) adalah algoritma yang digunakan untuk menemukan jalur terpendek dari satu simpul ke simpul lain dalam graf. Algoritma ini bekerja dengan cara menelusuri semua simpul yang berdekatan dengan simpul awal, kemudian menelusuri semua simpul yang berdekatan dengan simpul-simpul tersebut, dan seterusnya.

Algoritma BFS menggunakan struktur data queue untuk menyimpan simpul-simpul yang akan ditelusuri. Berikut adalah langkah-langkah algoritma BFS:

1. Inisialisasi queue dan visited array.
Queue adalah struktur data yang menyimpan data dengan prinsip First-In-First-Out (FIFO).
Visited array digunakan untuk menandai simpul yang sudah dikunjungi.

2. Tandai simpul awal sebagai visited dan masukkan ke queue.
Simpul awal ditandai sebagai visited karena sudah dikunjungi.
Simpul awal dimasukkan ke queue untuk ditelusuri selanjutnya.

3. Loop selama queue tidak kosong
Loop ini akan terus berjalan selama masih ada simpul yang belum dikunjungi

4. Ambil simpul dari depan queue.
Simpul yang diambil dari queue adalah simpul yang akan ditelusuri selanjutnya.

5. Cetak simpul.
Simpul yang diambil dari queue dicetak sebagai hasil traversal.

6. Telusuri semua tetangga simpul current.
Tetangga simpul current adalah simpul-simpul yang terhubung dengan simpul current.
Setiap tetangga simpul current yang belum dikunjungi akan ditandai sebagai visited dan dimasukkan ke queue.

7. Ulangi langkah 3 sampai 6 sampai queue kosong.
Queue akan kosong ketika semua simpul sudah dikunjungi.
Prinsip Queue dalam Algoritma BFS
Queue digunakan dalam algoritma BFS untuk menyimpan simpul-simpul yang akan ditelusuri. Queue menggunakan prinsip FIFO, sehingga simpul yang pertama kali dimasukkan ke queue akan menjadi simpul yang pertama kali ditelusuri.

Penggunaan queue dalam algoritma BFS memiliki beberapa keuntungan :
Efisiensi: Queue memungkinkan algoritma BFS untuk menelusuri semua simpul dalam graf dengan cara yang efisien.
Sederhana: Penggunaan queue membuat algoritma BFS mudah dipahami dan diimplementasikan.

Berikut adalah beberapa contoh penggunaan queue dalam algoritma BFS:
Menemukan jalur terpendek dalam graf.
Menemukan semua simpul yang terhubung dengan simpul tertentu.
Menemukan semua siklus dalam graf.
Implementasi Algoritma BFS dalam Bahasa C

Berikut adalah penjelasan kode tersebut:

1. Deklarasi struktur data.

Struktur `Graph` digunakan untuk merepresentasikan graf.
Struktur `Queue` digunakan untuk menyimpan simpul-simpul yang akan ditelusuri.

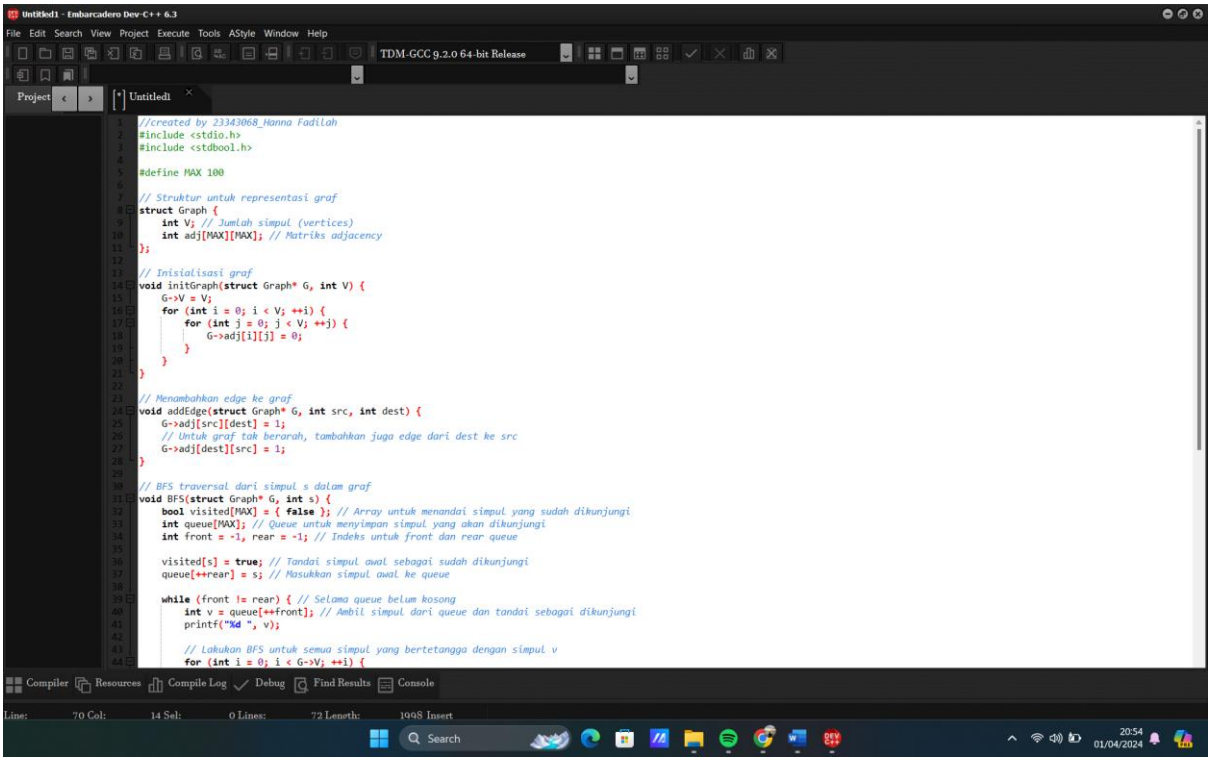
2. Inisialisasi graf dan queue.
Fungsi `initGraph()` digunakan untuk menginisialisasi graf.
Fungsi `initQueue()` digunakan untuk menginisialisasi queue.

3. Menambahkan edge ke graf.
Fungsi `addEdge()` digunakan untuk menambahkan edge ke graf.

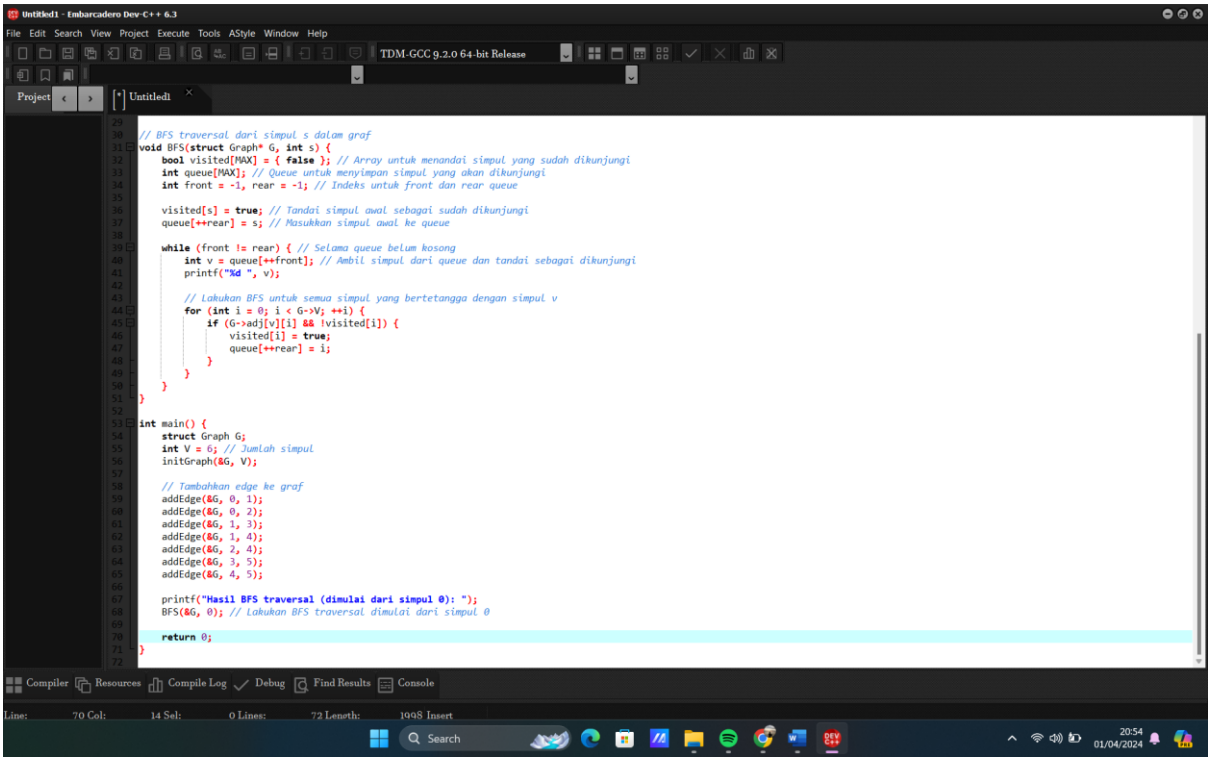
4. Melakukan BFS traversal.
Fungsi `BFS()` digunakan untuk melakukan BFS traversal dari simpul s.

5. Menjalankan program.
Fungsi `main()` digunakan untuk menjalankan program.
Penjelasan kode:

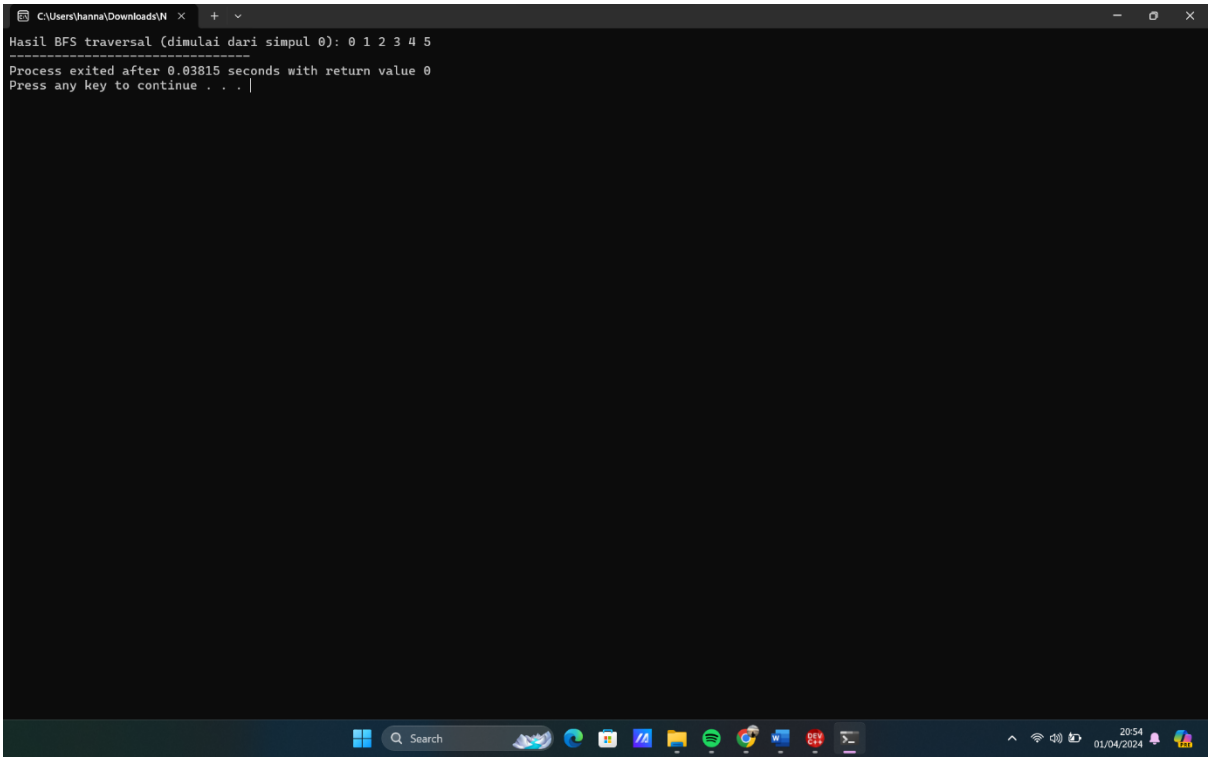
`G.V` : Jumlah simpul dalam graf.
`G.adj[i][j]`: Menunjukkan apakah ada edge antara simpul i dan simpul j.
`visited[i]`: Menunjukkan apakah simpul i sudah dikunjungi.
`Queue` : Queue untuk menyimpan simpul-simpul yang akan ditelusuri.
`front`: Indeks untuk front queue.
`rear`: Indeks untuk rear queue.



```c
//created by 23343068_Hanna Fadilah
#include <stdio.h>
#include <stdbool.h>

#define MAX 100

// Struktur untuk representasi graf
struct Graph {
    int V; // Jumlah simpul (vertices)
    int adj[MAX][MAX]; // Matriks adjacency
};

// Inisialisasi graf
void initGraph(struct Graph* G, int V) {
    G->V = V;
    for (int i = 0; i < V; ++i) {
        for (int j = 0; j < V; ++j) {
            G->adj[i][j] = 0;
        }
    }
}

// Menambahkan edge ke graf
void addEdge(struct Graph* G, int src, int dest) {
    G->adj[src][dest] = 1;
    // Untuk graf tak berarah, tambahkan juga edge dari dest ke src
    G->adj[dest][src] = 1;
}

// BFS traversal dari simpul s dalam graf
void BFS(struct Graph* G, int s) {
    bool visited[MAX] = { false }; // Array untuk menandai simpul yang sudah dikunjungi
    int queue[MAX]; // Queue untuk menyimpan simpul yang akan dikunjungi
    int front = -1, rear = -1; // Indeks untuk front dan rear queue

    visited[s] = true; // Tandai simpul awal sebagai sudah dikunjungi
    queue[++rear] = s; // Masukkan simpul awal ke queue

    while (front != rear) { // Selama queue belum kosong
        int v = queue[++front]; // Ambil simpul dari queue dan tandai sebagai dikunjungi
        printf("%d ", v);

        // Lakukan BFS untuk semua simpul yang bertetangga dengan simpul v
        for (int i = 0; i < G->V; ++i) {
```



```c
// BFS traversal dari simpul s dalam graf
void BFS(struct Graph* G, int s) {
    bool visited[MAX] = { false }; // Array untuk menandai simpul yang sudah dikunjungi
    int queue[MAX]; // Queue untuk menyimpan simpul yang akan dikunjungi
    int front = -1, rear = -1; // Indeks untuk front dan rear queue

    visited[s] = true; // Tandai simpul awal sebagai sudah dikunjungi
    queue[++rear] = s; // Masukkan simpul awal ke queue

    while (front != rear) { // Selama queue belum kosong
        int v = queue[++front]; // Ambil simpul dari queue dan tandai sebagai dikunjungi
        printf("%d ", v);

        // Lakukan BFS untuk semua simpul yang bertetangga dengan simpul v
        for (int i = 0; i < G->V; ++i) {
            if (G->adj[v][i] && !visited[i]) {
                visited[i] = true;
                queue[++rear] = i;
            }
        }
    }
}

int main() {
    struct Graph G;
    int V = 6; // Jumlah simpul
    initGraph(&G, V);

    // Tambahkan edge ke graf
    addEdge(&G, 0, 1);
    addEdge(&G, 0, 2);
    addEdge(&G, 1, 3);
    addEdge(&G, 1, 4);
    addEdge(&G, 2, 4);
    addEdge(&G, 3, 5);
    addEdge(&G, 4, 5);

    printf("Hasil BFS traversal (dimulai dari simpul 0): ");
    BFS(&G, 0); // Lakukan BFS traversal dimulai dari simpul 0

    return 0;
}
```


C:\Users\hanna\Downloads\N
Hasil BFS traversal (dimulai dari simpul 0): 0 1 2 3 4 5

Process exited after 0.03815 seconds with return value 0
Press any key to continue . . .
Search
20:54
01/04/2024