Jobsheet 7

Praktikum Struktur Data

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

Kode Kelas : 202323430158

Disusun Oleh :

Gema Pratama Mahadi Putera
23343066

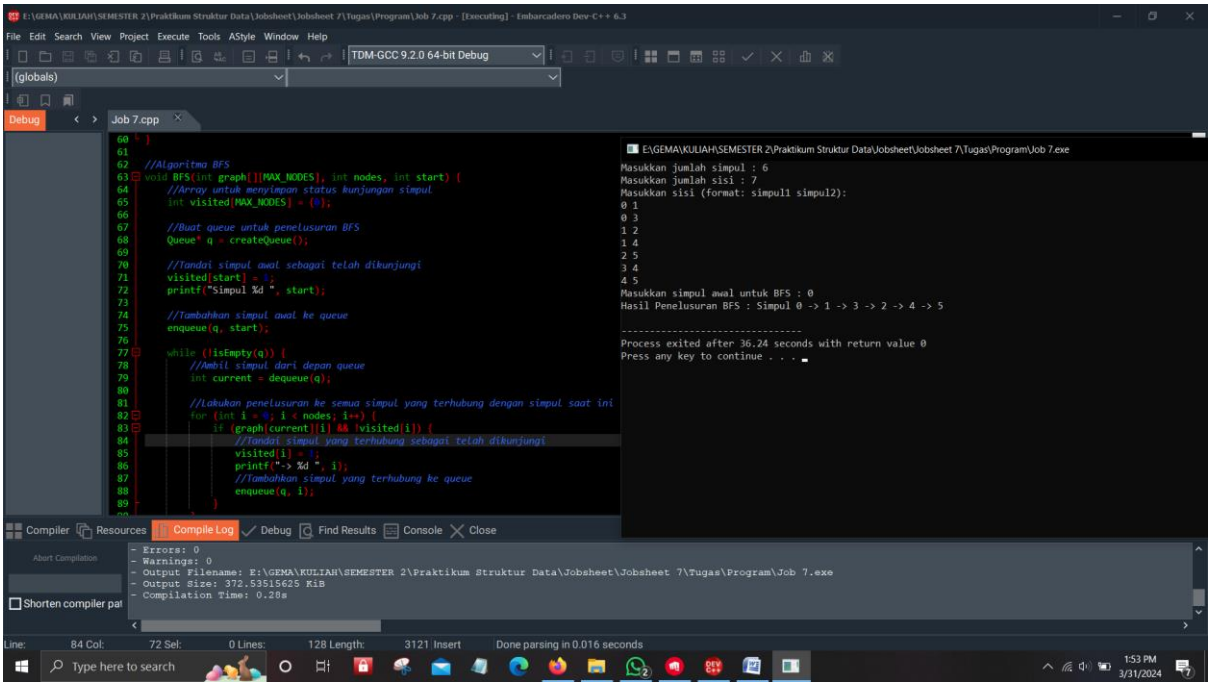
PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

1. **Program :**

```
//Created by 23343066_Gema Pratama

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

//Jumlah maksimal simpul (nodes)
#define MAX_NODES 100

//Struktur simpul (node)
typedef struct Node {
    int value;
    struct Node* next;
} Node;

//Struktur queue
typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;

//Inisialisasi queue
Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = q->rear = NULL;
    return q;
}

//Memeriksa apakah queue kosong
int isEmpty(Queue* q) {
    return (q->front == NULL);
}

//Menambahkan elemen ke queue
void enqueue(Queue* q, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->value = value;
    newNode->next = NULL;
    if (isEmpty(q)) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}

//Menghapus elemen dari queue dan mengembalikan nilainya
int dequeue(Queue* q) {
    if (isEmpty(q)) {
        printf("Antrian kosong\n");
```

```
        return -1;
    } else {
        Node* temp = q->front;
        int value = temp->value;
        q->front = q->front->next;
        if (q->front == NULL)
            q->rear = NULL;
        free(temp);
        return value;
    }
}

//Algoritma BFS
void BFS(int graph[][MAX_NODES], int nodes, int start) {
    //Array untuk menyimpan status kunjungan simpul
    int visited[MAX_NODES] = {0};

    //Buat queue untuk penelusuran BFS
    Queue* q = createQueue();

    //Tandai simpul awal sebagai telah dikunjungi
    visited[start] = 1;
    printf("Simpul %d ", start);

    //Tambahkan simpul awal ke queue
    enqueue(q, start);

    while (!isEmpty(q)) {
        //Ambil simpul dari depan queue
        int current = dequeue(q);

        //Lakukan penelusuran ke semua simpul yang terhubung
dengan simpul saat ini
        for (int i = 0; i < nodes; i++) {
            if (graph[current][i] && !visited[i]) {
                //Tandai simpul yang terhubung sebagai telah
dikunjungi
                visited[i] = 1;
                printf("-> %d ", i);
                //Tambahkan simpul yang terhubung ke queue
                enqueue(q, i);
            }
        }
    }
    printf("\n");
}

int main() {
    int nodes, edges, start;
    int graph[MAX_NODES][MAX_NODES];
```

```
    printf("Masukkan jumlah simpul : ");
    scanf("%d", &nodes);

    printf("Masukkan jumlah sisi : ");
    scanf("%d", &edges);

    //Inisialisasi matriks adjacency
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            graph[i][j] = 0;
        }
    }

    //Masukkan sisi
    printf("Masukkan sisi (format: simpul1 simpul2):\n");
    for (int i = 0; i < edges; i++) {
        int simpul1, simpul2;
        scanf("%d %d", &simpul1, &simpul2);
        graph[simpul1][simpul2] = 1;
        graph[simpul2][simpul1] = 1;
    }

    printf("Masukkan simpul awal untuk BFS : ");
    scanf("%d", &start);

    printf("Hasil Penelusuran BFS : ");
    BFS(graph, nodes, start);

    return 0;
}
```



2. Algoritma Breadth First Search (BFS) :

Algoritma Breadth First Search (BFS) adalah salah satu algoritma penelusuran (traversal) graf yang digunakan untuk menjelajahi atau menemukan semua simpul dalam graf yang terhubung dengan simpul awal tertentu. Prinsip utama dari BFS adalah menelusuri semua simpul yang terhubung dengan simpul awal secara bertahap, dengan memulai dari simpul awal dan menyebar ke simpul-simpul tetangga sebelum menjelajahi simpul-simpul yang lebih jauh. Langkah-langkah utama dalam algoritma BFS:

- **Inisialisasi**: Mulai dari simpul awal, tandai simpul tersebut sebagai sudah dikunjungi dan tambahkan ke dalam queue. Queue digunakan untuk menyimpan simpul-simpul yang telah dikunjungi dan menunggu untuk dieksplorasi lebih lanjut.
- **Eksplorasi**: Lakukan iterasi terus menerus sampai tidak ada lagi simpul yang tersisa dalam queue. Pada setiap iterasi, ambil simpul dari depan queue dan periksa semua simpul yang terhubung langsung dengan simpul tersebut.
- **Enqueue**: Jika ada simpul yang belum pernah dikunjungi dan terhubung langsung dengan simpul yang sedang dieksplorasi, tandai simpul tersebut sebagai sudah dikunjungi dan tambahkan ke dalam queue. Hal ini dilakukan untuk memastikan simpul-simpul tersebut akan dieksplorasi setelah simpul-simpul yang lebih dekat telah ditelusuri.
- **Dequeue**: Setelah semua simpul yang terhubung dengan simpul saat ini telah diperiksa, simpul tersebut dihapus dari queue.
- **Ulangi**: Proses di atas diulangi sampai semua simpul yang terhubung dengan simpul awal telah dikunjungi.

Algoritma BFS akan terus berlanjut hingga semua simpul yang terhubung telah dikunjungi. Selain itu, algoritma BFS menggunakan struktur data queue untuk mengatur urutan penelusuran simpul-simpul. Queue memastikan bahwa simpul-simpul yang dimasukkan pertama kali juga dikeluarkan pertama kali (konsep FIFO - First In First Out). Dengan menggunakan queue, algoritma BFS dapat memastikan bahwa simpul-simpul yang lebih dekat dengan simpul awal akan dieksplorasi terlebih dahulu sebelum simpul-simpul yang lebih jauh. Hal ini memungkinkan algoritma BFS untuk menemukan jalur terpendek antara dua simpul dalam graf tidak berbobot dan juga memungkinkan untuk melakukan pencarian jarak terpendek dalam graf berbobot yang setara dengan BFS.

3. Prinsip Queue :

Dalam algoritma BFS, prinsip queue digunakan untuk mengatur urutan penelusuran simpul-simpul. Queue memungkinkan algoritma untuk mengakses simpul-simpul yang belum dieksplorasi secara terurut, yaitu dengan cara memasukkan simpul ke dalam queue saat simpul tersebut dikunjungi untuk pertama kali, dan kemudian mengeluarkan simpul tersebut dari queue setelah semua simpul yang terhubung dengannya telah diperiksa. Cara prinsip queue digunakan dalam algoritma BFS:

- **Menambahkan ke Queue**: Saat sebuah simpul dikunjungi, simpul tersebut dimasukkan ke dalam queue. Pada awalnya, simpul awal dimasukkan ke dalam queue. Setelah itu, setiap kali sebuah simpul terhubung langsung dengan simpul yang sedang dieksplorasi, simpul tersebut juga dimasukkan ke dalam queue jika belum pernah dikunjungi sebelumnya.
- **Menghapus dari Queue**: Setelah semua simpul yang terhubung dengan simpul yang sedang dieksplorasi telah diperiksa, simpul tersebut dihapus dari queue. Dengan menggunakan prinsip FIFO (First In First Out), simpul yang dimasukkan pertama kali ke dalam queue akan dikeluarkan terlebih dahulu.
- **Menjaga Urutan**: Queue memastikan bahwa simpul-simpul yang lebih dekat dengan simpul awal akan dieksplorasi terlebih dahulu sebelum simpul-simpul yang lebih jauh. Hal ini memungkinkan algoritma BFS untuk menemukan jalur terpendek antara dua simpul dalam graf tidak berbobot dan juga memungkinkan untuk melakukan pencarian jarak terpendek dalam graf berbobot yang setara dengan BFS.

Dengan menggunakan struktur data queue, algoritma BFS dapat memastikan bahwa penelusuran dilakukan secara sistematis dan terurut, sehingga semua simpul yang terhubung dengan simpul awal akan dieksplorasi dengan benar dan tidak ada simpul yang terlewatkan.