**LAPORAN JOBSHEET 7**

**“Queue / antrian”**

****

**Dosen pengampu :**

**Randi Proska Sandra, M.Sc.**

**NIP. 221048**

**Disusun oleh :**

**Muhaammad Devin Rahadi**

**2023/23343076**

**INFORMATIKA**

**JURUSAN ELEKTRONIKA**

**FAKULTAS TEKNNIK**

**UNIVERSITAS NEGERI PADANG**

**2024**

Kode progam :

/\* Nama : MUHAMMAD DEVIN RAHADI

NIM : 23343076

PRODI : INFORMATIKA

KELAS : 202323430157, SELASA 08:50 - 10:30

\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

// Struktur untuk queue

typedef struct {

int items[MAX];

int front;

int rear;

} Queue;

// Fungsi untuk membuat queue baru

Queue\* buatQueue() {

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->front = -1;

q->rear = -1;

return q;

}

// Fungsi untuk mengecek apakah queue kosong

int isEmpty(Queue\* q) {

return q->front == -1;

}

// Fungsi untuk menambah elemen ke dalam queue

void enqueue(Queue\* q, int value) {

if (q->rear == MAX - 1) {

printf("Queue penuh!\n");

return;

} else {

if (q->front == -1)

q->front = 0;

q->rear++;

q->items[q->rear] = value;

}

}

// Fungsi untuk menghapus elemen dari queue

int dequeue(Queue\* q) {

int item;

if (isEmpty(q)) {

printf("Queue kosong!\n");

item = -1;

} else {

item = q->items[q->front];

q->front++;

if (q->front > q->rear) {

q->front = q->rear = -1;

}

}

return item;

}

// Fungsi untuk BFS

void BFS(int adjMatrix[MAX][MAX], int startVertex, int numVertices) {

Queue\* q = buatQueue();

int visited[MAX] = {0};

visited[startVertex] = 1;

enqueue(q, startVertex);

printf("Hasil BFS: ");

while (!isEmpty(q)) {

int currentVertex = dequeue(q);

printf("%d ", currentVertex);

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

if (adjMatrix[currentVertex][i] == 1 && !visited[i]) {

visited[i] = 1;

enqueue(q, i);

}

}

}

printf("\n");

}

int main() {

int numVertices = 6;

int adjMatrix[MAX][MAX] = {

{0, 1, 1, 0, 0, 0},

{1, 0, 0, 1, 1, 0},

{1, 0, 0, 0, 0, 1},

{0, 1, 0, 0, 0, 0},

{0, 1, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0}

};

BFS(adjMatrix, 0, numVertices);

return 0;

}

**Penjelasan Algoritma Breadth First Search (BFS)**

Algoritma Breadth First Search (BFS) digunakan untuk menelusuri atau mencari semua simpul dari sebuah graf (graph) secara sistematis. BFS mulai dari sebuah simpul awal dan menjelajahi tetangganya terlebih dahulu sebelum bergerak ke simpul-simpul lain. Prinsip kerja BFS adalah menggunakan struktur data antrian (queue) untuk melacak simpul-simpul yang akan dieksplorasi selanjutnya.

**Prinsip Queue dalam BFS**

1. **Inisialisasi**:

Simpul awal ditandai sebagai dikunjungi dan dimasukkan ke dalam queue.

1. **Proses**:

Selama queue tidak kosong:

* + - Ambil (dequeue) simpul dari depan queue.
    - Periksa semua simpul yang terhubung (tetangga) dengan simpul saat ini.
    - Jika simpul tetangga belum dikunjungi, tandai sebagai dikunjungi, dan masukkan (enqueue) simpul tersebut ke dalam queue.

1. **Pencetakan**:

Cetak atau proses simpul yang sedang di-dequeue.

**Langkah-langkah Detail**

1. **Inisialisasi Queue dan Status Dikunjungi**:
   * Buat queue kosong untuk menyimpan simpul yang akan dieksplorasi.
   * Buat array visited untuk menandai simpul yang sudah dikunjungi.
2. **Mulai dari Simpul Awal**:
   * Tandai simpul awal sebagai dikunjungi.
   * Masukkan simpul awal ke dalam queue.
3. **Proses Queue**:

Selama queue tidak kosong, ambil simpul dari depan queue dan proses:

* + - Periksa semua tetangga dari simpul saat ini.
    - Jika tetangga belum dikunjungi, tandai sebagai dikunjungi, dan masukkan ke dalam queue.

1. **Cetak Hasil**:
   * Cetak simpul yang sedang diproses saat dikeluarkan dari queue.

**Penjelasan Implementasi**

1. **Struktur Queue**:

Queue digunakan untuk menyimpan simpul-simpul yang akan dieksplorasi. Queue diimplementasikan dengan array dan dua variabel front dan rear untuk melacak elemen pertama dan terakhir dalam queue.

1. **Fungsi buatQueue**:

Membuat queue baru dan menginisialisasi front dan rear ke -1 untuk menandakan bahwa queue kosong.

1. **Fungsi isEmpty**:

Mengecek apakah queue kosong dengan memeriksa apakah front adalah -1.

1. **Fungsi enqueue**:

Menambahkan elemen ke dalam queue. Jika queue penuh, fungsi akan mencetak pesan kesalahan. Jika queue kosong, front diatur ke 0.

1. **Fungsi dequeue**:

Menghapus elemen dari depan queue. Jika queue kosong, fungsi akan mencetak pesan kesalahan dan mengembalikan -1. Jika front lebih besar dari rear, queue diatur ulang ke kondisi kosong.

1. **Fungsi BFS**:
   * Mengimplementasikan algoritma BFS. Menerima matriks ketetanggaan, simpul awal, dan jumlah simpul sebagai parameter.
   * Memulai dari simpul awal, menandai sebagai dikunjungi, dan menambahkannya ke queue.
   * Selama queue tidak kosong, simpul di-dequeue dan semua tetangga yang belum dikunjungi dimasukkan ke dalam queue setelah ditandai sebagai dikunjungi.
2. **Fungsi main**:

Menyusun graf menggunakan matriks ketetanggaan dan memanggil fungsi BFS dengan simpul awal 0.

HASIL PROGAM

