

**JOB SHEET 7**  
**Praktikum Struktur Data**  
**Algoritma Breadth First Search (BFS)**



**DOSEN PENGAMPU :**

**Randi Proska Sandra, M.Sc**

**DISUSUN OLEH :**

**Humaira Mutia**  
**23343069**

**PROGRAM STUDI INFORMATIKA (NK)**

**FAKULTAS TEKNIK**

**2023**



## A. Source Code

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

// Definisi struktur untuk Graf
struct Graf {
    int jumlahSimpul;
    int** daftarAdj;
    int* dikunjungi;
};

// Fungsi untuk membuat graf baru
struct Graf* buatGraf(int jumlahSimpul) {
    struct Graf* graf = (struct Graf*)malloc(sizeof(struct Graf));
    graf->jumlahSimpul = jumlahSimpul;

    graf->daftarAdj = (int**)malloc(jumlahSimpul * sizeof(int*));
    for (int i = 0; i < jumlahSimpul; i++) {
        graf->daftarAdj[i] = (int*)malloc(jumlahSimpul * sizeof(int));
        for (int j = 0; j < jumlahSimpul; j++) {
            graf->daftarAdj[i][j] = 0;
        }
    }

    graf->dikunjungi = (int*)malloc(jumlahSimpul * sizeof(int));
    for (int i = 0; i < jumlahSimpul; i++) {
        graf->dikunjungi[i] = 0;
    }

    return graf;
}

// Fungsi untuk menambahkan sisi pada graf
void tambahSisi(struct Graf* graf, int asal, int tujuan) {
    graf->daftarAdj[asal][tujuan] = 1;
    graf->daftarAdj[tujuan][asal] = 1; // Jika graf tidak berarah
}

// Fungsi untuk melakukan Breadth-First Search (BFS)
void BFS(struct Graf* graf, int simpulAwal) {
    int* antrian = (int*)malloc(jumlahSimpul * sizeof(int));
    int depan = 0;
    int belakang = 0;

    graf->dikunjungi[simpulAwal] = 1;
    antrian[belakang++] = simpulAwal;

    while (depan < belakang) {
        int simpulSaatIni = antrian[depan++];
        printf("Dikunjungi %d\n", simpulSaatIni);

        for (int i = 0; i < graf->jumlahSimpul; i++) {
            if (graf->daftarAdj[simpulSaatIni][i] == 1 && graf->dikunjungi[i] == 0) {
                graf->dikunjungi[i] = 1;
                antrian[belakang++] = i;
            }
        }
    }
}
```

```

        free(antrian);
    }

int main() {
    int jumlahSimpul = 6;
    struct Graf* graf = buatGraf(jumlahSimpul);

    tambahSisi(graf, 0, 1);
    tambahSisi(graf, 0, 2);
    tambahSisi(graf, 1, 2);
    tambahSisi(graf, 1, 3);
    tambahSisi(graf, 2, 4);
    tambahSisi(graf, 3, 4);
    tambahSisi(graf, 3, 5);
    tambahSisi(graf, 4, 5);

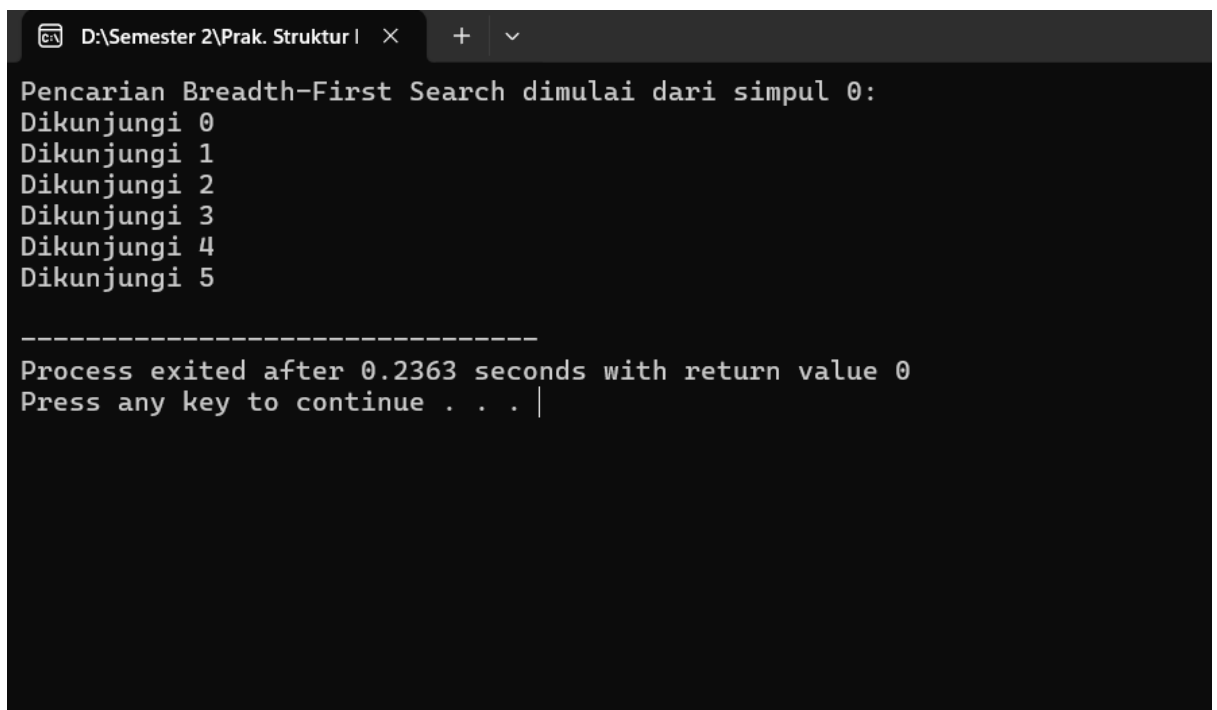
    printf("Pencarian Breadth-First Search dimulai dari simpul 0:\n");
    BFS(graf, 0);

    for (int i = 0; i < jumlahSimpul; i++) {
        free(graf->daftarAdj[i]);
    }
    free(graf->daftarAdj);
    free(graf->dikunjungi);
    free(graf);

    return 0;
}

```

## B. Output Program



```

D:\Semester 2\Prak. Struktur I
Pencarian Breadth-First Search dimulai dari simpul 0:
Dikunjungi 0
Dikunjungi 1
Dikunjungi 2
Dikunjungi 3
Dikunjungi 4
Dikunjungi 5

-----
Process exited after 0.2363 seconds with return value 0
Press any key to continue . . .

```

### C. Penjelasan

**Algoritma Breadth-First Search (BFS)** digunakan untuk mencari jalur terpendek atau penelusuran dalam graf dari simpul awal ke simpul lainnya. Dalam konteks ini, algoritma ini membantu menemukan lintasan terpendek atau jalur terpendek dari simpul 0 ke simpul lain dalam graf. Dengan menggunakan BFS, kita dapat menemukan jalur terpendek dari simpul awal ke simpul tujuan, yang berguna dalam berbagai aplikasi seperti pencarian jalur terpendek dalam peta, perutean jaringan komputer, atau penelusuran dalam struktur data graf yang kompleks. Dengan menelusuri simpul-simpul secara berurutan berdasarkan jaraknya dari simpul awal, BFS memastikan bahwa simpul-simpul yang lebih dekat dengan simpul awal akan dikunjungi terlebih dahulu sebelum menjelajahi simpul-simpul yang lebih jauh.

**Prinsip queue** yang digunakan dalam program ini adalah prinsip First In First Out (FIFO). Queue digunakan untuk melacak simpul-simpul yang akan dikunjungi berikutnya selama proses Breadth-First Search (BFS). Berikut adalah penjelasan prinsip queue yang digunakan dalam program:

- **Inisialisasi Antrian:**

Sebuah array dinamis antrian digunakan untuk menyimpan nomor simpul yang akan dikunjungi.

Variabel depan dan belakang digunakan untuk menunjukkan posisi depan dan belakang antrian.

- **Enqueue (Memasukkan ke Antrian):**

Saat sebuah simpul dikunjungi, nomor simpul tersebut dimasukkan ke dalam array antrian di posisi belakang.

Kemudian, posisi belakang diperbarui untuk menunjukkan simpul baru yang telah dimasukkan.

- **Dequeue (Mengeluarkan dari Antrian):**

Saat proses BFS berlanjut, simpul yang telah dikunjungi akan dikeluarkan dari depan antrian (depan).

Simpul yang dikeluarkan ini kemudian diproses dan tetangga-tetangganya yang belum dikunjungi dimasukkan kembali ke dalam antrian.

- **Iterasi Sampai Antrian Kosong:**

Proses enqueue dan dequeue berulang terus menerus sampai antrian kosong, yang menandakan bahwa semua simpul yang dapat dijangkau dari simpul awal telah dikunjungi.

Program ini akan menampilkan simpul-simpul yang dikunjungi dan mencetak urutan traversal dari simpul awal ke simpul lainnya.