

TUGAS JOBSHEET 7

PRAKTIKUM STRUKTUR DATA

Oleh : Nurul Aldi (23343048)

Program Tentang algoritma Breadth First Search (BFS) dalam bahasa C, serta penjelasan dan bagaimana prinsip queue digunakan dalam program tersebut :

Source Code :

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

// sebuah struct untuk node pada list yang berdekatan
struct node {
    int vertex;
    struct node* next;
};

// sebuah struct untuk list yang berdekatan
struct adj_list {
    struct node* head;
};

// sebuah struct untuk graph
struct graph {
    int num_vertices;
```

```

    struct adj_list* adj_lists;
    int* visited;
};

// buat sebuah node baru pada list yang berdekatan
struct node* new_node(int vertex) {
    struct node* new_node = (struct node*)malloc(sizeof(struct node));
    new_node->vertex = vertex;
    new_node->next = NULL;
    return new_node;
}

// buat sebuah graph dengan n vertex
struct graph* create_graph(int n) {
    struct graph* graph = (struct graph*)malloc(sizeof(struct graph));
    graph->num_vertices = n;
    graph->adj_lists = (struct adj_list*)malloc(n * sizeof(struct
adj_list));
    graph->visited = (int*)malloc(n * sizeof(int));

    int i;
    for (i = 0; i < n; i++) {
        graph->adj_lists[i].head = NULL;
        graph->visited[i] = 0;
    }

    return graph;
}

// tambahkan sebuah edge kepada graph
void add_edge(struct graph* graph, int src, int dest) {
    // tambah sebuah edge dari src ke dest

```

```

    struct node* new_node1 = new_node(dest);
    new_node1->next = graph->adj_lists[src].head;
    graph->adj_lists[src].head = new_node1;

// tambahkan sebuah edge dari dest ke src
    struct node* new_node2 = new_node(src);
    new_node2->next = graph->adj_lists[dest].head;
    graph->adj_lists[dest].head = new_node2;
}

// melintasi graph menggunakan algoritma bfs dimulai dari vertex v
void bfs(struct graph* graph, int v) {
    // buat sebuah queue untuk BFS
    int queue[1000];
    int front = -1;
    int rear = -1;

    // tandai node saat ini sebagai "visited"/dikunjungi dan di-enqueue
    graph->visited[v] = 1;
    queue[++rear] = v;

// kunjungi semua vertex di graph
    while (front != rear) {
        // dequeue vertex dari queue dan tampilkan
        int current_vertex = queue[++front];
        printf("%d ", current_vertex);

        // dapatkan semua tetangga dari vertex yang di-dequeue
        struct node* temp = graph->adj_lists[current_vertex].head;
        while (temp != NULL) {
            int adj_vertex = temp->vertex;

```

```
        // jika "tetangganya" belum dikunjungi, maka tandai sebagai
        dikunjungi/visited dan di-enqueue
```

```
        if (graph->visited[adj_vertex] == 0) {
            graph->visited[adj_vertex] = 1;
            queue[++rear] = adj_vertex;
        }
```

```
        temp = temp->next;
```

```
    }
```

```
}
```

```
}
```

```
int main() {
```

```
    // Membuat sebuah graph dengan 6 vertex
```

```
    struct graph* graph = create_graph(6);
```

```
    // menambahkan edges kepada graph
```

```
    add_edge(graph, 0, 1);
```

```
    add_edge(graph, 0, 2);
```

```
    add_edge(graph, 1, 3);
```

```
    add_edge(graph, 1, 4);
```

```
    add_edge(graph, 2, 4);
```

```
    add_edge(graph, 3, 4);
```

```
    add_edge(graph, 3, 5);
```

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

```
    // algoritma bfs dilakukan dimulai dari vertex 0
```

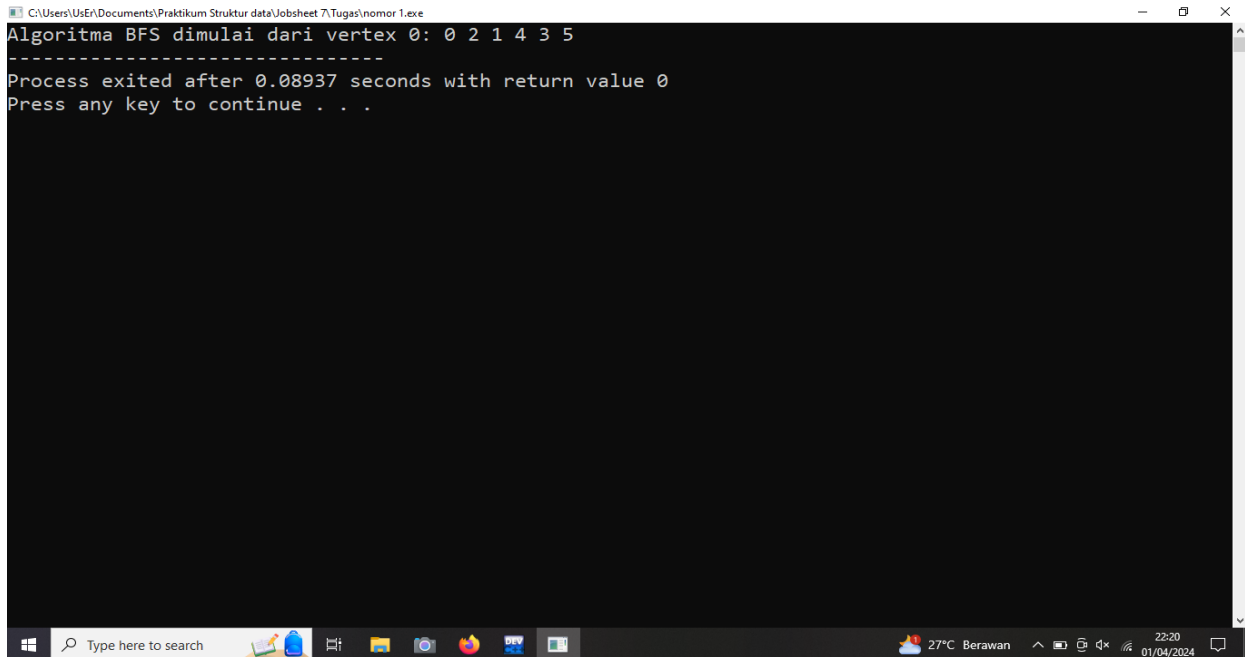
```
    printf("Algoritma BFS dimulai dari vertex 0: ");
```

```
    bfs(graph, 0);
```

```
    return 0;
```

```
}
```

Screenshot Output :



```
C:\Users\UsE\Documents\Praktikum Struktur data\Jobsheet 7\Tugas\nomor 1.exe
Algoritma BFS dimulai dari vertex 0: 0 2 1 4 3 5
-----
Process exited after 0.08937 seconds with return value 0
Press any key to continue . . .
```

Penjelasan :

Kode program di atas membuat sebuah graph dengan 6 vertex dan menambahkan edges diantara vertex – vertex tersebut. Setelah itu, Program menerapkan algoritma BFS untuk melintasi graph dimulai dari vertex 0 dan menampilkan urutan vertex yang dikunjungi.

Pada Program ini, 'struct node' merepresentasikan sebuah node pada list yang berdekatan, 'struct adj_list' merepresentasikan list yang berdekatan, dan 'struct graph' merepresentasikan keseluruhan graph. Fungsi 'create_graph' membuat sebuah graph dengan 'n' vertex, dan fungsi 'add_edge' menambahkan edge diantara dua vertex.

Fungsi 'bfs' mengimplementasikan algoritma BFS, mulai dari vertex 'v'. **Sebuah QUEUE dan array yang telah dikunjungi digunakan untuk menandai vertex mana yang harus dikunjungi selanjutnya.** Fungsi 'main' membuat sebuah graph dan menambahkan beberapa edges, kemudian memanggil fungsi 'bfs' mulai dari vertex 0

