

Laporan Jobsheet 7



Dosen pengampu : Randi Proska Sandra, M.Sc

Kode Kelas : 202323430158

Disusun Oleh :

**M.Gilang Mulya Putra
23343073**

**PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024**

Pengantar

Mata kuliah Praktek Struktur Data merupakan salah satu mata kuliah yang wajib diambil oleh mahasiswa teknik informatika. Dalam mata kuliah ini, kita akan belajar mengenai konsep-konsep dasar dari struktur data, seperti pointer, struct, dan array. Selain itu, kita juga akan mempelajari tentang beberapa jenis linked list, yaitu link list, double link list, dancircular link list. Berikut adalah ringkasan mengenai materi-materi tersebut.

Pointer, Struct, dan Array: Pada dasarnya, pointer, struct, dan array adalah konsep-konsep dasar dari bahasa pemrograman C. Pada praktik struktur data, kita akan mempelajari cara menggunakannya dalam membuat struktur data yang lebih kompleks.

Source Code

```
//created by M.Gilang Mulya Putra_23343073
#include <stdio.h>
#include <stdlib.h>

#define MAX_VERTICES 100

struct Queue {
    int items[MAX_VERTICES];
    int front;
    int rear;
};

struct Graph {
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
    int numVertices;
};

void initGraph(struct Graph *G, int numVertices);
void addEdge(struct Graph *G, int src, int dest);
void BFS(struct Graph *G, int startVertex);
void enqueue(struct Queue *queue, int value);
int dequeue(struct Queue *queue);
int isEmpty(struct Queue *queue);

void initGraph(struct Graph *G, int numVertices) {
    G->numVertices = numVertices;
    for (int i = 0; i < numVertices; ++i) {
        for (int j = 0; j < numVertices; ++j) {
            G->adjMatrix[i][j] = 0;
        }
    }
}
```

```
}
```

```
void addEdge(struct Graph *G, int src, int dest) {  
    G->adjMatrix[src][dest] = 1;  
    //G->adjMatrix[dest][src] = 1; // Tidak diaktifkan untuk graf  
    berarah  
}
```

```
void BFS(struct Graph *G, int startVertex) {  
    struct Queue *queue = (struct Queue*)malloc(sizeof(struct  
Queue));  
    queue->front = -1;  
    queue->rear = -1;  
  
    int visited[MAX_VERTICES] = {0};  
    visited[startVertex] = 1;  
    enqueue(queue, startVertex);  
  
    while (!isEmpty(queue)) {  
        int currentVertex = dequeue(queue);  
        printf("%d ", currentVertex);  
  
        for (int i = 0; i < G->numVertices; ++i) {  
            if (G->adjMatrix[currentVertex][i] == 1 && !visited[i]) {  
                visited[i] = 1;  
                enqueue(queue, i);  
            }  
        }  
    }  
  
    free(queue);  
}
```

```

void enqueue(struct Queue *queue, int value) {
    if (queue->rear == MAX_VERTICES - 1)
        printf("\nQueue is Full!!");
    else {
        if (queue->front == -1) queue->front = 0;
        queue->rear++;
        queue->items[queue->rear] = value;
    }
}

```

```

int dequeue(struct Queue *queue) {
    int item;
    if (queue->front == -1)
        printf("\nQueue is Empty!!");
    else {
        item = queue->items[queue->front];
        queue->front++;
        if (queue->front > queue->rear) {
            queue->front = queue->rear = -1;
        }
        return item;
    }
}

```

```

int isEmpty(struct Queue *queue) {
    if (queue->front == -1)
        return 1;
    else
        return 0;
}

```

```

int main() {

```

```

struct Graph G;

int numVertices = 6;

initGraph(&G, numVertices);

addEdge(&G, 0, 1);
addEdge(&G, 0, 2);
addEdge(&G, 1, 3);
addEdge(&G, 1, 4);
addEdge(&G, 2, 4);
addEdge(&G, 3, 4);
addEdge(&G, 3, 5);
addEdge(&G, 4, 5);

printf("Breadth First Search starting from vertex 0: ");
BFS(&G, 0);

return 0;
}

```

The screenshot shows a C++ IDE with the following code in `game.cpp`:

```

1 //Created by N.Gilang Nulya Putra_23343073
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define MAX_VERTICES 100
5
6 struct Queue {
7
8 }
9
10 int main() {
11     struct Graph G;
12     int numVertices = 6;
13     initGraph(&G, numVertices);
14
15     addEdge(&G, 0, 1);
16     addEdge(&G, 0, 2);
17     addEdge(&G, 1, 3);
18     addEdge(&G, 1, 4);
19     addEdge(&G, 2, 4);
20     addEdge(&G, 3, 4);
21     addEdge(&G, 3, 5);
22     addEdge(&G, 4, 5);
23
24     printf("Breadth First Search starting from vertex 0: ");
25     BFS(&G, 0);
26
27     return 0;
28 }

```

The output of the program is displayed in a terminal window:

```

Breadth First Search starting from vertex 0: 0 1 2 3 4 5
-----
Process exited after 0.03333 seconds with return value 0
Press any key to continue . . .

```

Penjelasan Program

Algoritma ini adalah algoritma Breadth First Search (BFS) untuk menjelajahi atau mencari

jalur dalam sebuah graf. BFS dimulai dari suatu simpul tertentu (dalam contoh ini, simpul 0) dan secara bertahap menjelajahi simpul-simpul terdekat sebelum menjelajahi simpul-simpul yang lebih jauh.

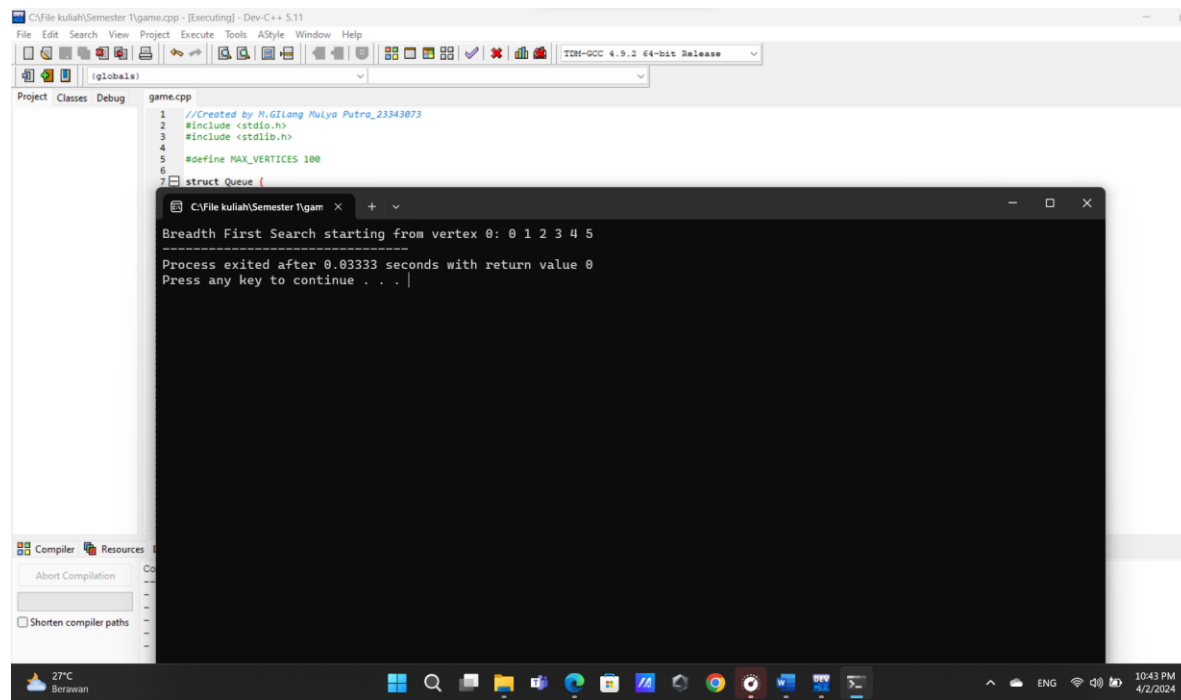
Prinsip dasar BFS adalah menggunakan antrian (queue) untuk menyimpan simpul-simpul yang akan dieksplorasi. Algoritma tersebut bekerja sebagai berikut:

1. Dimulai dengan menginisialisasi sebuah graf dan antrian kosong.
2. Simpul awal dimasukkan ke dalam antrian dan ditandai sebagai sudah dikunjungi.
3. Selama antrian tidak kosong, lakukan langkah-langkah berikut:
 - Keluarkan simpul pertama dari antrian.
 - Cetak simpul tersebut (atau lakukan operasi yang diinginkan).
 - Untuk setiap simpul yang bertetangga dengan simpul yang diambil dari antrian, jika simpul tersebut belum dikunjungi, tandai simpul tersebut sebagai sudah dikunjungi dan masukkan ke dalam antrian.
4. Ulangi langkah-langkah ini sampai antrian kosong.

Prinsip antrian digunakan dalam algoritma ini karena kita ingin menjelajahi simpul-simpul terdekat terlebih dahulu sebelum menjelajahi simpul-simpul yang lebih jauh. Dengan menggunakan antrian, simpul-simpul baru yang ditemukan akan ditambahkan ke belakang antrian, sehingga simpul-simpul tersebut akan dieksplorasi setelah simpul-simpul yang ada dalam antrian dieksplorasi terlebih dahulu.

Dalam implementasi kode yang diberikan, struktur data antrian (queue) digunakan untuk menyimpan simpul-simpul yang akan dieksplorasi. Fungsi `enqueue` digunakan untuk menambahkan simpul ke dalam antrian, sedangkan fungsi `dequeue` digunakan untuk mengeluarkan simpul dari antrian. Selain itu, terdapat fungsi `isEmpty` untuk memeriksa apakah antrian kosong atau tidak. Dengan menggunakan prinsip ini, algoritma BFS dapat secara efisien menjelajahi graf dengan urutan yang sesuai.

Output



The screenshot shows a C++ IDE with a file named `game.cpp` open. The code includes standard headers and a queue structure. The output window displays the execution results of a Breadth First Search algorithm.

```
1 //Created by M.Gilang Mulya Putra_23343073
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 #define MAX_VERTICES 100
6
7 struct Queue {
```

Output:

```
Breadth First Search starting from vertex 0: 0 1 2 3 4 5
-----
Process exited after 0.03333 seconds with return value 0
Press any key to continue . . . |
```

The IDE interface includes a menu bar (File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help), a toolbar, a project explorer showing a project named `(global)`, and a compiler panel at the bottom left with an `Abort Compilation` button and a checkbox for `Shorten compiler paths`. The Windows taskbar at the bottom shows the system clock as 10:43 PM on 4/2/2024.

Referensi

- <https://www.guru99.com/breadth-first-search-bfs-graph-example.html>
- <https://www.interviewcake.com/concept/java/bfs>
- <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>
- https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm
- <https://www.perplexity.ai/search/created-by-fhandy-OlpDpf09RFy90vsmYo6AFg>