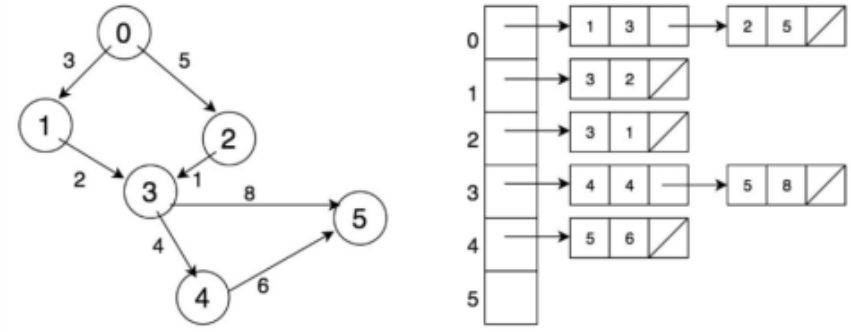
Nama : Yanuar Nurul Hilal
NIM : 222112418
Kelas : 2KS4

Penugasan Struktur Data Praktikum 13

1. Buatlah program untuk untuk graph berarah dan berbobot dan representasinya berikut ini:




```c
#include <stdio.h>
#include <stdlib.h>
#define N 6 // misal maksimum node adalah 6

// Struktur data untuk menyimpan adjacency list dari  node pada graph
struct Node
{
    int dest, weight;
    struct Node *next;
};
typedef struct Node *ptrNode;

// Struktur data untuk menyimpan onjek graph
struct Graph
{
    // array pointer ke node untuk representasi adjacency list
    ptrNode head[N];
};
typedef struct Graph *ptrGraph;

// Struktur data untuk menyimpan edge graph
struct Edge
{
    int src, dest, weight;
};
```

```c
// Fungsi untuk membuat adjacency list dari edge tertentu
ptrGraph createGraph(struct Edge edges[], int n)
{
    // alokasi memori untuk menyimpan struktur data graph
    ptrGraph graph = (ptrGraph)malloc(sizeof(struct Graph));

    // inisialiasi semua pointer head ke null
    for (int i = 0; i < N; i++)
    {
        graph->head[i] = NULL;
    }

    // menambahkan edge satu demi satu
    for (int i = 0; i < n; i++)
    {
        // ambil source dan destination dari node
        int src = edges[i].src;
        int dest = edges[i].dest;
        int weight = edges[i].weight;

        // buat node baru dari src ke dest
        ptrNode newNode = (ptrNode)malloc(sizeof(struct Node));
        newNode->dest = dest;
        newNode->weight = weight;

        // point node baru ke head
        newNode->next = graph->head[src];

        // point head ke node baru
        graph->head[src] = newNode;
    }

    return graph;
}

// Fungsi print representasi adjacency list
void printGraph(ptrGraph graph)
{
    int i;
    for (i = 0; i < N; i++)
    {
        // print node dan semua yang terhubung
```

```
        ptrNode ptr = graph->head[i];
        while (ptr != NULL)
        {
            printf("%d -> %d (%d)\t", i, ptr->dest, ptr->weight);
            ptr = ptr->next;
        }

        printf("\n");
    }
}

void main()
{
    // input array pasangan dari x ke y
    struct Edge edges[] ={{0, 1, 3}, {0, 2, 5}, {1, 3, 2}, {2, 3, 1}, {3,
5, 8}, {3, 4, 4}, {4, 5, 6}};

    // menghitung jumlah edge
    int n = sizeof(edges) / sizeof(edges[0]);

    // membuat graph
    ptrGraph graph = createGraph(edges, n);

    // print graph
    printGraph(graph);
}
```

2. Buat program untuk contoh representasi graph tak berarah.

```
ptrGraph createGraph(struct Edge edges[], int n)
{
    // alokasi memori untuk menyimpan struktur data graph
    ptrGraph graph = (ptrGraph)malloc(sizeof(struct Graph));

    // inisialiasi semua pointer head ke null
    for (int i = 0; i < N; i++)
    {
        graph->head[i] = NULL;
    }

    // menambahkan edge satu demi satu
```

```
    for (int i = 0; i < n; i++)
    {
        // ambil source dan destination dari node
        int src = edges[i].src;
        int dest = edges[i].dest;
        int weight = edges[i].weight;

        // buat node baru dari src ke dest
        ptrNode newNode = (ptrNode)malloc(sizeof(struct Node));
        newNode->dest = dest;
        newNode->weight = weight;

        // point node baru ke head
        newNode->next = graph->head[src];

        // point head ke node baru
        graph->head[src] = newNode;

        //Tambahkan ini untuk undirected
        //Undirected Graph selalu 2 arah sehingga kalau dalam bentuk
matriks mereka bakal simetris

        // buat node baru dari dest ke src
        newNode = (ptrNode)malloc(sizeof(struct Node));
        newNode->dest = src;
        newNode->weight = weight;

        // point node baru ke head
        newNode->next = graph->head[dest];

        // point head ke node baru
        graph->head[dest] = newNode;
    }

    return graph;
}
```