SOAL PRAKTIKUM KOM120H STRUKTUR DATA
PERTEMUAN 9 APLIKASI *GRAPH TRAVERSAL*

Dengan representasi *graph* menggunakan *adjacency matrix* seperti pada pertemuan sebelumnya, pada pertemuan ini kita akan membuat implementasi dari deteksi *cycle* dan *topological sort*.

Potongan program di bawah ini mengimplementasikan algoritme **DFS_Visit** dengan menggunakan representasi *adjacency matrix* yang digunakan sebelumnya dan akan dimodifikasi agar dapat melakukan deteksi *cycle*.

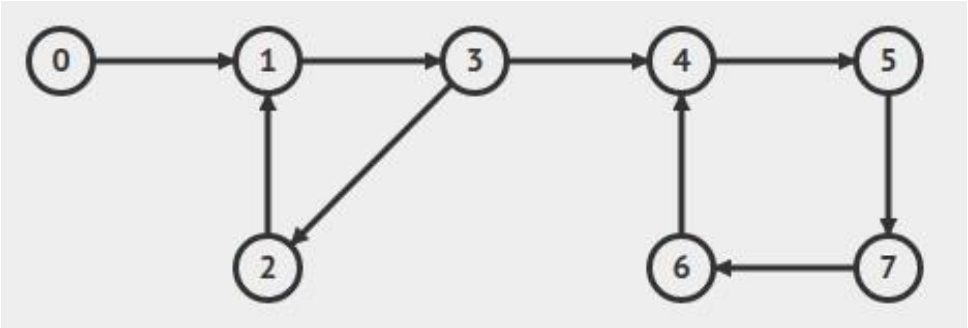
```
void DFS_visit(Graph *g, COLOR *vertex_colors, int v)
{
    int i;
    // Tampilkan v
    printf("%d ", v);

    // Tandai v dengan warna GRAY
    vertex_colors[v] = GRAY;

    // Cari vertex yang adjacent terhadap v, jika masih WHITE, panggil DFS_visit vertex itu
    for (i = 0; i < g->n_vertices; i++)
    {
        if (/* Soal 1: Tuliskan kondisi dimana ada vertex i adjacent thd v dan berwarna BUKAN white */)
            printf("Cycle! \n");
        if (g->adjacency_matrix[v][i] == 1 && vertex_colors[i] == WHITE)
            DFS_visit(g, vertex_colors, i);
    }

    vertex_colors[v] = BLACK;
}
```

Soal 1. Lengkapi bagian di atas agar algoritme DFS dapat melakukan deteksi *cycle* pada *graph* masukan!

Soal 2. Apakah keluaran dari program yang telah dilengkapi di atas jika diberikan *graph* di bawah ini?

Soal 3. Lakukan modifikasi sedemikan rupa sehingga:

- Fungsi/prosedur **DFS_Visit()** tidak lagi mencetak keluaran berupa kata ‘Cycle’ ketika terdapat *cycle* pada *graph* masukan.
- Fungsi/prosedur utama **DFS**() akan mengembalikan keluaran berupa bilangan **0** atau **1**, yang menunjukkan adanya *cycle* atau tidak pada graph masukan **(0** = tidak ada *cycle*, **1** = ada *cycle*)



Selanjutnya, kita akan melakukan modifikasi terhadap DFS untuk dapat melakukan *topological sort*. Pertama, kita tambahkan variabel pencatat waktu *finish* sebagai berikut:

```
typedef enum {WHITE, GRAY, BLACK} COLOR;

#define Inf 1000000000      // Untuk menyatakan nilai Inf/tak-hingga
int finish_time[MAXNUM_VERTICES];
int time = 0;               // Waktu global

/*!
Prosedur yang mengimplementasikan DFS_visit.
*/
void DFS_visit(Graph *g, COLOR *vertex_colors, int v)
{
    int i;
    // Tampilkan v
    printf("%d ", v);
```

dan inisialisasi nilainya pada prosedur DFS().

```
/*!
Prosedur yang mengimplementasikan DFS.
*/
void DFS(Graph *g)
{
    COLOR vertex_colors[MAXNUM_VERTICES];
    int i;
    for (i = 0; i < g->n_vertices; i++)
        vertex_colors[i] = WHITE;
    for (i = 0; i < g->n_vertices; i++)
        finish_time[i] = Inf;

    for (i = 0; i < g->n_vertices; i++)
    {
        if(vertex_colors[i] == WHITE)
            DFS_visit(g, vertex_colors, i);
    }

    printf("\n");
}
```

Selanjutnya, kita harus melakukan modifikasi terhadap fungsi DFS_Visit() agar mencatat waktu *finish* dengan benar.

```c
/*!
Prosedur yang mengimplementasikan DFS_visit.
*/
void DFS_visit(Graph *g, COLOR *vertex_colors, int v)
{
    int i;
    // Tampilkan v
    printf("%d time = %d ", v, time);

    // Tandai v dengan warna GRAY
    vertex_colors[v] = GRAY;
    // Increment nilai waktu

    // Cari vertex yang adjacent terhadap v, jika masih WHITE, panggil DFS_visit vertex itu
    for (i = 0; i < g->n_vertices; i++)
    {
        if (g->adjacency_matrix[v][i] == 1 && vertex_colors[i] == WHITE)
            DFS_visit(g, vertex_colors, i);
    }

    vertex_colors[v] = BLACK;
    // Increment nilai waktu
    // Catat waktu finish untuk vertex v
}
```

Soal 4. Lengkapi potongan fungsi DFS_Visit() di atas agar dapat mencatat waktu *finish* dari setiap *vertex* dalam *graph* masukan.

Soal 5. Lengkapi potongan kode pada fungsi main berikut untuk menampilkan waktu *finish* setiap *vertex* pada *graph* masukan.

```c
DFS(&g);

for (i = 0; i < g.n_vertices; i++)
    // Cetak waktu finish vertex i
printf("\n");
```

Soal 6. Gunakan fungsi **qsort** dari library **stdlib.h** untuk mengurutkan waktu *finish* dari setiap *vertex* dalam urutan **menurun**, dan kemudian gunakan ini untuk menghasilkan *topological sort* dari *graph* masukan yang diberikan (dengan asumsi tidak ada *cycle*).

Soal 7. Buatlah sebuah program lengkap yang melakukan hal berikut: diberikan sebuah *graph* masukan **G**, keluaran yang dihasilkan adalah:

- Kata “Cycle” jika **G** mengandung *cycle*.
- *Topological sort* dari semua *vertex* di **G** jika **G** tidak mengandung *cycle*.