Nama : Anugerah Prima Bagaskara

NPM : 140810180006

Tugas 6

Tugas Anda

1. Dengan menggunakan undirected graph dan adjacency matrix berikut, buatlah koding programmnya menggunakan bahasa C++.

Kode Program :

```
/*
Nama     : Anugerah Prima Bagaskara
NPM      : 140810180006
Kelas    : B
Program  : Program Undirected Graph
Tanggal  : 7 April 2020
*/

#include <iostream>
using namespace std;

void input(int &n){
   cout << "Banyak simpul dalam graph : ";cin >> n;
}

void input(int arr[], int size){
   for (int i = 0; i < size; i++){
      cout << "Nilai simpul ke-" << i+1 << " : ";cin >> arr[i];
   }
}

void input(int &num, int value){
   num = value;
```

```
}

void print(int arr[], int n) {
	for (int i = 0; i < n; i++) {
		cout << arr[i] << "\t";
   }
}

bool check(int arr[], int n, int value) {
	for (int i = 0; i < n; i++) {
		if (arr[i] == value){
			return true;
	}
	return false;
	}
}

int main(){
	int n, val;
	input(n);
	int simpul[n];
	input(simpul, n);
	int garis[n][n];

	for (int i = 0; i < n; i++) {
		int edge;
      cout << "\nBanyak garis simpul ke-" << i+1 << " (" << simpul[i] << ") : ";cin >>
edge;
      for (int j = 0; j < edge; j++) {  bool found = false;
        do {
           cout << "Simpul garis ke-" << i+1 << " : ";cin >> val;
```

```
            found = check(simpul, n, val);                if (!found)
                cout << endl << "Simpul tidak ditemukan!" << endl;
        } while (!found);

        input(garis[i][j], val);
    }
  }
  cout << endl << "Adjacency matrix dari undirected graph : \n\t";print(simpul, n);

  for (int i = 0; i < n; i++) {
                cout << endl << simpul[i] << "\t";

    for (int j = 0; j < n; j++) {
                        bool found = false;
        for (int k = 0; k < n; k++) {
                                if (garis[i][k] == simpul[j])
                                found = true;
        }

        if(found)               cout << "1\t";          else
            cout << "0\t";
    }
  }
}
```

2. Dengan menggunakan undirected graph dan representasi adjacency list, buatlah koding programmnya menggunakan bahasa C++.

Kode Program :

```
/*
Nama     : Anugerah Prima Bagaskara
NPM      : 140810180006
Kelas    : B
Program  : Program Undirected Graph
Tanggal  : 7 April 2020
 */

#include <iostream>
using namespace std;

struct edge {
	int data;
	edge *next;
};

struct node {
	int data;
	node *next;
	edge *edge;
};

void create(node *&newNode) {
	newNode = new node;
	newNode->next = NULL;
	newNode->edge = NULL;
	cin >> newNode->data;
}
```

```
void create(edge *&newEdge) {
        newEdge = new edge;
        newEdge->next = NULL;
        cin >> newEdge->data;
}

void insert(node *&list, node *newNode) {
        if (list == NULL)
                list = newNode;
        else {
                node *temp = list;
                while (temp->next){
                        temp = temp->next;
                        temp->next = newNode;
                }
   }
}

void insert(node *&parent, edge *newEdge) {
   if (parent->edge == NULL)          parent->edge = newEdge;
        else {
      edge *temp = parent->edge;
                while (temp->next){
                        temp = temp->next;
                        temp->next = newEdge;
                }
   }
}

void print(node *list){
```

```
    if (list == NULL)
        cout << "List simpul kosong." << endl;

            else {
        cout << "\nAdjacency list dari undirected graph : ";
                    node *temp = list;
                    while (temp) {
        edge *trav = temp->edge;

        cout << endl << temp->data << ": ";
                            while (trav){
            cout << trav->data;
                                    if (trav->next){
                                        cout << "->";
                                            trav = trav->next;
                                    }
        }
        temp = temp->next;
      }
    }
}

int main() {
    node *n, *list = NULL;     edge *e;     int s, g;

    cout << "Banyak simpul dalam graph : ";     cin >> s;

    for (int i = 0; i < s; i++) {
        cout << endl << "Nilai simpul ke-" << i+1 << " : ";
                            create(n);
                            insert(list, n);
```

```
        cout << "Banyak garis dari simpul " << n->data << " : ";         cin >> g;

            for (int j = 0; j < g; j++) {
            cout << "Simpul garis ke-" << j+1 << " : ";
            create(e);              insert(n, e);
        }
    }
    print(list);
}
```

3. Buatlah program Breadth First Search dari algoritma BFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan undirected graph sehingga menghasilkan tree BFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!

Kode Program :

```
/*
Nama     : Anugerah Prima Bagaskara
NPM      : 140810180006
Kelas    : B
Program  : Program Breadth First Search
Tanggal  : 7 April 2020
*/

#include <iostream>
#include <list>
using namespace std;

struct Graph {
            int vertex;
            list<int>* edge;
};
```

```
void makeGraph(Graph& G, int vertex) {
   G.vertex = vertex;
   G.edge = new list<int>[vertex];
}

void addEdge(Graph& G, int i, int j) {
   G.edge[i].push_back(j);
}

void traversal(Graph G) {
   for (int i = 0; i < G.vertex; ++i) {
      cout << "\n vertex "<<i<<"\n head";
                  for (auto x : G.edge[i])
                        cout << " -> " << x;
                        cout << endl;
   }
}

void BFS(Graph G, int s) {
         bool *visited = new bool[G.vertex];
         for (int i=0; i<G.vertex; i++)
                  visited[i] = false;

   list<int> queue;
         visited[s] = true;
         queue.push_back(s);

         while (!queue.empty()) {
                  s = queue.front();
                  cout<<s<<" ";queue.pop_front();
```

```
        for (list<int>::iterator i=G.edge[s].begin(); i != G.edge[s].end(); ++i) {
                        if (!visited[*i]) {
                                visited[*i] = true;
                                queue.push_back(*i);
                }
            }
    }
}

int main() {
    Graph G;
    int n, m, val, edge;

    cout << "Banyak simpul dalam graph : ";     cin >> n;     makeGraph(G, n);

    for (int i = 0; i < n; i++) {
        cout << endl << "Nilai simpul ke-" << i+1 << " : ";         cin >> val;
        cout << "Banyak edge dari simpul " << val << " : ";         cin >> m;

        for (int j = 0; j < m; j++) {
            cout << "Simpul edge ke-" << j+1 << " : ";             cin >> edge;             addEdge(G, val,
edge);
        }

    }
    cout << "\nBFS dimulai dari simpul "          << *G.edge[0].begin() - 1 << endl;
    BFS(G, *G.edge[0].begin() - 1);
}
```

Kompleksitas waktu asimptotik:

V : Jumlah Vertex

E : Jumlah Edge

- menandai setiap vertex belum dikunjungi : O(V)
- menandai vertex awal telah dikunjungi lalu masukan ke queue : O(1)
- keluarkan vertex dari queue kemudian cetak : O(V)
- kunjungi setiap vertex yang belum dikunjungi kemudian masukan ke queue : O(E) maka :

$T(n) = O(V) + O(1) + O(V) + O(E)$

$= O(maks(V,1)) + O(V) + O(E)$

$= O(V) + O(V) + O(E)$

$= O(maks(V,V)) + O(E)$

$= O(V) + O(E)$

$= O(V+E)$

4. Buatlah program Depth First Search dari algoritma DFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan undirected graph sehingga menghasilkan tree DFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!

Kode Program :

```
/*
Nama     : Anugerah Prima Bagaskara
NPM      : 140810180006
Kelas    : B
Program  : Program Depth First Search
Tanggal  : 7 April 2020
*/

#include <iostream>
#include <list>
using namespace std;

struct Graph {
        int vertex;
```

```
        list<int>* edge;
};

void makeGraph(Graph& G, int vertex) {
   G.vertex = vertex;
   G.edge = new list<int>[vertex];
}

void addEdge(Graph& G, int i, int j) {
   G.edge[i].push_back(j);
}

void traversal(Graph G) {
   for (int i = 0; i < G.vertex; ++i) {
      cout << "\n vertex "<<i<<"\n head";
                        for (auto x : G.edge[i])
                                cout << " -> " << x;
                                cout << endl;
   }
}

void DFSUtil(Graph G, int v, bool visited[]) {
        visited[v] = true;
        cout << v << " ";

   for (list<int>::iterator i = G.edge[v].begin(); i != G.edge[v].end(); ++i)
                if(!visited[*i])
        DFSUtil(G, *i, visited);
}
void DFS(Graph G, int s) {
        bool *visited = new bool[G.vertex];
```

```
                for (int i=0; i < G.vertex; i++)
                        visited[i] = false;

    for (int i=0; i < G.vertex; i++)
                if (visited[i] == false)
                        DFSUtil(G, i, visited);
}

int main() {
    Graph G;
    int n, m, val, edge;
    cout << "Banyak simpul dalam graph : ";cin >> n;makeGraph(G, n);

    for (int i = 0; i < n; i++) {
        cout << endl << "Nilai simpul ke-" << i+1 << " : ";        cin >> val;
        cout << "Banyak edge dari simpul " << val << " : ";        cin >> m;

        for (int j = 0; j < m; j++) {            cout << "Simpul edge ke-" << j+1 << " : ";            cin >>
edge;            addEdge(G, val, edge);
        }

    }     cout << "\nDFS dimulai dari simpul "          << *G.edge[0].begin() - 1 << endl;
    DFS(G, *G.edge[0].begin());
}
```

Kompleksitas waktu asimptotik:

V : Jumlah Vertex

E : Jumlah Edge

- menandai vertex awal telah dikunjungi kemudian cetak : O(1)
- rekursif untuk semua vertex : T(E/1)
- tandai semua vertex belum dikunjungi : O(V) • rekursif untuk mencetak DFS : T(V/1)

  maka :

$$
\begin{aligned}
T(n) &= O(1) + T(E/1) + O(V) + T(V/1) \\
&= O(1) + O(E) + O(V) + O(V) \\
&= O(mak(1,E)) + O(V) + O(V) \\
&= O(E) + O\,(V) + O(V) \\
&= O(maks(V,V)) + O(E) \\
&= O(V) + O(E) \\
&= O(V+E)
\end{aligned}
$$