

ISI

A. Tujuan praktikum

1. Mahasiswa mampu mengimplementasikan struktur Tries dan Graph dalam program.
2. Mahasiswa mampu menganalisis dan merancang program penyelesaian kasus menggunakan Tries dan Graph.

B. Dasar Teori

Trie adalah struktur data serupa tree yang disusun dari node-node linked list. Perbedaan trie dengan struktur tree adalah pada trie hirarki node tidak lain adalah hirarki alphabet yang disusun untuk membentuk kata kunci.

Graph dalam struktur data adalah koleksi node-node yang saling berhubungan dan tidak dalam bentuk hirarki. Dalam graph, setiap node bisa merepresentasikan data apa saja, misalnya data tentang kota-kota yang saling berhubungan. Garis-garis penghubungnya diibaratkan jalan-jalan yang menghubungkan kota dan seterusnya.

C. Soal dan Jawaban

1. Buatlah program kamus kata bahasa bugis dan makassar.
2. Buatlah program graph yang menyimpan jaringan setiap kecamatan di makassar dan jarak dari satu kecamatan ke kecamatan lain. Program dapat menampilkan jarak antara kecamatan berdasarkan input pengguna.

Jawaban :

1. Kode Program

```
#include <iostream>
#include <string>
#include <unordered_map>

using namespace std;

const int ALPHABET_SIZE = 26;

// Struktur untuk node dalam trie
struct TrieNode
{
    unordered_map<char, TrieNode *> children;
    bool isEndOfWord;
};

// Fungsi untuk menciptakan node baru dalam trie
TrieNode *getNode()
{
    TrieNode *node = new TrieNode;
    node->isEndOfWord = false;
    return node;
}

// Fungsi untuk menambahkan kata ke dalam trie
void insert(TrieNode *root, string word)
{
    TrieNode *pCrawl = root;

    for (int i = 0; i < word.length(); i++)
    {
        char ch = word[i];
        if (pCrawl->children.find(ch) == pCrawl->children.end())
            pCrawl->children[ch] = getNode();
        pCrawl = pCrawl->children[ch];
    }

    // Menandai akhir dari kata
    pCrawl->isEndOfWord = true;
}

// Fungsi untuk mencari kata dalam trie
bool search(TrieNode *root, string word)
{

```

```

TrieNode *pCrawl = root;

for (int i = 0; i < word.length(); i++)
{
    char ch = word[i];
    if (pCrawl->children.find(ch) == pCrawl->children.end())
        return false;
    pCrawl = pCrawl->children[ch];
}
return (pCrawl != NULL && pCrawl->isEndOfWord);
}

int main()
{
    // Inisialisasi root node
    TrieNode *root = getNode();

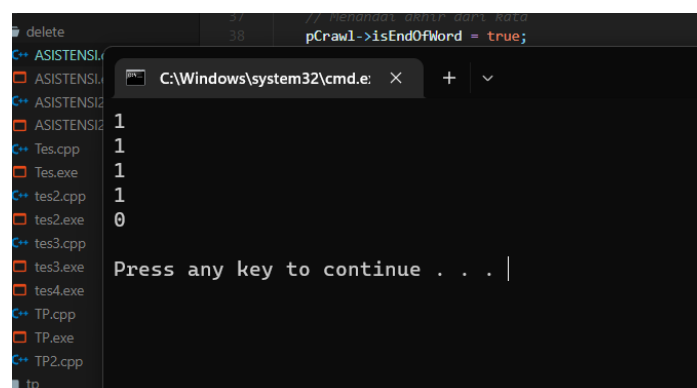
    // Menambahkan beberapa kata ke dalam trie
    insert(root, "bugis");
    insert(root, "nganre");
    insert(root, "nganre");
    insert(root, "tinro");

    insert(root, "ikan");

    // Mencari beberapa kata dalam trie
    cout << search(root, "bugis") << endl; // output: 1
    cout << search(root, "nganre") << endl; // output: 1
    cout << search(root, "nganre") << endl; // output: 1
    cout << search(root, "tinro") << endl; // output: 1
    cout << search(root, "jakarta") << endl; // output: 0

    return 0;}

```



```

C:\Windows\system32\cmd.e  x  +  v
1
1
1
1
0
Press any key to continue . . . |

```

2. Kode Program

```
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

// Struktur untuk menyimpan informasi tentang kecamatan dan jarak ke
// kecamatan lain
struct District
{
    string name;
    unordered_map<string, int> distances;
};

// Struktur untuk menyimpan graf yang terdiri dari setiap kecamatan
struct Graph
{
    unordered_map<string, District> districts;
};

// Fungsi untuk menambahkan kecamatan baru ke dalam graf
void addDistrict(Graph &graph, string name)
{
    graph.districts[name] = { name };
}

// Fungsi untuk menambahkan jarak antar kecamatan ke dalam graf
void addDistance(Graph &graph, string src, string dest, int distance)
{
    graph.districts[src].distances[dest] = distance;
    graph.districts[dest].distances[src] = distance;
}

// Fungsi untuk menampilkan jarak antar kecamatan berdasarkan input
// pengguna
void showDistance(Graph &graph)
{
    string src, dest;
    cout << "Masukkan kecamatan asal: ";
    cin >> src;
    cout << "Masukkan kecamatan tujuan: ";
    cin >> dest;

    int distance = graph.districts[src].distances[dest];
```

```

        cout << "Jarak dari " << src << " ke " << dest << " adalah " << distance << "
        km." << endl;
    }

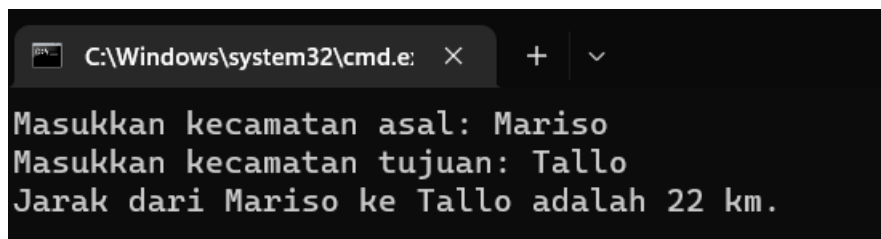
int main()
{
    Graph graph;

    // Menambahkan kecamatan ke dalam graf
    addDistrict(graph, "Tamalate");
    addDistrict(graph, "Manggala");
    addDistrict(graph, "Makassar");
    addDistrict(graph, "Mariso");
    addDistrict(graph, "Tallo");

    // Menambahkan jarak antar kecamatan ke dalam graf
    addDistance(graph, "Tamalate", "Manggala", 10);
    addDistance(graph, "Tamalate", "Makassar", 15);
    addDistance(graph, "Tamalate", "Mariso", 12);
    addDistance(graph, "Manggala", "Makassar", 20);
    addDistance(graph, "Manggala", "Tallo", 25);
    addDistance(graph, "Makassar", "Mariso", 18);
    addDistance(graph, "Mariso", "Tallo", 22);

    // Menampilkan jarak antar kecamatan berdasarkan input pengguna
    showDistance(graph);
    return 0;
}

```



```

C:\Windows\system32\cmd.e:
Masukkan kecamatan asal: Mariso
Masukkan kecamatan tujuan: Tallo
Jarak dari Mariso ke Tallo adalah 22 km.

```

Kesimpulan

Trie adalah struktur data serupa tree yang disusun dari node-node linked list. Perbedaan trie dengan struktur tree adalah pada trie hirarki node tidak lain adalah hirarki alphabet yang disusun untuk membentuk kata kunci. graph dalam struktur data adalah koleksi node-node yang saling berhubungan dan tidak dalam bentuk hirarki.

