

PRESENTASI

STRUKTUR DATA

DAN ALGORITMA

KELOMPOK 8



**ANGGOTA
KELOMPOK 8**



Aggra kurnia idhan

(G1A02003)

Ahmad Febri Yan

(G1A024005)

Muhammad Nathan Algibran

(G1A024057)

Tri Haiji Januarli

(G1A024069)

Farhan Muhammad Rizki

(G1A024093)

M. Arif Rahman

(G1A0240109)



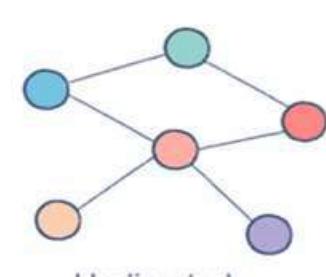
GRAPH

PENGERTIAN GRAPH

Graph merupakan jenis struktur data umum yang susunan datanya tidak berdekatan satu sama lain (non-linier), dimana tipe ini mengandung beberapa node atau simpul berhingga untuk menyimpan data dan antara dua simpul terdapat hubungan saling berkaitan.

JENIS-JENIS GRAPH

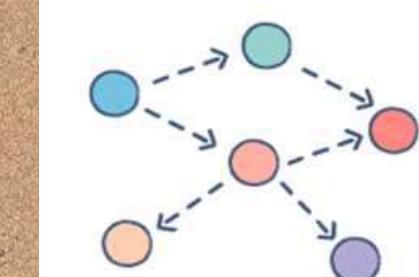
Undirected graph



Undirected

Simpul-simpul yang saling terhubung dengan edge yang sifatnya dua arah, Ketika simpul 1 dan 2 saling terhubung, kita dapat menjelajahi dari simpul 1 ke simpul 2, begitupun sebaliknya. Contohnya jalan dua arah, aliran data, peta jalan satu arah, system pengiriman barang.

Directed Graph



Directed

Simpul-simpul terhubung oleh edge yang hanya dapat menjelajahi satu arah pada simpul yang ditunjuk, jika ada simpul A yang terhubung ke simpul B, arah panah menuju simpul B maka hanya dapat menjelajah (traversing) dari simpul A ke simpul B, tidak berlaku sebaliknya. Contohnya jalan satu arah, jaringan komunikasi, jaringan sosial.

ADA TIGA STUKTUR DATA GRAPH

Adjacency List

Struktur data yang menghubungkan setiap simpul (verteks) dengan daftar simpul yang terhubung (edges).

Adjacency Matrix

Merupakan matrix dimana baris dan kolom mewakili simpul – simpul, dan entri dalam matriks menunjukkan keberadaan sisi antara dua simpul.

Edge List

Merepresentasikan graph dengan hanya mencatat semua tepinya (hubungan antar simpul).

KELEBIHAN STRUKTUR DATA GRAPH

- Memudahkan menemukan jalur terpendek dan tetangga dari node.
- Graph digunakan untuk mengimplementasikan algoritma seperti DFS untuk menelusuri kedalam satu arah sejauh mungkin sebelum mundur (eksplorasi) dan BFS untuk menemukan jalur terpendek dalam graph tak berbobot.
- Graph membantu dalam mengatur data.
- Karena strukturnya non-linier, ini membantu dalam memahami masalah yang kompleks dan visualisasinya

KEKURANGAN STRUKTUR DATA GRAPH

- Graph menggunakan banyak pointer yang bisa rumit untuk ditangani.
- Memiliki kompleksitas memori yang besar jika graph direpresentasikan dengan adjacency matrix, maka edge tidak memungkinkan untuk sejajar dan operasi perkalian graph juga sulit untuk dilakukan.

GRAPH TRAVERSAL

PENGERTIAN GRAPH TRAVERSAL

Graph traversal adalah teknik yang digunakan untuk mengunjungi setiap node dalam graf, baik untuk keperluan pencarian data, pemrosesan informasi, maupun analisis struktur jaringan.

DUA METODE ALGORITMA TRAVERSAL

Breadth-First Search (BFS)

Pada metode Breadth-First Search, semua node pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi node-node pada level $n+1$. Pencarian dimulai dari node awal, kemudian berlanjut ke level pertama dari kiri ke kanan, lalu berpindah ke level berikutnya dengan urutan yang sama hingga solusi ditemukan.

Depth – First Search (DFS)

Pada Depth First Search, proses pencarian akan dilaksanakan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi, Proses ini diulangi terus hingga ditemukannya solusi.

SHORTEST PATH TREE

PENGERTIAN SHORTEST PATH TREE

Shortest path tree adalah pohon jalur pendek yang menggambarkan dari satu titik ke titik lainnya dalam suatu grafik, di shortest path tree ada Breadth-first search (BFS) yang bisa digunakan untuk memecahkan unweighted shortest paths problem.

ADA 3 VARIASI SHORTEST PATH TREE SEARCH

Single pair shortest paths

Contohnya diberikan titik sumber s dan titik tujuan t, dan mencari berapa lintasan terpendek dari titik s ke titik t.

Single source shortest paths

contohnya diberikan titik sumber s, dan mencari apa saja jalur terpendek dari titik s ke semua titik di dalam graph.

Multiple source shortest paths

Contohnya diberikan sekumpulan titik sumber s, lalu mencari apa saja lintasan terpendek dari titik manapun di titik s ke semua titik di dalam graph.

DIJKSTRA'S ALGORITHM

PENGERTIAN DIJKSTRA'S ALGORITHM

Dijkstra's algorithm adalah algoritma yang paling terkenal untuk menemukan SPT berbobot dalam suatu graph. Algoritma Dijkstra juga adalah algoritma yang paling terkenal untuk menemukan shortest path tree dalam grafik berbobot. Algoritma Dijkstra secara bertahap membangun shortest path tree pada setiap iterasi loop while. Algoritma dijkstra's memilih simpul yang belum dikunjungi berikutnya berdasarkan sum total cost of its shortest path.

ALGORITMA YANG PALING UMUM UNTUK MENEMUKN PENGURUTAN TOPOLOGI

- DFS (Depth-first search)
- DFS postorder
- Cara khusus untuk menjalankan penelusuran mendalam. DFS bersifat rekursif, jadi kita dapat memproses setiap nilai sebelum atau setelah pemanggilan rekursif ke semua tetangga. Preorder memproses nilai sebelum pemanggilan rekursif sementara postorder memproses nilai setelah pemanggilan rekursif.
- Reverse DFS postorder. The DFS postorder list, tetapi dalam urutan mundur (terbalik).

STUDI KASUS

Seorang traveler berencana melakukan perjalanan darat dari Bengkulu menuju Banda Aceh. Karena jarak yang ditempuh cukup jauh dan melibatkan banyak kota besar di Pulau Sumatera, diperlukan strategi perencanaan rute yang tepat untuk menghemat waktu, biaya, serta energi selama perjalanan

JARAK TEMPuh ANTAR KOTA

```
{"Bengkulu", {"Lubuklinggau": 130}, {"Palembang": 300}},
{"Lubuklinggau", {"Palembang": 260}, {"Jambi": 400}},
 {"Palembang", {"Jambi": 200}},
 {"Jambi", {"Pekanbaru": 450}, {"Medan": 900}},
 {"Pekanbaru", {"Duri": 120}},
 {"Duri", {"Medan": 500}},
 {"Medan", {"Lhokseumawe": 400}},
 {"Lhokseumawe", {"Banda Aceh": 270}},
 {"Banda Aceh", {}}
```

PENYELESAIAN MASALAH

Traversal Graph

DUA METODE ALGORITMA TRAVERSAL

Breadth-First Search (BFS)

```
void bfs(string start, string end) {
    unordered_map<string, bool> visited;
    unordered_map<string, string> previous;

    queue<string> q;
    q.push(start);
    visited[start] = true;

    bool found = false;

    while (!q.empty()) {
        string current = q.front();
        q.pop();

        if (current == end) {
            found = true;
            break;
        }

        for (auto neighbor : graph[current]) {
            string nextNode = neighbor.first;
            if (!visited[nextNode]) {
                visited[nextNode] = true;
                previous[nextNode] = current;
                q.push(nextNode);
            }
        }
    }
}
```

Depth – First Search (DFS)

```
void dfs(string start, string end) {
    unordered_map<string, bool> visited;
    vector<string> path;

    bool found = dfsUtil(start, end, visited, path);

    if (!found) {
        cout << "Tidak ada jalur dari " << start << " ke " << end << endl;
    } else {
        cout << "Rute dari " << start << " ke " << end << " (DFS):" << endl;
        for (int i = 0; i < path.size(); i++) {
            cout << path[i];
            if (i != path.size() - 1) cout << " -> ";
        }
        cout << endl;
        cout << "Jumlah kota yang dilewati: " << path.size() << endl;
    }
}
```

**PENYELESAIAN
MASALAH**

**Algoritma
Dijkstra**

```
void dijkstra(string start, string end) {  
  
    unordered_map<string, int> distances;  
    unordered_map<string, string> previous;  
  
    for (const auto& node : graph) {  
        distances[node.first] = INT_MAX;  
    }  
    distances[start] = 0;  
  
    priority_queue<NodeDistance, vector<NodeDistance>, greater<NodeDistance>> pq;  
    pq.push({0, start});  
  
    while (!pq.empty()) {  
        int currentDistance = pq.top().first;  
        string currentNode = pq.top().second;  
        pq.pop();  
  
        if (currentNode == end) {  
            break;  
        }  
  
        for (auto neighbor : graph[currentNode]) {  
            string nextNode = neighbor.first;  
            int weight = neighbor.second;  
            int newDistance = currentDistance + weight;  
  
            if (newDistance < distances[nextNode]) {  
                distances[nextNode] = newDistance;  
                previous[nextNode] = currentNode;  
                pq.push({newDistance, nextNode});  
            }  
        }  
    }  
}
```

KAITAN TRAVERSAL DAN DIJKSTRA DALAM STUDI KASUS

Pada studi kasus perjalanan Bengkulu - Banda Aceh:

- BFS/DFS digunakan untuk mengecek semua kemungkinan jalur.
- Dijkstra digunakan untuk mencari jalur dengan jarak total minimum.

HASIL ANALISIS

Output Travelsals

```
PS D:\Tugas Graph SDA> cd "d:\Tugas Graph SDA\GRAPH\Travelsals\" ; if ($?) { g++ bfs.cpp -o bfs } ; if ($?) { .\bfs
Rute dari Bengkulu ke Banda Aceh (BFS)
Bengkulu -> Lubuklinggau -> Jambi -> Medan -> Lhokseumawe -> Banda Aceh
Jumlah kota yang dilewati: 6
PS D:\Tugas Graph SDA\GRAPH\Travelsals> cd "d:\Tugas Graph SDA\GRAPH\Travelsals\" ; if ($?) { g++ dfs.cpp -o dfs } ; if ($?) { .\dfs
Rute dari Bengkulu ke Banda Aceh (DFS):
Bengkulu -> Lubuklinggau -> Palembang -> Jambi -> Pekanbaru -> Duri -> Medan -> Lhokseumawe -> Banda Aceh
Jumlah kota yang dilewati: 9
PS D:\Tugas Graph SDA\GRAPH\Travelsals>
```

Output Djikstra Algorithm

```
PS D:\Tugas Graph SDA> cd "d:\Tugas Graph SDA\GRAPH\Shortest path tree\" ; if ($?) { g++ djikstra.cpp -o djikstra } ; if ($?) { .\djikstra
● Jarak terpendek dari Bengkulu ke Banda Aceh adalah 2070 km.
Rutenya: Bengkulu -> Palembang -> Jambi -> Medan -> Lhokseumawe -> Banda Aceh
○ PS D:\Tugas Graph SDA\GRAPH\Shortest path tree>
```

KESIMPULAN STUDI KASUS

Penerapan traversal graph terbukti efektif dalam memetakan semua jalur yang tersedia, sedangkan algoritma Dijkstra membantu dalam menentukan jalur tercepat. Melalui analisis ini, perencanaan perjalanan dari Bengkulu ke Banda Aceh dapat dilakukan dengan lebih baik, mengoptimalkan waktu tempuh, mengurangi biaya perjalanan, dan meningkatkan kenyamanan.

KESIMPULAN

Graph merupakan struktur data non-linear yang mengandung kumpulan simpul (node) berhingga untuk menyimpan data dan antara dua simpul terdapat hubungan (edge) saling berkaitan saling berkaitan, antar elemennya many-to-one, one-to-many, dan many-to-many, dengan melalui undirected, directed, dan weighted, unweighted graph. Dan mempunyai 3 komponen yaitu graph data type, graph traversal dan shortest path tree



TERIMA KASIH