EKSPLORASI DAN ANALISIS STRATEGI ALGORITMA

PERANCANGAN DAN ANALISIS ALGORITMA

202323430048

DOSEN PENGAMPU:

Randi Proska Sandra, S.Pd., M.Sc.

OLEH:

Wahyu Isnan - 22343031

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2024

A. PENJELASAN PROGRAM/ALGORITMA

1. Prim's Algorithm

Algoritma Prim adalah salah satu algoritma greedy dalam teori graf yang digunakan untuk menemukan pohon rentang minimum (MST) dalam sebuah graf berbobot yang berarah, yang merupakan subgraf yang terhubung dan tidak mengandung siklus dengan bobot total minimum. MST adalah subgraf yang terhubung dan tidak mengandung siklus dengan bobot total minimum di antara semua MST dari graf tersebut. Algoritma ini dinamai dari ilmuwan komputer Amerika, Robert C. Prim, yang mengembangkan algoritma ini pada tahun 1957. Tujuan utamanya adalah untuk menemukan himpunan subset simpul dari graf yang membentuk pohon dengan total bobot minimum, di mana setiap simpul terhubung dengan tepat satu simpul lainnya di pohon.[1]

Algoritma Prim dimulai dengan memilih simpul awal dan secara berulang menambahkan simpul terdekat yang belum termasuk dalam pohon secara bertahap. Proses ini dilakukan hingga seluruh simpul telah disertakan dalam pohon. Ada beberapa langkah untuk memahami mengenai algoritma ini:[2]

1. **Inisialisasi:** Pilih simpul awal secara acak dari graf. Simpan simpul-simpul yang sudah dipilih dalam sebuah himpunan (misalnya, *MSTSet* yang berarti "Minimum Spanning Tree Set"). Inisialisasi juga mempersiapkan struktur data untuk menyimpan bobot sisi yang terhubung dengan simpul yang sudah dipilih dan simpul yang belum dipilih.
2. **Iterasi:** Selama himpunan simpul dalam *MSTSet* belum mencakup semua simpul dalam graf:
    a. Pilih Sisi Minimum: Cari sisi dengan bobot minimum yang terhubung dengan simpul yang sudah dipilih dan simpul yang belum dipilih.
    b. Tambahkan simpul: Tambahkan simpul yang terhubung dengan sisi tersebut ke dalam *MSTSet*.
    c. Update Bobot: Perbarui bobot sisi yang terhubung dengan simpul yang belum dipilih, jika bobot baru lebih kecil dari bobot sebelumnya.
3. **Output:** Setelah semua simpul termasuk dalam *MSTSet*, output adalah himpunan sisi yang membentuk pohon rentang minimum.

Algoritma Prim memiliki banyak aplikasi dalam berbagai bidang, terutama dalam pemodelan dan analisis jaringan, infrastruktur, serta optimasi sumber daya. Berikut adalah beberapa contoh pengaplikasiannya:

1. **Jaringan Komunikasi**: Algoritma Prim dapat digunakan dalam perencanaan jaringan komunikasi, seperti jaringan telekomunikasi atau jaringan komputer. Dalam konteks ini, simpul dapat mewakili titik akses atau node dalam jaringan, dan tepi dapat mewakili jalur komunikasi antara node. Dengan menggunakan Algoritma Prim, perusahaan telekomunikasi dapat merencanakan jaringan dengan efisien untuk mengoptimalkan sinyal, throughput, atau biaya.
2. **Manajemen Sumber Daya Komputer**: Dalam pusat data atau sistem komputasi terdistribusi, Algoritma Prim dapat digunakan untuk

merencanakan koneksi jaringan antar server atau node. Hal ini membantu dalam mengoptimalkan aliran data, mengurangi latensi, dan memastikan ketersediaan layanan dengan cara yang efisien.

3. **Routing dalam Jaringan Sensor Nirkabel**: Dalam jaringan sensor nirkabel (WSN), Algoritma Prim dapat digunakan untuk merencanakan rute pengiriman data yang efisien antar node sensor. Dengan memilih jalur dengan bobot terendah, Algoritma Prim membantu dalam menghemat energi, meningkatkan efisiensi komunikasi, dan memperpanjang umur baterai node sensor.
4. **Jaringan Listrik**: Algoritma Prim juga dapat diterapkan dalam perencanaan jaringan listrik, terutama dalam distribusi daya dan penyediaan energi. Simpul dapat mewakili stasiun pembangkit, pembagi, atau pengguna, sementara tepi mewakili jalur distribusi. Dengan menggunakan Algoritma Prim, perusahaan listrik dapat merencanakan jaringan yang efisien untuk memastikan pasokan daya yang andal dan efisien.[3]
5. **Rancang Bangun Jalan dan Jembatan**: Dalam proyek rancang bangun infrastruktur seperti jalan, jembatan, atau rel kereta api, Algoritma Prim dapat digunakan untuk merencanakan jaringan transportasi yang efisien. Di sini, simpul dapat mewakili titik-titik tujuan atau pertemuan, sedangkan tepi dapat mewakili segmen jalan atau jalur rel. Algoritma Prim membantu dalam menentukan rute-rute terpenting yang harus dibangun terlebih dahulu untuk menghubungkan semua titik secara efisien.[4]

Dengan berbagai aplikasi ini, Algoritma Prim menjadi alat yang penting dalam pemodelan, perencanaan, dan optimasi berbagai sistem yang melibatkan pengaturan jaringan, alokasi sumber daya, dan pengambilan keputusan.

Algoritma Prim memiliki kompleksitas waktu yang bergantung pada implementasi, tetapi biasanya memiliki kompleksitas waktu O(V^2) dengan representasi matriks adjacency dan O(E log V) dengan representasi daftar adjacency menggunakan struktur data heap seperti priority queue, di mana V adalah jumlah simpul dalam graf dan E adalah jumlah sisi.[5]

B. PSEUDOCODE

```
Graf:
    Fungsi __init__(self, simpul):
        inisialisasi JmlhSimpul dengan simpul
        inisialisasi graf dengan matriks nol berukuran simpul x
simpul


    Fungsi cetakMST(self, induk):
        Cetak "Sisi \tBobot"
        Untuk i dari 1 hingga JmlhSimpul-1:
            Cetak induk[i], "-", i, "\t", graf[i][induk[i]]
```

```
    Fungsi indeksMin(self, kunci, mstSet):
        min = nilai maksimum integer
        Untuk setiap simpul v:
            Jika kunci[v] < min dan mstSet[v] == False:
                Set min sama dengan kunci[v]
                Set min_index sama dengan v
        Kembalikan min_index

    Fungsi primMST(self):
        Buat array kunci dengan nilai maksimum integer sebanyak
JmlhSimpul
        Buat array induk dengan nilai None sebanyak JmlhSimpul
        Set kunci[0] sama dengan 0
        Buat array mstSet dengan nilai False sebanyak
JmlhSimpul
        Set induk[0] sama dengan -1

        Untuk setiap iterasi dari 0 hingga JmlhSimpul:
            Set u sama dengan indeksMin(kunci, mstSet)
            Set mstSet[u] sama dengan True

            Untuk setiap simpul v:
                Jika graf[u][v] > 0 dan mstSet[v] == False dan
kunci[v] > graf[u][v]:
                    Set kunci[v] sama dengan graf[u][v]
                    Set induk[v] sama dengan u

        Panggil cetakMST(induk)

Jika __name__ sama dengan '__main__':
    Buat objek g dengan Graf(5)
    Set graf dalam objek g dengan matriks yang telah ditentukan

    Panggil primMST pada objek g
```

C. SOURCE CODE

```python
# Program Algoritma Prim untuk Minimum Spanning Tree (MST)

import sys

class Graf():
    def __init__(self, simpul):
        self.JmlhSimpul = simpul
        self.graf = [[0 for kolom in range(simpul)] for baris in range(simpul)]

    def cetakMST(self, induk):
        print("Sisi \tBobot")
        for i in range(1, self.JmlhSimpul):
            print(induk[i], "-", i, "\t", self.graf[i][induk[i]])

    def indeksMin(self, kunci, mstSet):
        min = sys.maxsize

        for v in range(self.JmlhSimpul):
            if kunci[v] < min and mstSet[v] == False:
                min = kunci[v]
                min_index = v

        return min_index

    def primMST(self):
        kunci = [sys.maxsize] * self.JmlhSimpul
        induk = [None] * self.JmlhSimpul
        kunci[0] = 0
        mstSet = [False] * self.JmlhSimpul
        induk[0] = -1

        for cout in range(self.JmlhSimpul):
            u = self.indeksMin(kunci, mstSet)
            mstSet[u] = True

            for v in range(self.JmlhSimpul):
                if self.graf[u][v] > 0 and mstSet[v] == False \
                and kunci[v] > self.graf[u][v]:
                    kunci[v] = self.graf[u][v]
                    induk[v] = u

        self.cetakMST(induk)


if __name__ == '__main__':
    # Inisialisasi graf dengan jumlah simpul 5
    g = Graf(5)

    # Definisi matriks ketetanggaan graf
    g.graf = [[0, 2, 0, 6, 0],
              [2, 0, 3, 8, 5],
              [0, 3, 0, 0, 7],
              [6, 8, 0, 0, 9],
              [0, 5, 7, 9, 0]]

    # Temukan dan cetak Minimum Spanning Tree (MST)
    g.primMST()

```

D. ANALISIS KEBUTUHAN WAKTU

1. Analisis Menyeluruh

Algoritma Prim terdiri dari beberapa fungsi dan loop. Berikut adalah perkiraan kebutuhan waktu untuk setiap bagian:

a. Fungsi `__init__`:
   - Menginisialisasi variabel `JmlhSimpul`, `graf`, `kunci`, `induk`, dan `mstSet`.
   - Total operasi: O(1)

b. Fungsi `cetakMST`:
   - Melakukan loop sebanyak `JmlhSimpul - 1` kali
   - Di setiap iterasi, melakukan operasi `` `print` `` dan beberapa operasi aritmatika.
   - Total operasi: O(V)

c. Fungsi `indeksMin`:
   - Melakukan loop sebanyak `JmlhSimpul` kali.
   - Di setiap iterasi, melakukan beberapa operasi perbandingan dan assignment.
   - Total operasi: O(V)

d. Fungsi `primMST`:
   - Melakukan loop sebanyak `` `JmlhSimpul` `` kali (loop luar).
   - Di dalam loop luar, terdapat loop lain yang melakukan operasi `indeksMin` dan beberapa operasi assignment.
   - Total operasi dalam loop luar: O(V^2)
   - Di luar loop luar, terdapat beberapa operasi assignment dan `print`.
   - Total operasi: O(V)

Total kebutuhan waktu: O(1) + O(V) + O(V) + O(V^2) + O(V) = O(V^2)

2. Analisis Berdasarkan Operasi Abstrak

Algoritma Prim menggunakan operasi abstrak berikut:

a. Pembuatan MST: O(E)
b. Pencarian simpul dengan bobot minimum: O(V)

Total operasi abstrak: O(E + V)

3. Analisis Best-Case, Worst-Case, dan Average-Case

a. Best-Case:
   - Graf terhubung dengan hanya satu MST.
   - Algoritma Prim akan langsung menemukan MST tanpa perlu melakukan banyak perbandingan.
   - Kebutuhan waktu: O(E)

b. Worst-Case:
   - Graf terhubung dengan banyak MST.
   - Algoritma Prim perlu melakukan banyak perbandingan untuk menemukan MST dengan bobot minimum.
   - Kebutuhan waktu: O(V^2)

c. Average-Case:
   - Tergantung pada struktur graf dan distribusi bobot edge.
   - Kebutuhan waktu: Diperkirakan O(E log V)

E. REFERENSI

[1] "Introduction to the Design and Analysis of Algorithms (3rd ed.) [Levitin 2011-10-09]".

[2] T. Syahputra, D. Setiawan, P. Studi, S. Informasi, and R. Kisaran, "IMPLEMENTASI ALGORITMA PRIM DENGAN TEORI GRAPH PADA WPF GRAPH."

[3] D. W. Nugraha, "IMPLEMENTASI ALGORITMA PRIM PADA JARINGAN DISTRIBUSI LISTRIK PRIMER DENGAN MENGGUNAKAN PROGRAM BERBASIS GIS."

[4] M. Sam, "PENERAPAN ALGORITMA PRIM UNTUK MEMBANGUN POHON MERENTANG MINIMUM (MINIMUM SPANNING TREE) DALAM PENGOPTIMALAN JARINGAN TRANSMISI NASIONAL PROVINSI SULAWESI SELATAN," 2016. [Online]. Available: www.djlpe.go.id

[5] D. W. Nugraha, "APLIKASI ALGORITMA PRIM UNTUK MENENTUKAN MINIMUM SPANNING TREE SUATU GRAF BERBOBOT DENGAN MENGGUNAKAN PEMROGRAMAN BERORIENTASI OBJEK," 2011.

F. LAMPIRAN LINK GITHUB

Lampiran link GitHub ada disini