Laporan praktikum daa

Nama : Yohanes Yeningga

Nim : 20220047

Matkul : Prak.daa latihan 2

1. Algoritma aliran maksimum adalah algoritma yang digunakan untuk mencari aliran maksimum dalam jaringan yang terdiri dari simpul-simpul yang terhubung oleh jalur-jalur. Aliran maksimum mengacu pada jumlah maksimum unit yang dapat mengalir melalui jaringan dari simpul asal (source) ke simpul tujuan (sink).

   Salah satu algoritma terkenal untuk mencari aliran maksimum adalah Algoritma Ford-Fulkerson, yang menggunakan pendekatan augmenting path. Algoritma ini bekerja dengan mengulangi langkah-langkah berikut:

   1) Inisialisasi aliran awal: Mengatur aliran pada setiap jalur awal ke 0.
   2) Cari jalur meningkat: Mencari jalur dari simpul asal ke simpul tujuan di dalam jaringan yang belum sepenuhnya dimanfaatkan oleh aliran saat ini.
   3) Menghitung kapasitas residu: Menghitung kapasitas residu yang tersisa pada setiap jalur dalam jaringan, yaitu seberapa banyak aliran tambahan yang dapat dilalui melalui jalur tersebut.
   4) Menambahkan aliran: Menambahkan aliran ke jalur yang telah ditemukan dalam langkah sebelumnya dengan jumlah yang sama dengan kapasitas residu terkecil di jalur tersebut.
   5) Mengupdate jaringan residu: Memperbarui jaringan residu dengan mengurangi aliran yang ditambahkan dari kapasitas residu pada setiap jalur, dan menambahkan aliran yang ditambahkan ke arah berlawanan pada jalur yang sama.
   6) Ulangi langkah 2-5: Melakukan langkah-langkah 2-5 secara berulang hingga tidak ada jalur meningkat yang tersisa di jaringan residu.

   Algoritma ini berakhir ketika tidak ada jalur meningkat yang tersisa dalam jaringan residu, yang menandakan bahwa aliran maksimum telah dicapai. Algoritma Ford-Fulkerson menghasilkan aliran maksimum dengan memperbarui jaringan residu berulang kali.

Dalam pemecahan masalah jaringan pada kehidupan nyata, algoritma aliran maksimum dapat diterapkan dalam berbagai konteks, seperti:

1. Pemecahan masalah transportasi: Algoritma aliran maksimum dapat digunakan untuk mengoptimalkan aliran barang atau orang dalam sistem transportasi. Misalnya, dalam perencanaan rute transportasi, algoritma ini dapat membantu menentukan jalur-jalur yang optimal untuk mengalirkan barang dari sumber ke tujuan dengan memaksimalkan kapasitas aliran.
2. Pemecahan masalah pengalokasian sumber daya: Dalam lingkungan bisnis, algoritma aliran maksimum dapat membantu dalam pengalokasian sumber daya, seperti mengalokasikan mesin atau tenaga kerja ke tugas-tugas yang berbeda. Dengan mengidentifikasi aliran maksimum yang dapat dilakukan melalui berbagai jalur, pengambilan keputusan mengenai pengalokasian sumber daya dapat dioptimalkan.
3. Jaringan komunikasi: Dalam pengaturan jaringan komunikasi, algoritma aliran maksimum dapat digunakan untuk mengoptimalkan aliran data atau informasi antara simpul-simpul dalam jaringan. Hal ini dapat membantu dalam perencanaan jaringan komunikasi yang efisien, seperti mengatur aliran data melalui router atau mengalokasikan jalur komunikasi untuk memaksimalkan throughput jaringan.
4. Analisis kebocoran jaringan pipa: Algoritma aliran maksimum dapat diterapkan dalam analisis jaringan pipa untuk mendeteksi dan memperbaiki kebocoran. Dengan memodelkan aliran air dalam jaringan pipa sebagai aliran maksimum, algoritma ini dapat membantu mengidentifikasi jalur-jalur yang mengalami kebocoran dan mengoptimalkan aliran untuk mengurangi kerugian air.

Dalam semua aplikasi ini, algoritma aliran maksimum membantu dalam mengoptimalkan aliran melalui jaringan dengan memastikan penggunaan sumber daya yang efisien dan optimal

2. Algoritma jaringan minimum tidak merupakan istilah yang lazim digunakan dalam konteks pemecahan masalah jaringan. Sejauh yang saya tahu, tidak ada algoritma yang secara khusus disebut sebagai "algoritma jaringan minimum" dalam literatur yang berhubungan dengan pemecahan masalah rute terpendek pada jaringan.

Namun, terdapat algoritma yang dikenal sebagai "algoritma rute terpendek" yang digunakan untuk mencari rute terpendek antara dua simpul dalam jaringan. Salah satu algoritma terkenal yang digunakan untuk tujuan ini adalah Algoritma Dijkstra. Berikut adalah penjelasan singkat mengenai Algoritma Dijkstra

Algoritma Dijkstra digunakan untuk mencari rute terpendek dari simpul asal (source) ke simpul tujuan (destination) dalam jaringan berbobot positif, di mana setiap jalur dihubungkan dengan bobot yang menunjukkan jarak, biaya, atau pengaruh lainnya. Berikut adalah langkah-langkah algoritma Dijkstra:

1) Inisialisasi: Mengatur jarak awal dari simpul asal ke semua simpul lain dalam jaringan sebagai tak terhingga, kecuali simpul asal sendiri yang diatur sebagai 0. Buat juga sebuah set kosong yang berisi simpul-simpul yang belum dikunjungi.
2) Pilih simpul saat ini: Mulai dari simpul asal, pilih simpul dengan jarak terpendek dari simpul yang belum dikunjungi.
3) Perbarui jarak: Untuk setiap tetangga dari simpul saat ini yang belum dikunjungi, perbarui jarak yang diketahui jika ada jalur yang lebih pendek melalui simpul saat ini. Jarak ini adalah jumlah jarak saat ini dari simpul asal ke simpul saat ini, ditambah bobot jalur dari simpul saat ini ke tetangganya.
4) Tandai simpul saat ini sebagai dikunjungi.
5) Ulangi langkah 2-4: Ulangi langkah-langkah ini hingga semua simpul telah dikunjungi atau jika jarak ke simpul tujuan tidak dapat lagi diperbarui (misalnya, jika ada jalur terputus ke simpul tujuan).

Setelah algoritma Dijkstra selesai, jalur terpendek dari simpul asal ke simpul tujuan dapat ditentukan dengan melacak jalur dari simpul tujuan ke simpul asal berdasarkan informasi yang disimpan selama algoritma berjalan.

Algoritma Dijkstra secara efektif digunakan dalam berbagai masalah optimisasi rute terpendek, seperti perencanaan rute dalam transportasi, routing dalam jaringan komunikasi, atau pemecahan masalah jadwal dalam logistik.

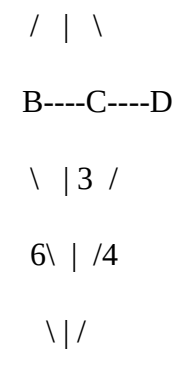
3. Untuk menentukan rute terpendek pada sebuah grafik yang memiliki bobot atau jarak antara setiap node yang berbeda menggunakan algoritma Dijkstra, Anda dapat mengikuti langkah-langkah berikut:
    1) Inisialisasi:

- Tetapkan jarak awal dari simpul asal ke semua simpul lain dalam grafik sebagai tak terhingga (kecuali simpul asal itu sendiri yang diatur sebagai 0).
- Buat sebuah himpunan kosong yang berisi simpul-simpul yang belum dikunjungi.

2. Pilih simpul saat ini sebagai simpul asal dan atur jaraknya menjadi 0.
3. Untuk setiap simpul tetangga dari simpul saat ini yang belum dikunjungi, lakukan langkah berikut:
   - Hitung jarak sementara dari simpul asal ke tetangga tersebut dengan menambahkan bobot dari jalur antara simpul saat ini dan tetangga tersebut ke jarak saat ini.
   - Jika jarak sementara yang dihitung lebih kecil daripada jarak yang saat ini diketahui untuk tetangga tersebut, perbarui jarak dengan jarak sementara yang baru.
4. Tandai simpul saat ini sebagai dikunjungi.
5. Jika masih ada simpul yang belum dikunjungi, pilih simpul dengan jarak terkecil sebagai simpul saat ini, dan ulangi langkah 3 dan 4.
6. Setelah semua simpul telah dikunjungi atau jika tidak ada simpul lain yang dapat dijangkau, jalur terpendek dari simpul asal ke simpul tujuan dapat ditentukan dengan melacak jalur dari simpul tujuan ke simpul asal berdasarkan informasi yang disimpan selama algoritma Dijkstra berjalan.

Berikut adalah contoh langkah-langkah algoritma Dijkstra untuk menentukan rute terpendek pada sebuah grafik:

Grafik:



Simpul asal: A

Simpul tujuan: E

Langkah-langkah:

a) Inisialisasi:
   - Set jarak dari A ke A = 0, A ke B = ∞, A ke C = ∞, A ke D = ∞, A ke E = ∞.
   - Simpul yang belum dikunjungi: {A, B, C, D, E}.

b) Pilih simpul saat ini: A. Atur jarak saat ini ke 0.

c) Perbarui jarak:
   - Hitung jarak sementara dari A ke B: 0 + 1 = 1. Dalam langkah ini, kita menemukan jarak terpendek dari A ke B.
   - Hitung jarak sementara dari A ke C: 0 + 5 = 5.
   - Hitung jarak sementara dari A ke D: 0 + 2 = 2.
   - Hitung jarak sementara dari A ke E: 0 + ∞ = ∞ (belum ada jalur langsung dari A ke E).

d) Tandai simpul saat ini sebagai dikunjungi. Simpul yang belum dikunjungi: {B, C,

4. Algoritma Ford-Fulkerson adalah algoritma yang digunakan untuk mencari aliran maksimum dalam grafik terarah dengan kapasitas edge yang ditentukan. Algoritma ini berbasis pada pendekatan augmenting path untuk secara bertahap meningkatkan aliran melalui grafik hingga mencapai aliran maksimum.

   Berikut adalah langkah-langkah algoritma Ford-Fulkerson:

   a. Inisialisasi: Atur aliran awal pada setiap edge dalam grafik menjadi 0.

b. Cari augmenting path: Cari jalur dari source (sumber) ke sink (tujuan) dalam grafik yang belum sepenuhnya dimanfaatkan oleh aliran saat ini. Jalur ini harus mematuhi dua kondisi:
    - Kapasitas residu (sisa kapasitas) pada setiap edge di jalur harus lebih besar dari 0.
    - Jalur harus mengikuti arah edge yang memiliki aliran positif dan melawan arah edge yang memiliki aliran negatif.
c. Hitung bottleneck capacity: Bottleneck capacity adalah kapasitas residu terkecil di antara semua edge dalam jalur augmenting yang ditemukan.
d. Perbarui aliran: Tambahkan bottleneck capacity ke setiap edge dalam jalur augmenting yang ditemukan. Ini akan meningkatkan aliran melalui jalur tersebut.
e. Perbarui kapasitas residu: Kurangi bottleneck capacity dari kapasitas residu pada setiap edge dalam jalur augmenting yang ditemukan. Tambahkan bottleneck capacity ke kapasitas residu pada edge yang melawan arah dalam jalur augmenting
f. Ulangi langkah 2-5: Ulangi langkah-langkah tersebut hingga tidak ada lagi jalur augmenting yang dapat ditemukan dalam grafik.
g. Aliran maksimum: Ketika tidak ada jalur augmenting yang tersisa, aliran maksimum telah dicapai. Aliran maksimum adalah jumlah total aliran yang melalui sink.

Algoritma Ford-Fulkerson berhenti ketika tidak ada jalur augmenting yang tersisa untuk ditemukan, yang menandakan bahwa aliran maksimum telah dicapai. Hasil akhir dari algoritma ini adalah aliran maksimum melalui grafik, yang dapat digunakan untuk mengoptimalkan penggunaan sumber daya atau mengatasi masalah pemetaan dalam grafik terarah dengan kapasitas edge.

Perlu dicatat bahwa implementasi algoritma Ford-Fulkerson dapat berbeda-beda, terutama dalam cara mencari jalur augmenting. Salah satu pendekatan yang umum digunakan adalah menggunakan algoritma DFS (Depth-First Search) untuk mencari jalur argmenting melalui grafik.

5. Algoritma jaringan dapat digunakan untuk menentukan rute pipa terpendek yang meminimalkan biaya pembangunan infrastruktur dalam proyek jaringan pipa minyak.

Dalam konteks ini, kita dapat menggunakan algoritma Dijkstra atau algoritma Bellman-Ford untuk mencari rute terpendek dengan mempertimbangkan biaya pembangunan pipa sebagai bobot pada setiap jalur.

Berikut adalah contoh penerapan algoritma Dijkstra dalam bahasa pemrograman Python untuk menentukan rute pipa terpendek dengan biaya minimal:

```python
import heapq

def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    queue = [(0, start)]
    while queue:
        cost, node = heapq.heappop(queue)
        if cost > distances[node]:
            continue
        for neighbor, neighbor_cost in graph[node].items():
            new_cost = cost + neighbor_cost
            if new_cost < distances[neighbor]:
                distances[neighbor] = new_cost
                heapq.heappush(queue, (new_cost, neighbor))
    return distances

# Contoh penggunaan
graph = {
    'A': {'B': 4, 'C': 2},
    'B': {'A': 4, 'C': 1, 'D': 5},
    'C': {'A': 2, 'B': 1, 'D': 8},
    'D': {'B': 5, 'C': 8}
}

start_node = 'A'
distances = dijkstra(graph, start_node)

print(f"Jarak terpendek dari simpul {start_node}:")
for node, distance in distances.items():
    print(f"{node}: {distance}")
```

```
Jarak terpendek dari simpul A:
A: 0
B: 3
C: 2
D: 8
```

Dalam contoh ini, kita memiliki grafik yang mewakili jaringan pipa minyak dengan beberapa simpul (misalnya A, B, C, D) dan bobot yang menunjukkan biaya pembangunan pipa antara simpul-simpul tersebut. Fungsi dijkstra

mengimplementasikan algoritma Dijkstra untuk mencari rute terpendek dari simpul awal (start) ke semua simpul lain dalam grafik. Hasilnya adalah jarak terpendek dari simpul awal ke semua simpul lain.

Dalam contoh ini, kita mencari rute terpendek dari simpul A ke semua simpul lain dalam grafik. Hasilnya akan menunjukkan jarak terpendek dari simpul A ke simpul-simpul lain dalam bentuk dictionary, di mana kunci adalah simpul tujuan dan nilainya adalah jarak terpendek.

Anda dapat memodifikasi contoh ini sesuai dengan kebutuhan proyek jaringan pipa minyak Anda, dengan mengganti grafik dengan informasi dan bobot yang relevan untuk lokasi-lokasi di Indonesia yang ingin Anda hubungkan dengan pipa minyak.