TUGAS EKSPLORASI MANDIRI

PERANCANGAN DAN ANALISIS ALGORITMA (INF1.62.4001)

“Warshall’s and Floyd’s Algorithms”

Dosen Pengampu : Randi Proska Sandra, S.Pd, M.Sc

Disusu Oleh :

Ridho Hamdani Putra

23343052

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

A. Ringkasan

1. Pendahuluan

Graf merupakan salah satu struktur data yang sangat penting dalam ilmu komputer karena banyak digunakan dalam berbagai aplikasi, seperti jaringan komunikasi, pemetaan geografis, sistem transportasi, dan analisis hubungan sosial. Graf terdiri dari simpul (vertex) dan sisi (edge), di mana hubungan antara simpul dapat berupa arah (graf berarah) atau tidak berarah (graf tak berarah) serta dapat berbobot atau tidak berbobot.

Dalam berbagai permasalahan yang melibatkan graf, dua operasi yang sering dilakukan adalah:

a. Menentukan apakah ada jalur antara dua simpul dalam graf (transitive closure).
b. Menemukan jalur terpendek antara dua simpul dalam graf berbobot.

Dua algoritma utama yang digunakan untuk menyelesaikan masalah ini adalah Algoritma Warshall dan Algoritma Floyd-Warshall. Algoritma Warshall digunakan untuk menentukan keterhubungan antar simpul dalam graf berarah (transitive closure), sedangkan Algoritma Floyd-Warshall digunakan untuk mencari jarak terpendek antara semua pasangan simpul dalam graf berbobot.

2. Algoritma Warshall

Algoritma Warshall adalah algoritma berbasis pemrograman dinamis yang digunakan untuk menghitung *transitive closure* dari suatu graf berarah. Algoritma ini memungkinkan kita mengetahui apakah terdapat jalur antara dua simpul dalam suatu graf dengan cara memperbarui matriks ketetanggaan berdasarkan hubungan transitif antar simpul.

Sebagai contoh, jika kita memiliki graf dengan hubungan:

- $A \rightarrow B$
- $B \rightarrow C$

Maka dengan *transitive closure*, kita dapat menyimpulkan bahwa A juga memiliki jalur menuju C, meskipun tidak secara langsung. Algoritma Warshall akan mendeteksi hubungan seperti ini dan memperbarui matriks ketetanggaan sehingga mencerminkan keterhubungan antar simpul secara lengkap.

Cara Kerja Algoritma Warshall

1. Representasikan graf dalam bentuk matriks ketetanggaan, dengan nilai 1 jika ada jalur langsung antara dua simpul, dan 0 jika tidak ada jalur.
2. Iterasi melalui setiap simpul sebagai simpul perantara dan periksa apakah ada jalur baru yang terbentuk melalui simpul tersebut.
3. Jika ditemukan jalur baru, perbarui matriks ketetanggaan.
4. Ulangi langkah ini hingga seluruh jalur dalam graf diperiksa.

Contoh Aplikasi Algoritma Warshall

1. Analisis Keterhubungan dalam Jaringan Sosial

   Dalam jejaring sosial seperti Facebook atau LinkedIn, Algoritma Warshall dapat digunakan untuk menentukan apakah ada jalur perantara yang menghubungkan dua individu dalam jaringan mereka.

2. Pemeriksaan Keterjangkauan dalam Basis Data Relasional

   Dalam sistem basis data, algoritma ini digunakan untuk menentukan apakah suatu entitas memiliki hubungan tidak langsung dengan entitas lain, misalnya dalam sistem manajemen perusahaan atau kepegawaian.

3. Analisis Keamanan dan Akses Kontrol

   Algoritma ini sering digunakan dalam sistem keamanan untuk menentukan apakah seorang pengguna memiliki akses ke suatu sumber daya berdasarkan hierarki izin yang diberikan.

3. Algoritma Floyd-Warshall

Algoritma Floyd-Warshall adalah algoritma berbasis pemrograman dinamis yang digunakan untuk menemukan jalur terpendek antara semua pasangan simpul dalam suatu graf berbobot. Algoritma ini sangat berguna dalam sistem navigasi, routing dalam jaringan komputer, dan optimasi transportasi.

Berbeda dengan Algoritma Dijkstra, yang hanya mencari jalur terpendek dari satu simpul ke semua simpul lainnya, Algoritma Floyd-Warshall menghitung jalur terpendek untuk setiap pasangan simpul dalam graf secara simultan. Algoritma ini bekerja dengan memperbarui matriks bobot secara bertahap berdasarkan jalur yang ditemukan melalui simpul perantara.

Cara Kerja Algoritma Floyd-Warshall

1. Representasikan graf dalam bentuk matriks bobot, dengan nilai bobot tertentu untuk jalur yang ada dan ∞ (tak hingga) untuk jalur yang tidak ada.
2. Iterasi melalui setiap simpul sebagai simpul perantara, lalu periksa apakah ada jalur yang lebih pendek melalui simpul tersebut.
3. Jika ditemukan jalur yang lebih pendek, perbarui nilai dalam matriks jarak.
4. Ulangi langkah ini hingga seluruh pasangan simpul diperiksa.

Contoh Aplikasi Algoritma Floyd-Warshall

1. Sistem Navigasi dan GPS
   Algoritma ini digunakan dalam sistem navigasi seperti Google Maps untuk menentukan jalur terpendek antara dua lokasi.
2. Optimasi Rute Transportasi
   Digunakan dalam perencanaan jaringan transportasi untuk menentukan rute tercepat dalam sistem kereta api atau penerbangan.
3. Routing dalam Jaringan Komputer

Dalam protokol routing seperti OSPF (Open Shortest Path First), algoritma ini digunakan untuk menentukan jalur optimal bagi paket data dalam jaringan internet.

B. Pseudocode

1. Algoritma Warshall

```
Warshall(Matriks):
  n ← jumlah simpul dalam Matriks

  Untuk k dari 0 hingga n-1 lakukan:
    Untuk i dari 0 hingga n-1 lakukan:
      Untuk j dari 0 hingga n-1 lakukan:
        Matriks[i][j] ← Matriks[i][j] OR (Matriks[i][k] AND Matriks[k][j])

  Kembalikan Matriks

// Program utama
Definisikan Matriks graf berbentuk adjacency (0: tidak terhubung, 1: terhubung)
Tampilkan Matriks sebelum Algoritma Warshall
Panggil Warshall(Matriks)
Tampilkan Matriks setelah Algoritma Warshall
```

2. Algoritma Floyd-Warshall

```
Fungsi Floyd_Warshall(Matriks):
  n ← jumlah simpul dalam Matriks

  Untuk k dari 0 hingga n-1 lakukan:
    Untuk i dari 0 hingga n-1 lakukan:
      Untuk j dari 0 hingga n-1 lakukan:
        Matriks[i][j] ← MIN(Matriks[i][j], Matriks[i][k] + Matriks[k][j])

  Kembalikan Matriks

// Program utama
Definisikan Matriks graf berbobot:
  - Jika tidak ada jalur langsung, gunakan nilai sangat besar (INF)

Tampilkan Matriks sebelum Algoritma Floyd-Warshall
Panggil Floyd_Warshall(Matriks)
Tampilkan Matriks setelah Algoritma Floyd-Warshall
```

C. Source Code

1. Algoritma Warshall

```python
def warshall(matriks):
    """Algoritma Warshall untuk menentukan keterhubungan antar simpul"""
    n = len(matriks)
    for k in range(n):
        for i in range(n):
            for j in range(n):
                matriks[i][j] = matriks[i][j] or (matriks[i][k] and matriks[k][j])

# Contoh graf (1 jika ada jalur, 0 jika tidak)
graf = [
    [1, 1, 0],
    [0, 1, 1],
    [1, 0, 1]
]

print("Matriks sebelum Algoritma Warshall:")
for baris in graf:
    print(baris)

warshall(graf)

print("\nMatriks setelah Algoritma Warshall:")
for baris in graf:
    print(baris)

```

Output

```
PS D:\KULIAH> & C:/Users/ridho/AppData/Local/Programs/Python
an Analisis Algoritma/Tugas/Ridho Hamdani Putra_Warshall's a
a Warshall/AlgoritmaWarshall.py"
Matriks sebelum Algoritma Warshall:
[1, 1, 0]
[0, 1, 1]
[1, 0, 1]

Matriks setelah Algoritma Warshall:
[1, 1, 1]
[1, 1, 1]
[1, 1, 1]
PS D:\KULIAH>
```

Program ini menerapkan Algoritma Warshall untuk menentukan keterhubungan antar simpul dalam graf berarah menggunakan matriks ketetanggaan. Jika ada jalur tidak langsung antara dua simpul melalui simpul lain, program akan memperbaruinya.

2. Algoritma Floyd-Warshall

```python
def floyd_warshall(matriks):
    """Algoritma Floyd-Warshall untuk mencari jarak terpendek antar simpul"""
    n = len(matriks)
    for k in range(n):
        for i in range(n):
            for j in range(n):
                matriks[i][j] = min(matriks[i][j], matriks[i][k] + matriks[k][j])

# Gunakan angka besar untuk menunjukkan tidak ada jalur langsung
INF = 99999
graf = [
    [0, 3, INF, 7],
    [8, 0, 2, INF],
    [5, INF, 0, 1],
    [2, INF, INF, 0]
]

print("Matriks sebelum Algoritma Floyd-Warshall:")
for baris in graf:
    print(baris)

floyd_warshall(graf)

print("\nMatriks setelah Algoritma Floyd-Warshall:")
for baris in graf:
    print(baris)

```

Output

```
PS D:\KULIAH> & C:/Users/ridho/AppData/Local/Programs/Python/Python31
an Analisis Algoritma/Tugas/Ridho Hamdani Putra_Warshall's and Floyd'
a Floyd Warshall/AlgoritmaFloydWarshall.py"
Matriks sebelum Algoritma Floyd-Warshall:
[0, 3, 99999, 7]
[8, 0, 2, 99999]
[5, 99999, 0, 1]
[2, 99999, 99999, 0]

Matriks setelah Algoritma Floyd-Warshall:
[0, 3, 5, 6]
[5, 0, 2, 3]
[3, 6, 0, 1]
[2, 5, 7, 0]
PS D:\KULIAH>
```

Program ini menerapkan Algoritma Floyd-Warshall untuk mencari jalur terpendek antara semua pasangan simpul dalam graf berbobot.

D. Analisis Kebutuhan Waktu

1. Analisis Berdasarkan Operator Assignment dan Aritmatika

a. Algoritma Warshall

1. Operator Assignment

Di Luar Loop	Di Dalam Loop
Tidak ada	matriks[i][j] = matriks[i][j] OR (matriks[i][k] AND matriks[k][j])

Total Assignment = (n^3) a
Karena ada 3 loop bersarang jadi total jumlah operasinya = n^3

2. Operator Aritmatika

Di Luar Loop	Di Dalam Loop
Tidak ada	Operasi logika OR, AND

Total Aritmatika = (n^3) b
Karena ada 3 loop bersarang jadi total jumlah operasinya = n^3

Jadi total persamaan waktu = (n^3)a+(n^3)b

b. Algoritma Floyd-Warshall

1. Operator Assignment

Di Luar Loop	Di Dalam Loop
Tidak ada	matriks[i][j] = min(matriks[i][j], matriks[i][k] + matriks[k][j])

Total Assignment = (n^3) a
Karena ada 3 loop bersarang jadi total jumlah operasinya = n^3

2. Operator Aritmatika

Di Luar Loop	Di Dalam Loop
Tidak ada	Operasi penjumlahan (+) dan perbandingan (min())

Total Aritmatika = (n^3) b
Karena ada 3 loop bersarang jadi total jumlah operasinya = n^3

Jadi total persamaan waktu = (n^3)a+(n^3)b

2. Analisis Berdasarkan Jumlah Operasi Abstrak

- Warshall: Operasi khas adalah logika (or, and), yang dilakukan n^3 kali → Kompleksitas $O(n^3)$.
- Floyd-Warshall: Operasi khas adalah aritmatika (+, min), yang juga dilakukan n^3 kali → Kompleksitas $O(n^3)$.

3. Analisis operator assignment dan aritmatika

a. Best-Case (Kasus Terbaik)

- Warshall: Tidak ada perubahan dalam matriks, artinya hubungan antar simpul sudah lengkap sejak awal. Semua operasi tetap dilakukan, sehingga kompleksitas tetap $O(n^3)$.
- Floyd-Warshall: Tidak ada perubahan jalur terpendek (misalnya jika semua bobot sudah optimal dari awal). Namun, karena tetap harus memeriksa semua pasangan simpul, kompleksitas tetap $O(n^3)$.

b. Worst-Case (Kasus Terburuk)

- Warshall: Semua elemen dalam matriks diperbarui, artinya setiap hubungan antar simpul berubah. Hal ini tetap membuat kompleksitas $O(n^3)$.
- Floyd-Warshall: Semua jalur optimal harus diperbarui, menyebabkan setiap iterasi melakukan perhitungan dan pembaruan nilai. Kompleksitas tetap $O(n^3)$.

c. Average-Case (Kasus Rata-Rata)

- Dalam kasus rata-rata, sebagian besar elemen matriks diperbarui.
- Karena algoritma tetap harus melakukan pengecekan pada semua pasangan simpul, kompleksitas tetap $O(n^3)$.

E. Referensi

GeeksforGeeks. (2025). Floyd Warshall Algorithm | Dynamic Programming (DP-16). *GeeksforGeeks*. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

Levitin, A. (2012). *Introduction to The Design And Analysis of Algorithms 3rd Edition*.

Programiz. (2025). Floyd-Warshall Algorithm. *Programiz*. https://www.programiz.com/dsa/floyd-warshall-algorithm

F. Link Github

https://github.com/RidhoHamdaniPutra/Tugas-Perancangan-dan-Analisis-Algoritma