

LAPORAN TUGAS BESAR

***STUDI EFEKTIVITAS ALGORITMA DFS (Depth First Search) MENGGUNAKAN
PENDEKATAN REKURSIF DAN ITERATIF***

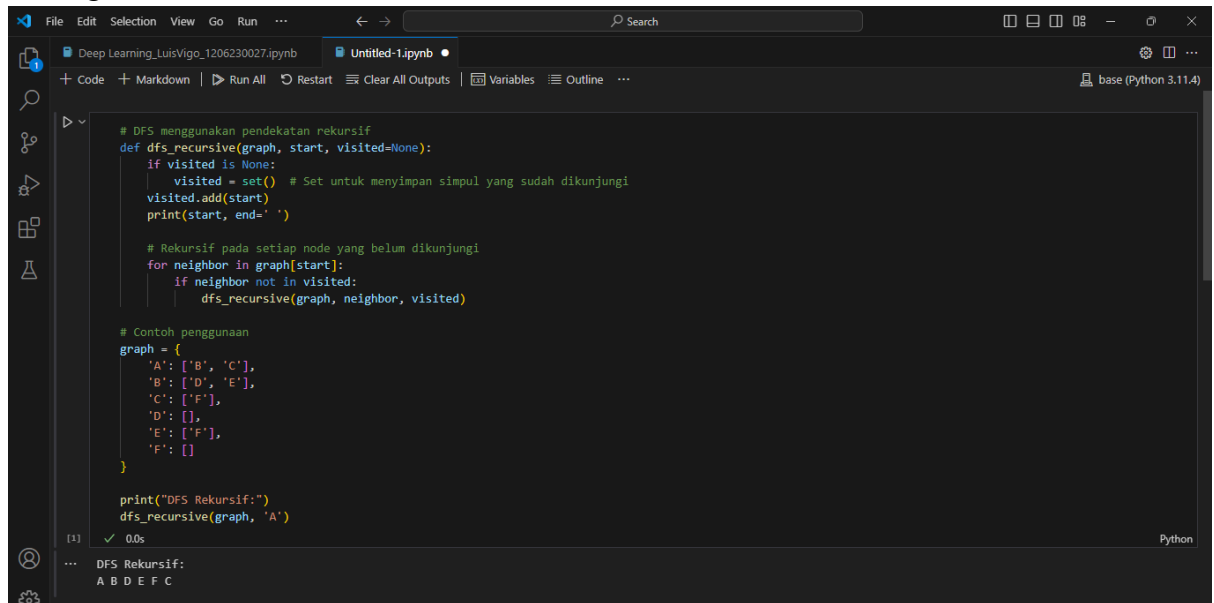


Di susun oleh :

1. Luis Vigo (1206230027)
2. Adi Maulana Ishaq (1206230039)

MATA KULIAH ANALISIS KOMPLEKSITAS ALGORITMA
PROGRAM STUDI SAINS DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY SURABAYA

1. Codingan Rekursif



```
# DFS menggunakan pendekatan rekursif
def dfs_recursive(graph, start, visited=None):
    if visited is None:
        visited = set() # Set untuk menyimpan simpul yang sudah dikunjungi
        visited.add(start)
        print(start, end=' ')

    # Rekursif pada setiap node yang belum dikunjungi
    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs_recursive(graph, neighbor, visited)

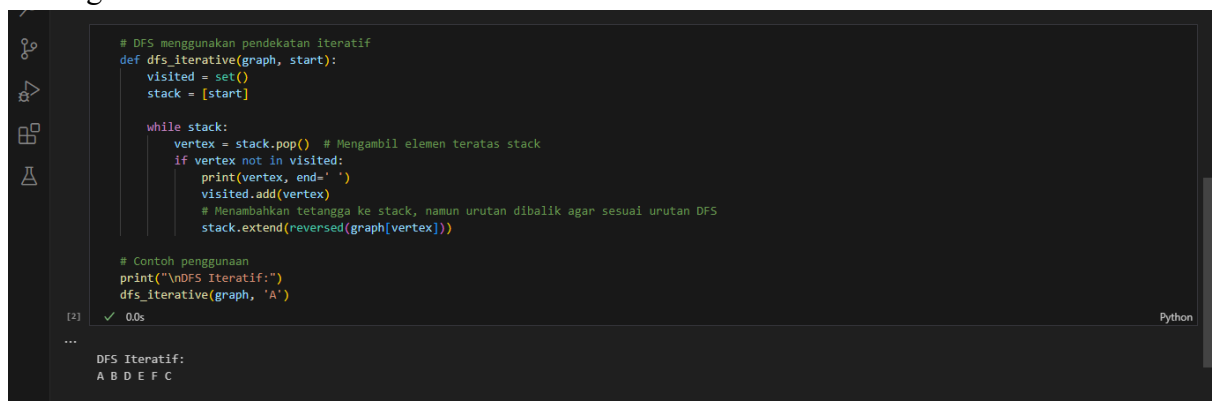
# Contoh penggunaan
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

print("DFS Rekursif:")
dfs_recursive(graph, 'A')
```

DFS Rekursif:
A B D E F C

Pendekatan rekursif dalam DFS menggunakan fungsi yang memanggil dirinya sendiri untuk menjelajahi setiap node atau simpul.

2. Codingan Iteratif



```
# DFS menggunakan pendekatan iteratif
def dfs_iterative(graph, start):
    visited = set()
    stack = [start]

    while stack:
        vertex = stack.pop() # Mengambil elemen teratas stack
        if vertex not in visited:
            print(vertex, end=' ')
            visited.add(vertex)
            # Menambahkan tetangga ke stack, namun urutan dibalik agar sesuai urutan DFS
            stack.extend(reversed(graph[vertex]))

# Contoh penggunaan
print("\nDFS Iteratif:")
dfs_iterative(graph, 'A')
```

DFS Iteratif:
A B D E F C

Pendekatan iteratif menggunakan stack eksplisit untuk mengelola urutan penjelajahan simpul.

3. Kompleksitas Iteratif

- Kompleksitas Waktu (Time Complexity)
Kompleksitas waktu algoritma DFS Iteratif dihitung berdasarkan jumlah simpul (V) dan sisi atau jalur (E) yang terdapat dalam graf.
 - ✓ Menjelajahi Simpul : Algoritma DFS Iteratif Mengunjungi setiap simpul dalam graf sekali saja. untuk sebuah graf dengan V simpul, waktu yang di butuhkan untuk proses ini adalah $O(V)$

- ✓ Menjelajahi Sisi/Jalur : setiap sisi atau jalur yang terhubung dengan simpul juga akan di proses sekali untuk mengecek tetangga atau menambahkan tetangga ke dalam stack. Waktu yang di perlukan untuk memproses sisi adalah $O(E)$

✚ Total Kompleksitas Waktu : $O(V + E)$

- Kompleksitas Ruang (Space Complexity)
 - ✓ Struktur Stack : algoritma menggunakan stack eksplisit untuk menyimpan simpul yang perlu di eksplorasi . pada graf dengan kedalaman d (derajat simpul) dan jumlah simpul maksimal V , stack dapat menampung hingga d simpul sekaligus.
 - ✓ Simpul yang di kunjungi : DFS menggunakan struktur data seperti set() atau array [] untuk melacak simpul yang sudah di kunjungi , dengan ukuran $O(V)$, karena setiap simpul akan di catat 1 kali.

✚ Total Kompleksitas Ruang : $O(V + d)$

Karena $d \leq V$, maka kompleksitas ruang dapat di sederhanakan menjadi $O(V)$

4. Kompleksitas Rekursif

- Kompleksitas Waktu (Time Complexity)
DFS Rekursif mengunjungi setiap simpul dan setiap sisi di graf. Untuk setiap simpul :
 - Fungsi rekursif di panggil untuk semua tetangganya (simpul yang terhubung dengan sisi)
 - Setiap simpul hanya di kunjungi satu kali untuk menghindari pengulangan (dengan melacak simpul yang telah di kunjungi menggunakan struktur data seperti set ()).
- Analisis
 1. Menjelajahi Simpul : setiap graf memiliki V simpul , DFS akan memproses setiap simpul sekali, dengan kompleksitas adalah $O(V)$.
 2. Menjelajahi sisi : DFS akan mengunjungi semua sisi yang terhubung dengannya . jika graf memiliki E sisi , DFS akan memproses masing – masing sisi sekali. Dengan kompleksitas $O(E)$

✚ Total Kompleksitas Waktu : $O(V + E)$

- Kompleksitas Ruang (Space Complexity)

Kompleksitas Ruang DFS Rekursif bergantung pada :

- Struktur Data untuk melacak simpul yang di kunjungi :
 1. DFS membutuhkan struktur data seperti set atau array untuk melacak simpul yang telah di kunjungi agar tidak memproses simpul yang sama berulang kali.
 2. Ukuran struktur data ini adalah $O(V)$, karena setiap simpul hanya di catat satu kali.
- Stack rekursif :
 1. DFS Rekursif memanfaatkan stack bawaan sistem untuk menyimpan informasi setiap kali fungsi rekursif di panggil.
 2. Dalam graf dengan kedalaman maksimal d , stack rekursif dapat menampung hingga d pemanggilan fungsi secara bersamaan.
 3. Jika $d = V$ (kedalaman maksimum sama dengan jumlah simpul) kompleksitas ruang untuk stack adalah $O(V)$.

🚦 Total Kompleksitas Ruang : $O(V)$

- Substitusi

- ✓ DFS Rekursif untuk graf memiliki hubungan rekursif :

$$T(V, E) = O(V) + O(E)$$

Karena setiap simpul (V) di kunjungi sekali , dan setiap sisi (E) di jelajahi sekali.

- ✓ Hubungan Rekursif standar DFS

$$\text{Pada setiap simpul : } T(n) = T(\text{Tetangga}) + O(1)$$

Misalkan graf memiliki d tetangga rata rata per simpul , maka :

$$T(V) = V + d + T(V - 1)$$

- ✓ Tebak Solusi

Bentuk solusi umum untuk rekursi adalah $T(V) = cV + k$, dimana c dan k adalah konstanta : $T(V) = O(V) + O(E)$

- ✓ Verifikasi Substitusi

Karena setiap simpul hanya di proses sekali ($O(V)$) dan setiap sisi hanya di jelajahi sekali ($O(E)$) , tebakan $T(V) = O(V + E)$ terbukti benar .

- Persamaan Karakteristik
 - ✓ Hubungan Algoritma DFS Rekursif :

$$T(V, E) = O(V) + O(E)$$

Namun untuk melihat DFS yang memanggil dirinya sendiri untuk setiap tetangga suatu simpul :

$$T(n) = T(n-1) + O(d)$$

Di mana :

- $T(n)$ adalah kompleksitas waktu untuk memproses n simpul
- d adalah derajat (jumlah tetangga) rata rata simpul
- $n-1$ adalah jumlah simpul yang harus di proses oleh DFS pada graf.

- ✓ Menentukan Persamaan Karakteristik
 Persamaan karakteristik digunakan untuk mengidentifikasi pola dan solusi dari hubungan rekursif. Untuk masalah ini, dapat menulis persamaan rekursif DFS yang lebih umum.

$$T(V) = T(V-1) + O(d)$$

Karena dalam setiap langkah rekursif untuk memproses satu simpul dan menjelajahi tetangga tetangganya. Namun, dalam konteks ini, $O(d)$ (jumlah tetangga rata rata per simpul), cenderung menjadi konstanta yang bergantung pada graf tertentu.

- ✓ Solusi persamaan rekursif
 untuk menyelesaikan persamaan dengan mencoba mengulang rekursi untuk mencari pola.

1. Substitusi Pertama : $T(n) = T(n-1) + O(d)$

2. Substitusi Kedua : $T(n-1) = T(n-2) + O(d)$

Ganti $T(n-1)$ pada persamaan pertama sehingga menjadi :

$$T(n) = T(n-2) + 2O(d)$$

3. Ganti iterasi berikutnya sehingga mencapai $T(0)$:

$$T(n) = T(0) + nO(d)$$

Karena $T(0)$ adalah kondisi dasar yang biasanya bernilai konstan $O(1)$, maka

$$T(n) = O(n \cdot d)$$

✓ Kompleksitas Waktu Total DFS Rekursif

Karena d pada graf sering kali tidak terlalu besar dan cenderung konstan maka kompleksitas waktu secara keseluruhan untuk DFS adalah

$$T(V, E) = O(V + E)$$

✓ Kesimpulan

Persamaan karakteristik di gunakan untuk menyelesaikan hubungan rekursif dengan menuliskan kembali persamaan tersebut dalam bentuk yang lebih terstruktur agar lebih mudah di pahami dalam kasus DFS rekursif

$$T(V, E) = O(V + E)$$

Ini menunjukkan bahwa kompleksitas waktu dari DFS adalah sebanding dengan jumlah simpul V dan sisi E dalam graf .

• Peubah Variabel

Dalam analisis kompleksitas rekursif, **peubah variabel** merujuk pada variabel yang digunakan untuk menggambarkan ukuran masalah yang sedang dianalisis dan sering kali berubah selama proses rekursif. Peubah variabel ini memainkan peran penting dalam membangun hubungan rekursif, serta dalam proses penyelesaian dan analisis kompleksitas algoritma.

➤ Peubah Variabel dalam DFS Rekursif

1. V : Jumlah simpul (vertices) dalam graf.

Peran dalam rekursi : Variabel ini menggambarkan ukuran masalah pada setiap langkah rekursif. Misalnya, saat DFS memanggil dirinya sendiri pada simpul tetangga, jumlah simpul yang belum dikunjungi akan berkurang, sehingga ukuran masalah secara keseluruhan akan berkurang.

2. E : Jumlah sisi dalam graf

Peran dalam Rekursi : Sisi (E) menggambarkan hubungan antar simpul dalam graf. DFS mengunjungi setiap sisi sekali untuk mengeksplorasi tetangga dari setiap simpul. Dalam analisis kompleksitas, sisi ini berkontribusi pada total waktu yang diperlukan DFS untuk menyelesaikan pencarian.

3. d : derajat rata rata simpul

Peran dalam Rekursi : Variabel ini menggambarkan rata-rata jumlah tetangga yang dimiliki oleh setiap simpul. Dalam beberapa graf, d dapat dianggap sebagai konstanta yang menggambarkan kepadatan graf. Derajat rata-rata ini digunakan untuk menghitung kompleksitas rekursif karena berhubungan dengan berapa banyak simpul yang akan dijelajahi oleh DFS pada setiap langkah rekursif.

4. n : di gunakan untuk mengganti V , jumlah simpul dalam graf

Peran dalam Rekursi : n menunjukkan ukuran masalah dalam hal jumlah simpul. Pada setiap langkah rekursif, DFS mengurangi ukuran masalah dengan mengunjungi satu simpul dan memanggil rekursi pada tetangga simpul tersebut.