

**QUIS PERANCANGAN DAN ANALISIS ALGORITMA**  
**“ALGORITMA BRUTE FORCE DAN EXHAUTIVE SEARCH**  
**DENGAN MENGGUNAKAN METODE DEPTH FIRST SEARCH”**

“Disusun sebagai tugas pada mata kuliah Perancangan dan Analisis Algoritma , dengan Dosen Randi Proska Sandra, M.Sc dan Widya Darwin S.Pd., M.Pd.T”



**Disusun Oleh :**

**IFDAL LISYUKRI**  
**21346012**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**JURUSAN TEKNIK ELEKTRONIKA**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS NEGERI PADANG**  
**TAHUN 2022**

## **A. METODE ALGORITMA DEPTH FIRST SEARCH DALAM STRATEGI ALGORITMA BRUTE FORCE DAN EXHAUTIVE SEARCH**

Algoritma Brute Force dan Exhaustive Search adalah suatu metode penyelesaian masalah dengan cara mencoba semua kemungkinan solusi secara berurutan dan memilih solusi yang tepat. Metode ini dapat digunakan untuk menyelesaikan berbagai masalah seperti pencarian, pemrosesan data, dan optimasi.

Dalam algoritma Brute Force, semua kemungkinan solusi yang mungkin akan dicoba satu per satu tanpa ada upaya untuk menghindari solusi yang tidak memungkinkan atau tidak optimal. Algoritma ini dapat menghasilkan solusi yang tepat, namun waktu dan sumber daya yang dibutuhkan untuk mencari solusi seringkali sangat besar dan kompleksitasnya tinggi. Oleh karena itu, algoritma Brute Force seringkali tidak efisien untuk menyelesaikan masalah yang kompleks.

Sementara itu, algoritma Exhaustive Search adalah suatu metode penyelesaian masalah dengan cara mencari semua kemungkinan solusi secara sistematis dan memilih solusi yang tepat. Metode ini juga dapat digunakan untuk menyelesaikan berbagai masalah seperti pencarian, pemrosesan data, dan optimasi. Namun, algoritma Exhaustive Search biasanya lebih efisien daripada algoritma Brute Force karena biasanya mencoba solusi yang memungkinkan saja dan tidak mencoba solusi yang tidak mungkin atau tidak optimal.

Meskipun kedua metode ini memiliki kelemahan dalam hal waktu dan kompleksitas, terkadang keduanya masih digunakan untuk menyelesaikan masalah tertentu yang membutuhkan akurasi dan ketelitian yang tinggi. Namun, penggunaan kedua metode ini harus dipertimbangkan dengan baik tergantung pada kompleksitas masalah yang akan diselesaikan dan waktu serta sumber daya yang tersedia.

Depth First Search (DFS) atau Pencarian dalam Kedalaman adalah suatu algoritma yang digunakan untuk menjelajahi dan mencari solusi pada sebuah struktur data graph atau pohon (tree) dengan mengunjungi setiap simpul (node) dan menjelajahi setiap cabang sampai ke ujung sebelum kembali ke simpul sebelumnya.

Proses DFS dimulai dengan memilih simpul awal (start node) pada graph atau pohon dan mengunjunginya. Kemudian, DFS melanjutkan proses dengan menjelajahi setiap simpul yang berdekatan (adjacent node) dari simpul yang sedang dikunjungi secara rekursif hingga ditemukan simpul tujuan (goal node) atau sampai tidak ada simpul lagi yang dapat dikunjungi. Proses ini dilakukan secara berulang hingga semua simpul yang dapat dijangkau telah dikunjungi.

Berikut adalah contoh penggunaan algoritma Depth First Search dalam pemrograman python:

## B. CONTOH PSEUDOCODE DARI ALGORITMA DEPTH FIRST SEARCH

Pseudocode DFS

(Mencari rute tersingkat antara node A ke node F)

Implementasi algoritma Brute Force dengan metode Depth First Search

**Deklarasi :**

```
awal = 'A'
tujuan = 'F'
path = DFSBruteForce(pola, awal, tujuan)
```

**Algoritma:**

```
def DFSBruteForce(pola, awal, tujuan):
    # Inisialisasi stack untuk DFS
    stack = [(awal, [awal])]

    # Loop hingga stack kosong
    while stack:
        (node, path) = stack.pop()    # ambil node dan path terakhir
        for next_node in pola[node]:
            if next_node == tujuan:
                # jika node tujuan ditemukan, tambahkan ke path dan kembalikan
                return path + [next_node]
            else:
                # tambahkan node yang belum dikunjungi ke stack
                if next_node not in path:
                    stack.append((next_node, path + [next_node]))

    # jika tidak ditemukan path, return None
    return None
```

**Output:**

```
if path:
    print(f"Pola tersingkat dari {awal} ke {tujuan} adalah {path}")
else:
    print(f"Tidak ditemukan path dari {awal} ke {tujuan}")
```

### C. CONTOH PROGRAM TENTANG ALGORITMA DEPTH FIRST SEARCH

```
# Implementasi algoritma Brute Force dengan metode Depth First Search

def DFSBruteForce(pola, awal, tujuan):
    # Inisialisasi stack untuk DFS
    stack = [(awal, [awal])]

    # Loop hingga stack kosong
    while stack:
        (node, path) = stack.pop() # ambil node dan path terakhir
        for next_node in pola[node]:
            if next_node == tujuan:
                # jika node tujuan ditemukan, tambahkan ke path dan kembalikan
                return path + [next_node]
            else:
                # tambahkan node yang belum dikunjungi ke stack
                if next_node not in path:
                    stack.append((next_node, path + [next_node]))

    # jika tidak ditemukan path, return None
    return None

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

awal = 'A'
tujuan = 'F'

path = DFSBruteForce(pola, awal, tujuan)

if path:
    print(f"Pola tersingkat dari {awal} ke {tujuan} adalah {path}")
else:
    print(f"Tidak ditemukan path dari {awal} ke {tujuan}")
```

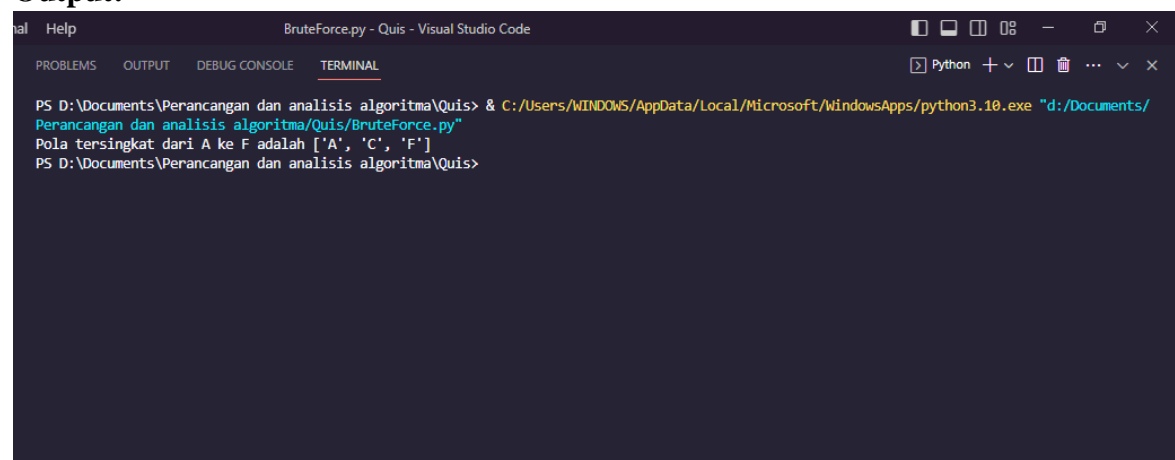
Program di atas merupakan sebuah implementasi dari algoritma Brute Force dengan menggunakan metode Depth First Search (DFS) dalam bahasa Python. Tujuan dari

program ini adalah untuk mencari path atau jalur terpendek dari suatu node awal ke suatu node tujuan pada sebuah graf yang direpresentasikan dalam bentuk dictionary.

Fungsi **DFSBruteForce** merupakan implementasi dari algoritma DFS dengan menggunakan metode Brute Force. Fungsi ini menerima tiga parameter yaitu **graph**, **start**, dan **goal**. **graph** merupakan representasi graf dalam bentuk dictionary, **start** merupakan node awal, sedangkan **goal** merupakan node tujuan. Fungsi ini akan mengembalikan path atau jalur terpendek dari node awal ke node tujuan jika ditemukan, atau None jika tidak ditemukan.

Contoh penggunaan dari program di atas adalah untuk mencari path dari node 'A' ke node 'F' pada sebuah graf yang direpresentasikan dalam dictionary. Program ini akan mencetak path terpendek dari node 'A' ke node 'F' jika ditemukan, atau mencetak pesan bahwa tidak ditemukan path jika tidak ditemukan.

## Output:



```
BruteForce.py - Quis - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python + - 
PS D:\Documents\Perancangan dan analisis algoritma\Quis> & C:/Users/WINDOWS/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/Documents/Perancangan dan analisis algoritma/Quis/BruteForce.py"
Pola tersingkat dari A ke F adalah ['A', 'C', 'F']
PS D:\Documents\Perancangan dan analisis algoritma\Quis>
```

***REFERENSI:***

(GPT, Chat, 2023), “*Brute Force in Python*”, <https://chat.openai.com/chat> , diakses pada 18 maret 2023.

***LINK GITHUB:***

<https://github.com/ifdall26/Quis/tree/main/Quis>