

1221604 - Arman Dwi Pangestu

Dasar Kecerdasan Artificial:

Perbandingan Empiris Algoritma Uninformed Search: BFS, DFS, dan UCS

Agenda

- Latar Belakang
- Rumusan Masalah
- Tinjauan Teori
- Analisis dan Implementasi
- Studi Kasus dan Hasil
- Kesimpulan dan Saran
- Penerapan Pada Dunia Nyata
- Penutup

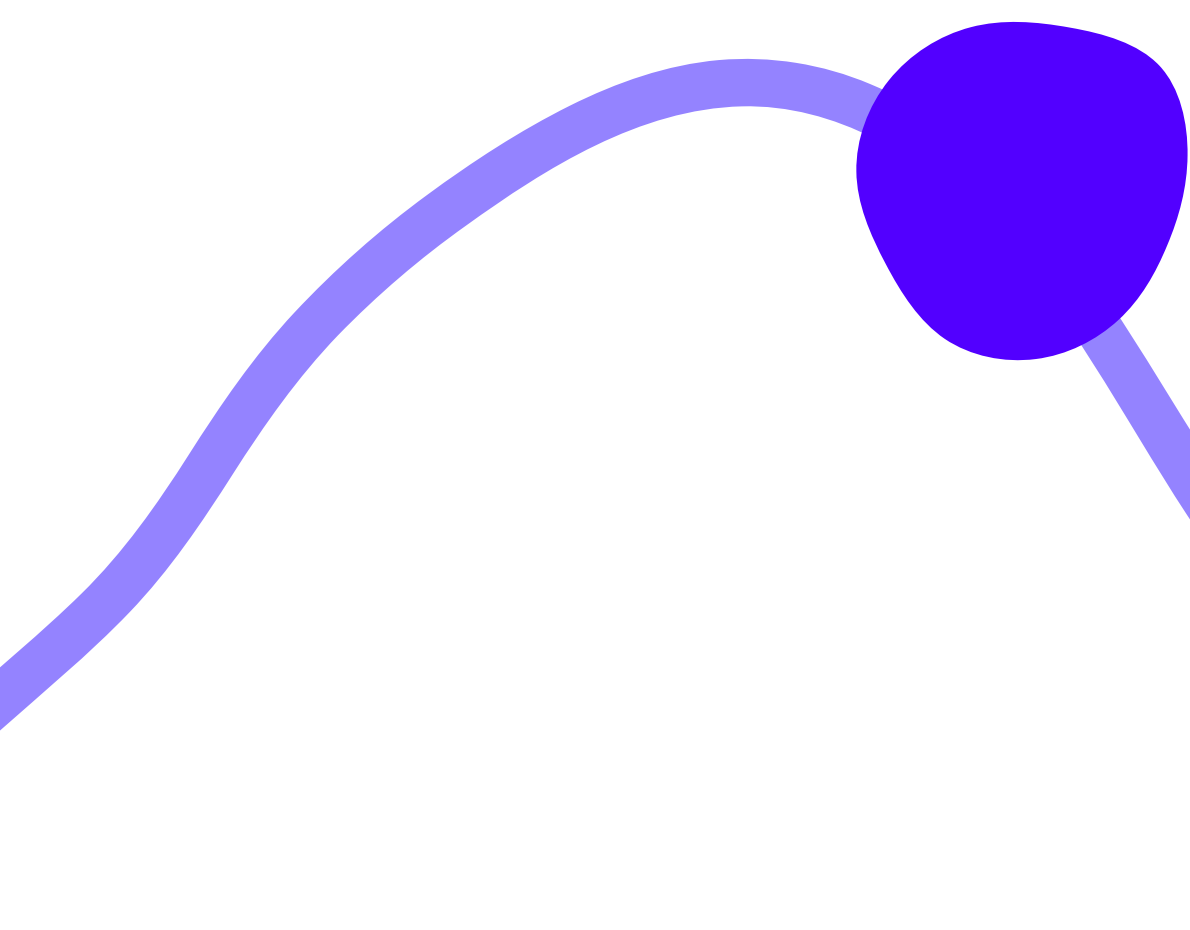
Latar Belakang

Pencarian (search) merupakan dasar dari kecerdasan buatan (AI). Banyak permasalahan AI dapat diformulasikan sebagai pencarian solusi dalam ruang keadaan (state space). Uninformed Search bekerja tanpa informasi tambahan (heuristic), hanya berdasarkan definisi masalah. Algoritma dasar yang termasuk dalam kategori ini adalah:

- Breadth-First Search (BFS)
- Depth-First Search (DFS)
- Uniform-Cost Search (UCS)

Perbandingan ketiganya penting untuk memahami efisiensi dan optimalitas dalam pemecahan masalah.

Rumusan Masalah?

- 
1. Bagaimana perbedaan performa antara BFS, DFS, dan UCS?
 2. Algoritma mana yang paling efisien berdasarkan waktu eksekusi dan kompleksitas ruang?
 3. Dalam kondisi apa masing-masing algoritma lebih sesuai digunakan?

Tinjauan Teori - Konsep Dasar

- **State Space:** Representasi semua kemungkinan keadaan dari suatu masalah.
- **Initial State:** Kondisi awal
- **Goal State:** Kondisi tujuan
- **Actions:** Langkah yang dapat dilakukan
- **Path Cost:** Biaya untuk mencapai suatu keadaan

Jenis Algoritma Uninformed Search

Algoritma	Struktur Data	Ciri Utama	Optimal	Lengkap
BFS	Queue (FIFO)	Menjelajah level demi level	Ya (jika cost sama)	Ya
DFS	Stack (LIFO)	Menyelam sedalam mungkin dulu	Tidak	Tidak
UCS	Priority Queue	Berdasarkan total biaya terendah	Ya	Ya

Analisis dan Implementasi

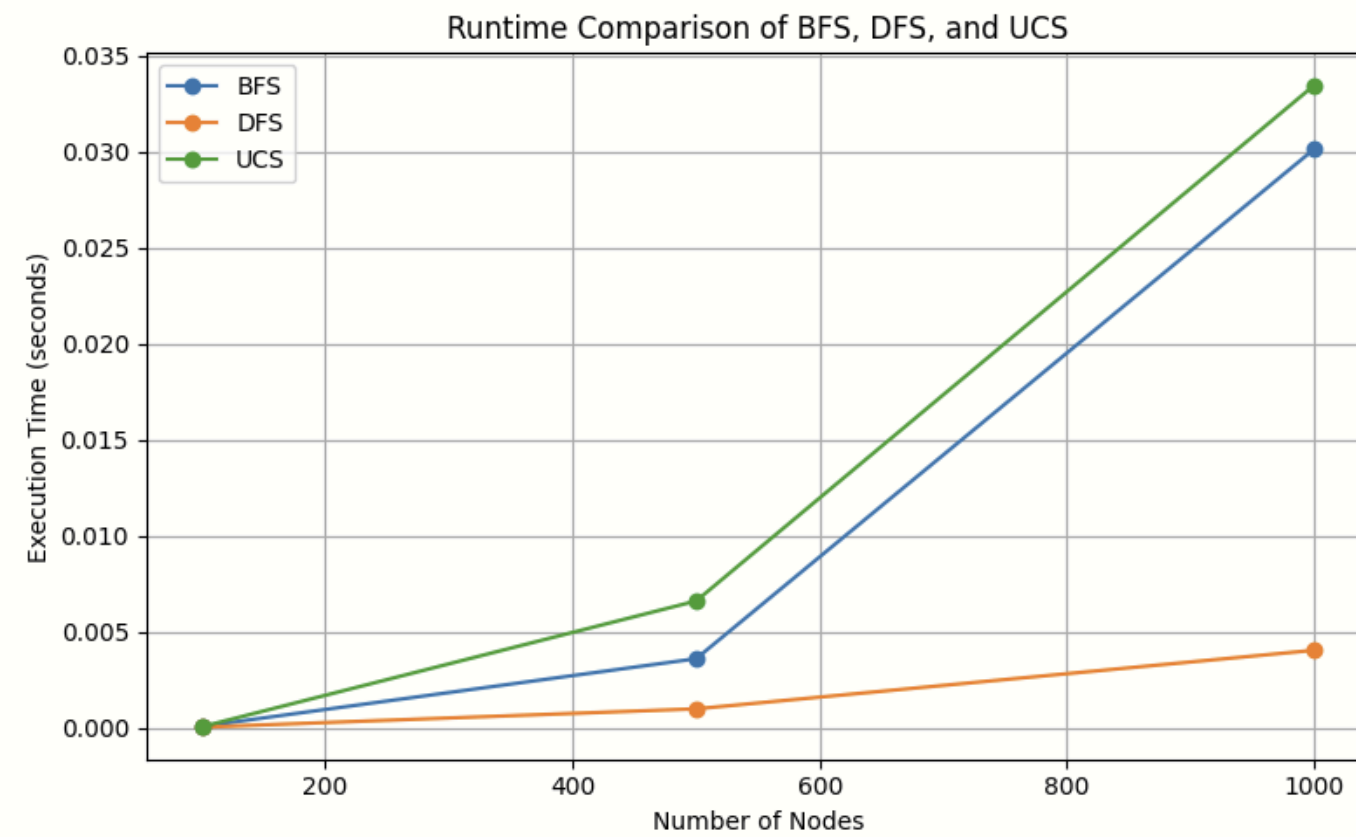
- Bahasa Pemrograman yang digunakan: **Python 3**
- Library:
 - **networkx** - berfungsi untuk membuat graph acak
 - **matplotlib** - berfungsi untuk membuat visualisasi grafik
 - **time, heapq, deque** - membantu pengurukan runtime dan struktur data
- Implementasi algoritma dibuat dari nol (from scratch) untuk:
 - **bfs.py** - eksplorasi level by level
 - **dfs.py** - eksplorasi mendalam di satu cabang
 - **ucs.py** - eksplorasi biaya minimum
- Pengujian dilakukan pada graph acak dengan **100, 500, dan 1000** node

Studi Kasus dan Hasil

Hasil Eksperimen

Nodes	BFS (s)	DFS (s)	UCS (s)
100	7.7486038208 00781e-05	3.7670135498 046875e-05	5.960464477 5390625e-05
500	0.003600358 9630126953	0.000990390 7775878906	0.0066046714 78271484
1000	0.0301244258 88061523	0.004039764 404296875	0.0334441661 8347168

Hasil Visualisasi

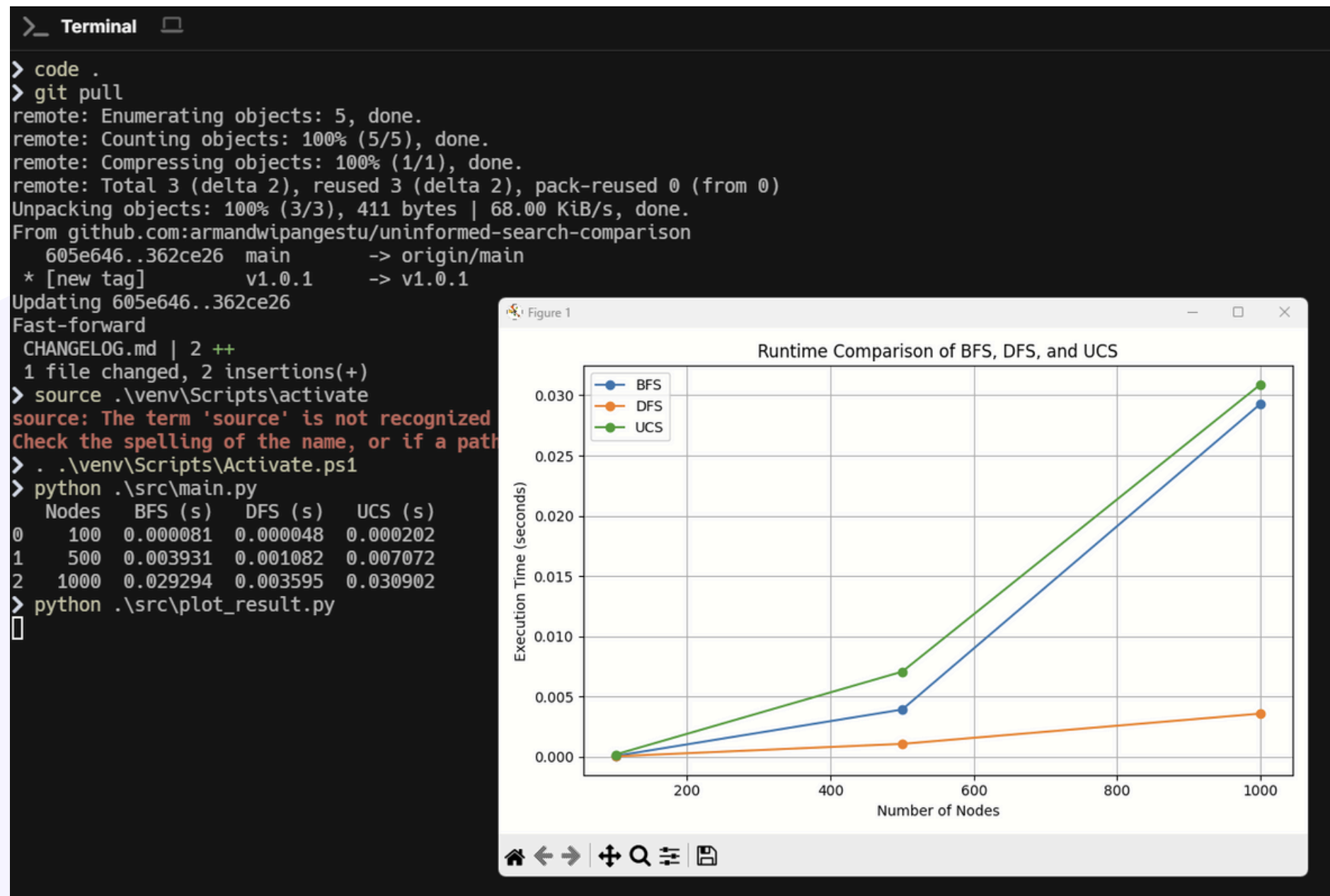


Interpretasi:

- **DFS paling cepat** - hanya fokus pada satu jalur pencarian, namun belum tentu optimal
- **BFS sedikit lebih lambat** - mengeksplor semua node di tiap level
- **UCS paling lambat** - karena perlu menghitung biaya total tiap langkah agar hasil optimal
- **Semakin besar jumlah node** - maka waktu eksekusi meningkat untuk semua algoritma.

Studi Kasus dan Hasil

Hasil Running



Kesimpulan dan Saran

Algoritma	Kelebihan	Kekurangan
BFS	Menjamin solusi optimal (equal cost), pencarian lengkap	Membutuhkan memory lebih banyak
DFS	Cepat dan hemat memory	Titdak optimal, dan berpotensi loop tak terhingga
UCS	Selalu menemukan solusi dengan biaya minimum	Lambat karena banyak perhitungan

- Pilih algoritma tergantung kebutuhan:
 - Jika ingin jalur terpendek (step minimum) – **gunakan BFS**
 - Jika ingin eksekusi cepat dan ringan – **gunakan DFS**
 - Jika ingin biaya total minimum – **gunakan UCS**

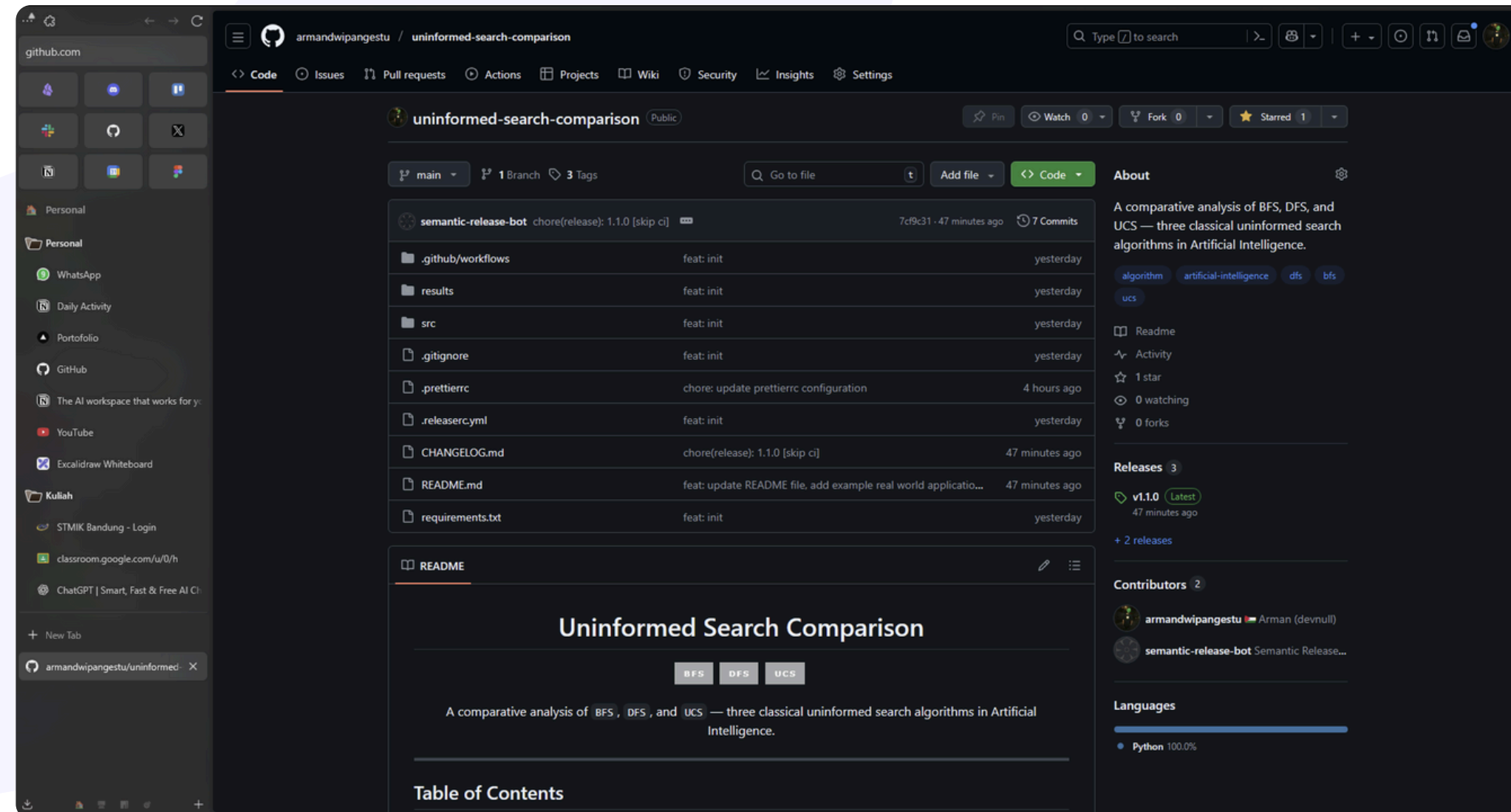
Penerapan Pada Dunia Nyata

Algoritma	Bidang	Contoh Penerapan	Deskripsi
BFS	Game & Puzzle	Pathfinding (Pac-Man), 8-Puzzle	Menemukan jalur terpendek
DFS	Maze & Compiler	Penyelesaian labirin, parsing Abstract Syntax Tree (AST)	Eksplorasi mendalam cepat dan
UCS	Navigasi & Robotika	Google Maps, robot path planning	Menemukan jalur dengan biaya termurah

- Penjelasan Singkat:
 - BFS cocok untuk **masalah dengan biaya seragam (uniform cost)**.
 - DFS cocok untuk **pencarian cepat di ruang kecil**.
 - UCS cocok untuk **masalah berbobot seperti navigasi, logistik, dan robotika**.

Terima Kasih

11



“Tidak ada algoritma yang terbaik untuk semua kasus. Pilih algoritma berdasarkan kebutuhan, kecepatan, memory, atau optimalitas.” – Arman Dwi Pangestu

<https://github.com/armandwipangestu/uninformed-search-comparison>