Perbandingan algoritma DFS dan BFS dalam mencari jalur tercepat

Nama anggota :
Randi Hasan Saksono || 1201230022
M. Dliiya'ul Haq Shidqey || 1201230024
Ahmad Naufal Ubaidillah Qisthi || 1201230006
Rizky Putra Ananda || 1201230029

A. Deskripsi Kasus Permasalahan

Dalam beberapa kasus video game, terdapat banyak permasalahan diantaranya sistem navigasi pencarian jalur tercepat di video game, namun untuk pencarian jalur tercepat sendiri terdapat 2 metode pelacakan, yaitu pelacakan dengan metode BFS dan DFS. Nah disini kita akan mencari tau manakah algoritma yang efisien dalam melakukan pelacakan, apakah dengan BFS (Iteratif) ataupun DFS (Rekursif).

B. Deskripsi Algoritma

BFS (Iteratif):

- BFS menggunakan **queue** untuk menjelajahi semua tetangga pada tingkat yang sama sebelum melanjutkan ke tingkat berikutnya.
- Hal ini membuat BFS menemukan **jalur terpendek** lebih cepat karena setiap level diproses sepenuhnya sebelum menjelajahi level berikutnya.

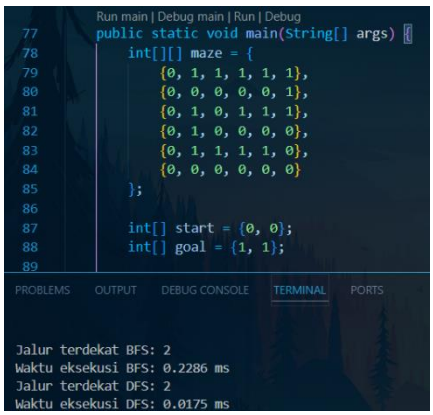
DFS (Rekursif):

- DFS menggunakan pendekatan **depth-first**, yaitu menjelajahi satu jalur hingga kedalaman maksimum sebelum mencoba jalur lain.
- DFS bisa lebih lama karena tidak secara langsung fokus pada jalur terpendek. Ia mungkin menjelajahi jalur yang tidak optimal sebelum menemukan solusi.

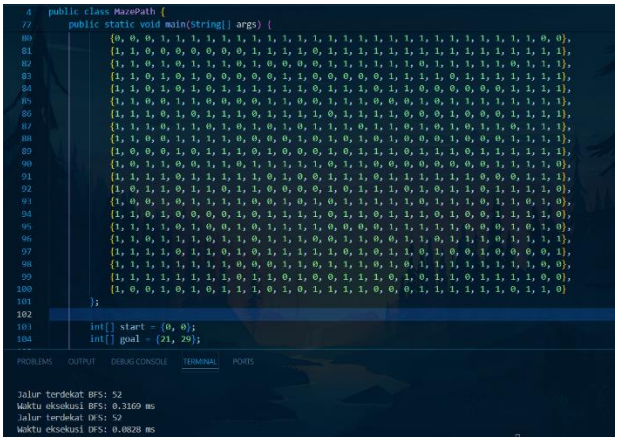
C. Perbandingan Runtime

Catatan : '0' adalah jalur yang bisa dilalui dan '1' adalah tembok.

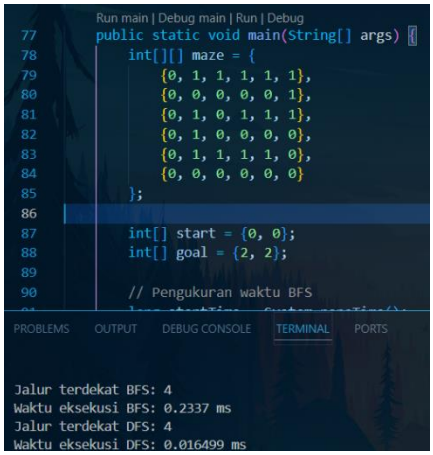
labirin 2x2 (goal pada koordinat [1,1])



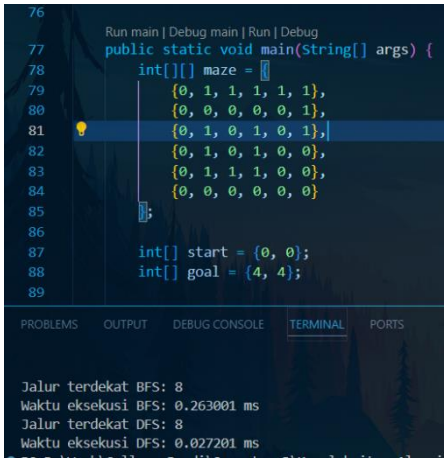
labirin 30x20 (goal pada koordinat [29,21])



labirin 3x3 (goal pada koordinat [2,2])

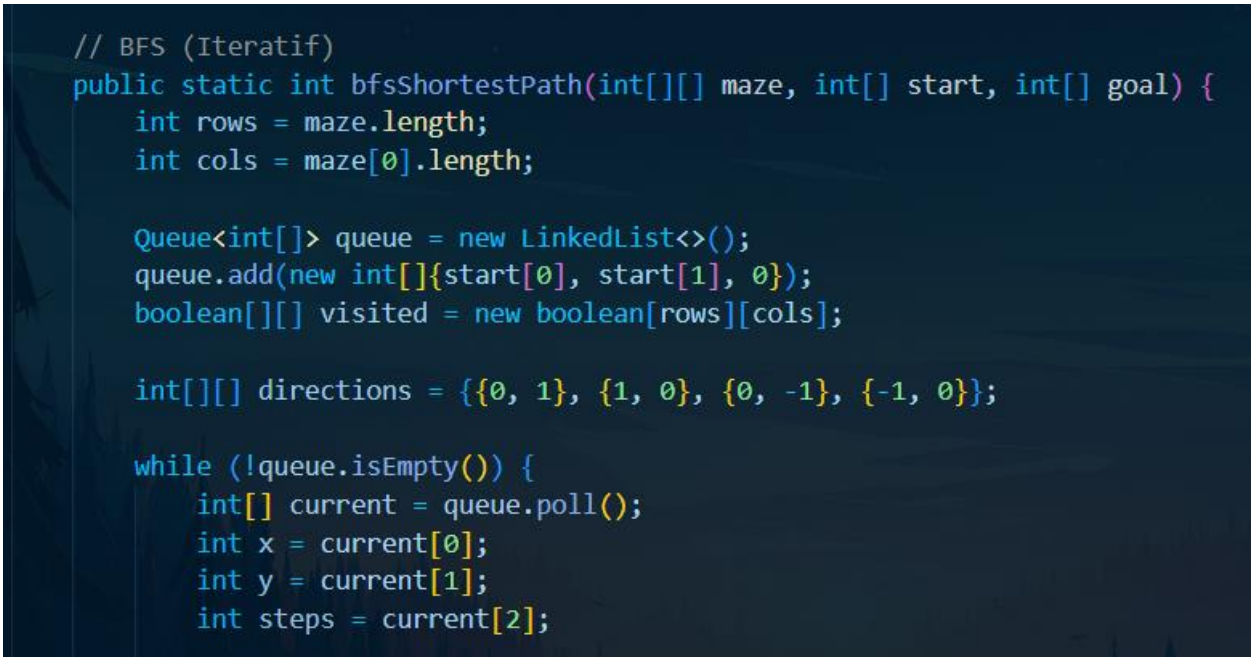


labirin 5x5 (goal pada koordinat [4,4])



D. Analisis perbandingan BFS dan DFS

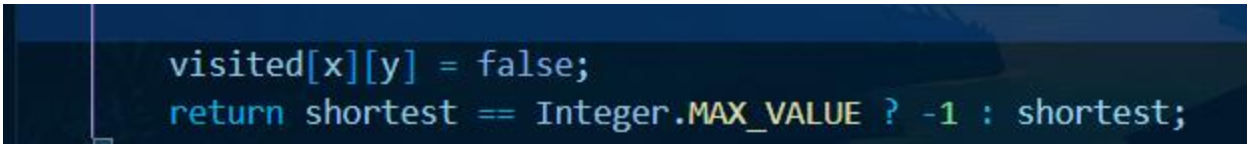
Seperti yang kita lihat, hasil runtime selalu unggul DFS jauh lebih cepat, dikarenakan pada algoritma BFS harus mencoba semua jalur terlebih dahulu baru menentukan hasilnya. Beda dengan BFS yang mencoba satu jalur hingga akhir, jika masih tidak sampai tujuan dan buntu DFS akan Kembali ke titik dimana jalur belum dilalui. Dan factor kedua penyebab BFS lebih lama adalah BFS membutuhkan ruang tambahan untuk antrean, sehingga memakan lebih banyak memori dibanding DFS.



```java
// BFS (Iteratif)
public static int bfsShortestPath(int[][] maze, int[] start, int[] goal) {
    int rows = maze.length;
    int cols = maze[0].length;

    Queue<int[]> queue = new LinkedList<>();
    queue.add(new int[]{start[0], start[1], 0});
    boolean[][] visited = new boolean[rows][cols];

    int[][] directions = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

    while (!queue.isEmpty()) {
        int[] current = queue.poll();
        int x = current[0];
        int y = current[1];
        int steps = current[2];
```

Untuk DFS sendiri kecepatan pada 2x2 dan 3x3 lebih cepat 3x3 dikarenakan DFS terdapat kesalahan pada pemilihan jalur sehingga melakukan backtracking.

Backtrack :



```java
    visited[x][y] = false;
    return shortest == Integer.MAX_VALUE ? -1 : shortest;
```

E. Referensi

https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=581161
https://lms.telkomuniversity.ac.id/course/view.php?id=8484