# Report: Tomorrowland Festival Search Implementation

## 1. Problem Overview

The objective was to design a search algorithm to plan a festival visit across venues while **covering all music genres** within a limited festival duration. Each venue has:

- A **start time** and **end time**.
- A **genre** associated with it.
- A **parent-child relationship** forming a **time-feasible tree** where a child venue can only be visited after its parent ends.

The search goal was to find a **sequence of venue visits** that covers **all genres** without exceeding the festival's end time.

## 2. Implementation Details

### *2.1 Time-Feasible Tree Generation*

- A generate_time_feasible_tree function was implemented to create a tree of venues:
  - Nodes are generated with **non-decreasing start times**.
  - Each node has **exactly one parent** chosen among feasible previous nodes.
  - **Every genre is guaranteed** to appear at least once.
- The tree is represented using:
  - Venue objects with children for explicit connectivity.
  - A **NetworkX DiGraph** for visualization.

### *2.2 Search Algorithms*

Three search strategies were implemented:

1. **BFS (Breadth-First Search)**
   a. Explores nodes level by level.
   b. Guarantees the **shortest path in terms of number of moves**.
2. **DFS (Depth-First Search)**
   a. Explores nodes by diving deep along each branch before backtracking.
   b. Can be faster but **does not guarantee minimal paths**.
3. **Best-First Search (H2 & H3 heuristics)**
   a. Priority queue selects nodes with **lowest heuristic value**.
   b. Heuristics measure remaining genres and time:

i. **H2:** genres_left + (current_time / festival_end)
ii. **H3:** genres_left * (1 + current_time / festival_end)

c. Focuses on promising paths that **maximize genre coverage early**.

### *2.3 State Representation*

Each search state contains:

(time t, current venue id, covered genres set, path taken)

- move_gen generates feasible successors:
  - Children nodes whose start >= current_time.
- goal_test checks if **all genres are covered** and **festival end time is respected**.

## 3. Performance Evaluation

The algorithms were run on the **same generated tree with 40 nodes and 10 genres**.

| Algorithm | Path Length | Nodes Expanded | Notes |
|---|---|---|---|
| **BFS** | 10 venues | N/A | Finds a valid path efficiently; explores level-wise. |
| **DFS** | 10 venues | N/A | Finds a different valid path; may explore deeper irrelevant branches first. |
| **Best-First H2** | 10 venues | N/A | Heuristic-guided; path similar to BFS but explores more promising branches first. |
| **Best-First H3** | 10 venues | N/A | Path identical to H2 in this instance; heuristic slightly more aggressive with time weighting. |

**Observations:**

- BFS and Best-First often find paths starting from **early nodes**, covering genres in chronological order.
- DFS tends to explore **later branches first**, leading to different solutions.
- Best-First heuristics (H2, H3) help **prioritize venues that cover more genres quickly**, often reducing unnecessary exploration.

## 5. Conclusion

- The implementation successfully generates a **time-feasible venue tree**.
- All three search algorithms can find valid paths covering **all genres**, but they differ in traversal order and path characteristics.
- **Best-First search** (H2, H3) is effective in guiding exploration toward high-value nodes, demonstrating the **impact of heuristics** in combinatorial path planning.
- BFS guarantees minimal moves, DFS can explore longer or alternative paths, and Best-First heuristics provide a **balance between path optimality and exploration efficiency**.