Exactly! Your summary is spot-on:

- **BFS (Breadth-First Search)**:
  Explores layer by layer (level by level) outwards from the start. This guarantees that the first time you reach the goal, you've found the *shortest* possible path in terms of number of steps. So BFS is ideal when you want the shortest path in a maze.

- **DFS (Depth-First Search)**:
  Explores deeply along one path before backtracking and trying others. It's usually simpler to implement (especially recursively). It will find *a* path if one exists, but it may not be the shortest. DFS might go down a very long winding path first before finding the goal.

Use BFS when:

- You want the shortest path.
- You don’t mind extra memory usage (since BFS can store many nodes at a level).

Use DFS when:

- You just want *any* path quickly.
- The maze is very large and you want a more memory-efficient approach (DFS generally uses less memory).
- You want to explore or solve recursively.

COMPARISON

Aspect	BFS (Breadth-First Search)	DFS (Depth-First Search)
Goal	Finds the **shortest path** from start to end.	Finds **any path** from start to end (not guaranteed shortest).
How it explores	Explores all neighbors at current depth before going deeper.	Explores as far as possible along one path before backtracking.
Data structure	Uses a **queue** (FIFO).	Uses a **stack** (LIFO), often implemented recursively.
Memory usage	Higher — stores all nodes at the current layer.	Generally lower — stores nodes along a single path.
Path length guarantee	Always finds shortest path (minimum steps).	May find a longer or suboptimal path.
Performance	Can be slower and consume more memory on large or complex mazes.	Can be faster initially but may explore longer paths unnecessarily.
Implementation complexity	Iterative, requires explicit queue.	Simpler to implement recursively.
Use cases	When shortest path is needed (e.g., GPS navigation, shortest route).	When any path suffices, or you want quick approximate solutions.
Behavior in mazes with loops	Uses visited set to avoid infinite loops and repeated states.	Same, but can get stuck exploring long branches if no goal nearby.