**🔍 1. Which algorithm gives the shortest path? Why?**

**✅ BFS (Breadth-First Search) gives the shortest path in terms of number of steps.**

**Why?**

* BFS explores all nodes **level-by-level** (i.e., by increasing depth).
* This guarantees that the **first time** it reaches a goal state, it’s the **minimum number of operations** needed to get there.
* BFS is **uninformed** (no heuristic), but still optimal in terms of path length for **uniform cost** problems like this one.

**❌ DFS (Depth-First Search)**

* DFS may find a **valid path**, but it’s **not guaranteed to be the shortest**.
* DFS explores deeply before considering other options, which may lead to longer paths or loops (without care).

**✅ A\* (A-Star Search)**

* A\* uses a **heuristic** (abs(x - GOAL) or abs(y - GOAL)) to guide the search.
* In your case, the heuristic is **very simple**, so A\* behaves similar to BFS but may perform fewer steps depending on heuristic quality.
* If the heuristic is **admissible** (never overestimates), A\* also guarantees the **shortest path**.
* With better heuristics, A\* can be **more efficient** than BFS while still finding optimal paths.

## 🔋 2. Which algorithm is more efficient in terms of ****time and memory****?

Let’s compare:

| **Algorithm** | **Time Efficiency** | **Memory Efficiency** | **Notes** |
| --- | --- | --- | --- |
| **BFS** | ❌ Slower for large spaces | ❌ High memory usage | Stores all paths at a level |
| **DFS** | ✅ Faster (can reach goal early) | ✅ Low memory usage | But may explore longer/irrelevant paths |
| **A\*** | ✅ Usually faster with good heuristics | ⚠️ Moderate memory | Trades space for speed |

### Summary:

| **Metric** | **Best Algorithm** |
| --- | --- |
| **Shortest Path** | ✅ **BFS** (or A\* with a good heuristic) |
| **Time Efficient** | ✅ **A\*** (usually), ✅ DFS (sometimes) |
| **Memory Efficient** | ✅ **DFS** |
| **Best All-Around (smart)** | ✅ **A\*** |

**✅ Which algorithm is more efficient in terms of time and memory?**

| **Algorithm** | **⏱ Time Efficiency** | **💾 Memory Efficiency** | **📝 Summary** |
| --- | --- | --- | --- |
| **BFS** (Breadth-First Search) | ❌ **Slow** (can explore many nodes before goal) | ❌ **High memory usage** (stores all frontier nodes) | Guarantees shortest path, but expensive |
| **DFS** (Depth-First Search) | ✅ **Faster** (may find goal quickly) | ✅ **Low memory usage** (only stores current path) | Memory efficient, but path is not guaranteed shortest |
| **A\*** (A-Star Search) | ✅ **Faster** than BFS (with good heuristic) | ⚠️ **Medium-High memory** (uses priority queue and stores paths) | Smart and fast if heuristic is good; otherwise can behave like BFS |

**🧠 Explanation:**

**🔹 Time Efficiency**

* **DFS** can be fast if it gets lucky and finds a goal early.
* **A\*** is typically **faster than BFS** because it uses a **heuristic to focus the search**.
* **BFS** is slow because it explores all possible states at each depth level before moving on.

**🔹 Memory Efficiency**

* **DFS** is best: it uses only a **single path** in memory (plus visited set).
* **BFS** is worst: it stores **all possible paths at each level**, which can grow exponentially.
* **A\*** falls in between: it uses a **priority queue (heap)** and stores more info (costs, heuristics, paths).