## Asymptotic Notation Classification Activity

**Instructions:**

1. For each pair of functions f(n) and g(n), determine whether f(n) falls under **Big-O (O)**, **Big-Omega (Ω)**, or **Big-Theta (Θ)** in relation to g(n).
2. Mark **Yes** if the relationship holds, otherwise mark **No**.
3. Discuss answers in teams and submit your final responses.

| f(n) | g(n) | O(g(n))? | Ω(g(n))? | Θ(g(n))? | Reasoning |
|---|---|---|---|---|---|
| | | | | | |
| $5n^2 + 3n + 7$ | $n^2$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | |
| | | | | | |
| $10n + \log(n)$ | $n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | |
| | | | | | |
| | | | | | |
| $3n^2 + 2n$ | $n^3$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | |
| | | | | | |
| $5n^2 + 3n + 7$ | $n^2$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $10n + \log(n)$ | $n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $3n^2 + 2n$ | $n^3$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $4n^3 + 5n$ | $n^3$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $n!$ | $2^n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $2^n + n^3$ | $2^n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| $n \log n + 5n$ | $n \log n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |
| 100 | 1 | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No | _____ |

| **f(n)** | **g(n)** | **O(g(n))?** | **Ω(g(n))?** | **Θ(g(n))?** |
|---|---|---|---|---|
| $5n^2 + 3n + 7$ | $n^2$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $10n + \log(n)$ | $n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $3n^2 + 2n$ | $n^3$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $4n^3 + 5n$ | $n^3$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $n!$ | $2^n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $2^n + n^3$ | $2^n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| $n \log n + 5n$ | $n \log n$ | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |
| 100 | 1 | [ ] Yes / No | [ ] Yes / No | [ ] Yes / No |

$\text{og}(n^2)$ $\log(n)$ ✅ ✅ ✅ $\log(n^2) = 2 \cdot \log(n)$

| f(n) | g(n) | O(g(n))? | Ω(g(n))? | Θ(g(n))? | |
|---|---|---|---|---|---|
| $n \log n$ | $n^2$ | ❌ | ✅ | ❌ | $n^2$ grows faster |
| $\sqrt{n}$ | $\log n$ | ❌ | ✅ | ❌ | sqrt(n) > log n |
| $n^3$ | $2^n$ | ✅ | ❌ | ❌ | exponential grows faster |
| $\frac{n}{\log n}$ | $n$ | ✅ | ❌ | ❌ | log n in denominator slows growth |
| $\log(n!)$ | $n \log n$ | ✅ | ✅ | ✅ | Stirling approximation |
| $2^n + 3^n$ | $3^n$ | ✅ | ✅ | ✅ | $3^n$ dominates |
| $n!$ | $n^n$ | ✅ | ❌ | ❌ | $n^n$ grows faster than $n!$ |

**Bonus Challenge:**

- Provide your reasoning for at least two of your answers.
- Create your own function pair and challenge another team to classify it correctly.

**Learning Objective:**

- Understand and apply asymptotic notation concepts (Big-O, Omega, Theta).
- Develop problem-solving skills in analyzing algorithm efficiency.
- Work collaboratively to enhance learning through discussion and competition.

# Puzzle Solving: 15-Puzzle Using Backtracking

**Objective:**

Students will manually solve the **15-puzzle** using **backtracking strategies**, helping them understand how **state-space search and constraint satisfaction** work in problem-solving.

---

**What is the 15-Puzzle?**

The **15-puzzle** is a **4×4 sliding puzzle** with **15 numbered tiles (1-15) and one empty space**. The goal is to arrange the numbers in ascending order from **1 to 15**, using the empty space to slide tiles.

**Example of Initial State:**

CopyEdit

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 _

Where **"_"** represents the empty space.

**Example of a Scrambled State:**

CopyEdit

1 2 3 4

5 6 _ 8

9 10 7 12

13 14 11 15

---

**How to Solve Using Backtracking?**

1. **Define the Possible Moves:**
    - The empty space can move **UP, DOWN, LEFT, or RIGHT**, but only within the 4×4 grid.
2. **Check for a Valid Move:**
    - A tile can only move into the empty space **if it's adjacent** (not outside the grid).
3. **Recursive Backtracking Approach:**

- Move a tile into the empty space.
- If this move gets closer to the goal, continue.
- If not, **backtrack** (undo the last move and try another path).
- Stop when the numbers are arranged correctly.

4. **Base Condition:**
   - If the puzzle is solved (goal state reached), **stop the recursion**.

---

**Student Activity: Solving Manually**

**Step 1: Provide a Scrambled 15-Puzzle on Paper**

- Give students different scrambled puzzle boards.

**Step 2: Ask Them to Solve it Step-by-Step Using Backtracking**

- Encourage them to record each move (up, down, left, right).
- If they reach a dead-end (no valid move left), they must **backtrack** and try another path.

**Step 3: Discussion**

- Have students compare the number of moves they took.
- Discuss optimal vs. non-optimal paths.
- Explain how computers can use **DFS-based backtracking** to solve this puzzle efficiently.

---

**Key Takeaways:**

- The **15-puzzle** is an example of a **constraint satisfaction problem**.
- **Backtracking helps** find a solution by exploring all possibilities and undoing incorrect moves.
- **State-space search** is used to track explored and unexplored moves.
- This approach is similar to solving **N-Queens, Graph Coloring, and Hamiltonian Cycle problems**.

Would you like a **Python implementation** of the 15-puzzle using backtracking? 😊

**Rubrics for 15-Puzzle Solving Using Backtracking (10 Marks)**

| Criteria | Excellent (10-9 Marks) | Good (8-7 Marks) | Satisfactory (6-5 Marks) | Needs Improvement (4-0 Marks) |
|---|---|---|---|---|
| **Understanding of Problem Statement (2 Marks)** | Clearly understands the puzzle rules and goal. Can explain backtracking approach accurately. | Understands the puzzle and backtracking but with minor gaps in explanation. | Basic understanding of the puzzle but struggles with backtracking concepts. | Unable to explain the puzzle rules or backtracking. |
| **Application of Backtracking (3 Marks)** | Applies backtracking efficiently, correctly identifies valid and invalid moves, and backtracks appropriately. | Applies backtracking correctly but may take longer paths or make unnecessary moves. | Uses trial-and-error approach with limited understanding of backtracking. | Moves tiles randomly without applying backtracking principles. |
| **Solution Accuracy (3 Marks)** | Reaches the correct solution with an optimal number of moves. Documents the path clearly. | Reaches the solution but with some unnecessary moves. Path is mostly clear. | Solves the puzzle with excessive moves or partial correctness. | Unable to solve the puzzle or reaches an incorrect solution. |
| **Clarity of Steps and Justification (2 Marks)** | Clearly writes down each step taken and justifies why a move was made. | Writes steps but lacks justification for some moves. | Steps are unclear or missing key parts of the process. | No written steps or explanations provided. |

**Total: 10 Marks**

This rubric ensures fair evaluation based on **concept understanding, problem-solving skills, accuracy, and explanation clarity**.

Would you like modifications or a sample grading scenario? 😊