Valid Parantheses

Solving the valid parantheses problem → a classic use case for a stack.

Core Idea: Use a stack to track open brackets. For each character in the string:

1. If it's an opening bracket - push it onto the stack
2. If it's a closing bracket:
   - If the stack is empty → ✗ invalid
   - Pop the top and check if it matches the expected open bracket → if not, ✗ invalid
3. At the end, the stack should be empty for a valid string.

Matching Pairs: Use a hash map for quick lookups:

```
unordered_map<char, char> match = {{')', '('}, {']', '['},
                                   {'}', '{'}};
```

Edge Cases:
- ")(" → ✗ invalid (wrong order)
- "(((" → ✗ invalid (never closed)
- "([]{})" → ✓ valid (correct nesting)

Time & Space Complexity:
- Time: O(n) - scan each character once
- Space: O(n) - worst-case stack usage