Tournament Results Sorting System (30 Points)

Problem Description

You are developing a system for an international gaming tournament. Each player has the following information:

- Player ID (string)
- Username (string)
- Score (integer)
- Win Rate (float, 0-100)
- Country Code (string, 2 characters)

Task Requirements

Create a program that implements the following requirements:

1. Create a sorting system that can:
   - Sort players by score (primary)
   - If scores are equal, sort by win rate (secondary)
   - If both are equal, sort by username alphabetically (tertiary)

2. The program must:
   - NOT use any Python built-in sorting functions
   - Implement at least 2 different sorting algorithms of your choice
   - Allow users to choose which sorting algorithm to use
   - Display the sorting process step by step

3. Additional Features:
   - Group players by country after sorting
   - Calculate average score per country
   - Find the median score per country

Input Format

```
players = [
  {"id": "P123", "username": "DragonMaster", "score": 2800, "win_rate": 75.5, "country": "US"},
  {"id": "P456", "username": "NinjaCoder", "score": 2800, "win_rate": 75.5, "country": "JP"},
  {"id": "P789", "username": "CyberKnight", "score": 2800, "win_rate": 80.0, "country": "US"},
  {"id": "P234", "username": "PixelWarrior", "score": 2750, "win_rate": 82.5, "country": "KR"},
  {"id": "P567", "username": "BinaryBeast", "score": 2900, "win_rate": 70.0, "country": "JP"}
]
```

Required Output

1. Sorted player list showing:

```
Sorting Process:
Step 1: [Current state of array]
Step 2: [Current state of array]
...

Final Sorted Result:
1. BinaryBeast (JP) - 2900 pts, 70.0%
2. CyberKnight (US) - 2800 pts, 80.0%
3. DragonMaster (US) - 2800 pts, 75.5%
4. NinjaCoder (JP) - 2800 pts, 75.5%
5. PixelWarrior (KR) - 2750 pts, 82.5%
```

2. Country Statistics:

```
Country Analysis:
JP: Average Score: 2850, Median Score: 2850
US: Average Score: 2800, Median Score: 2800
KR: Average Score: 2750, Median Score: 2750
```

Implementation Requirements

1. Create a Player class to store player information
2. Implement custom comparison functions
3. Implement at least 2 sorting algorithms
4. Create functions for statistical calculations
5. Include step-by-step visualization
6. Handle all edge cases

Sample Test Cases

```
test_cases = [
    # Test Case 1: Same scores, different win rates
    [
        {"id": "P1", "username": "Alpha", "score": 1000, "win_rate": 60.0, "country": "US"},
        {"id": "P2", "username": "Beta", "score": 1000, "win_rate": 70.0, "country": "US"}
    ],

    # Test Case 2: Same scores and win rates
    [
        {"id": "P3", "username": "Charlie", "score": 1500, "win_rate": 80.0, "country": "JP"},
        {"id": "P4", "username": "Alpha", "score": 1500, "win_rate": 80.0, "country": "KR"}
    ],

    # Test Case 3: Mixed cases
    [
        {"id": "P5", "username": "Delta", "score": 1200, "win_rate": 75.0, "country": "US"},
        {"id": "P6", "username": "Echo", "score": 1200, "win_rate": 75.0, "country": "JP"},
        {"id": "P7", "username": "Alpha", "score": 1200, "win_rate": 75.0, "country": "KR"}
    ]
]
```