Question: Could you design a new approach to improve the average-case?

Yes. By checking the most common scenario first (i.e., reordering the `if` conditions so that the "most likely" condition is evaluated first), the average number of comparisons/branches decreases.

If we know that most inputs are greater than 1 (e.g., uniform distribution on {0,1,2,3,4,5}, so 4 out of 6 inputs are indeed >1), we can rearrange the condition checks so that the frequent case is tested first.

```
int Classify(int a) {
    // Check a>1 first:
    if (a > 1)
        return 3;
    else if (a == 0)
        return 1;
    else
        return 2; // covers the case a == 1
}
```

Operation counting:

- **Case a > 1**:
  1. Compare `(a > 1)` → true
  2. branch
  3. Return
  4. That's only 3 operations total.

- **Case a == 0**:
  1. Compare `(a > 1)` → false
  2. Branch
  3. Compare `(a == 0)` → true
  4. Branch
  5. Return
  6. About 5 operations.

- **Case a == 1**:
  1. Compare `(a > 1)` → false
  2. Branch
  3. Compare `(a == 0)` → false
  4. Branch
  5. Return (the "else" case)
  6. Also 5 operations.

New Average (Uniform over {0,1,2,3,4,5})

- a = 2, 3, 4, 5: 4 inputs, each costs 3 operations
- a = 0, 1: 2 inputs, each costs 5 operations

New Average = (4 × 3 + 2 × 5) / 6 = 22/6

This is significantly better than the original one 28/6, thereby improving the average cost.

Question: What the average case would be if the input is through 1 to 3?

For the original one:

```
int Classify(int a) {
if (a == 0) return 1;
else if (a == 1) return 2;
else return 3;
}
```

For a = 1: The code does two comparisons, two branches, and one return = 5 operations.

For a = 2 or a = 3: Same path after failing both comparisons, so also 5 operations each.

Average = (5+ 5 + 5) / 3 = 5

For the new one:

```
int Classify(int a) {
if (a > 1)
return 3;
else if (a == 0)
return 1;
else
return 2;
}
```

For a = 1: The code does two comparisons, two branches, and one return = 5 operations.