Asymptotic Analysis - Example 2: Elements Uniqueness

```
Function being analyzed:

def uniqueElements(A):
    for i in range(0, len(A) - 1):
        for j in range(i + 1, len(A)):
            if A[i] == A[j]:
                return False
    return True
```

This analysis explains the cost T(n) of the 'uniqueElements' function, which checks whether all elements in an array A of size n are unique.

The function uses two nested loops and in the worst-case scenario (no duplicates), all comparisons are made.

1. T(n) = (n - 1) * c1 + ((n - 1)n)/2 * c2 + ((n - 1)n)/2 * c3 + 0 * c4 + c5
   - Line 5 runs (n - 1) times
   - Lines 6 and 7 (inner loop and comparison) run (n(n - 1))/2 times
   - Line 8 not run in worst case; Line 9 runs once
2. T(n) <= n * c1 + n^2 * (c2 + c3) + c5
   - Approximated (n - 1) as n, and dropped constant 1/2 for Big-O
3. T(n) <= n * c1 + n^2 * c6 + c5
   - c6 = c2 + c3 to simplify expression
4. T(n) <= n^2 * c1 + n^2 * c6 + n^2 * c5
   - Linear term replaced with quadratic upper bound
5. T(n) <= n^2 * (c1 + c6 + c5)
   - Factored out n^2
6. T(n) <= n^2 * c7

- All constants grouped into a single constant c7

Conclusion:

The total time complexity T(n) of the function is O(n^2), which means if the input size doubles, the effort increases by a factor of 4.

That is: T(2n) = 4 * T(n).