**Table Join Operation Project Report**

Kai Qi

In this project, I used four methods to find the join of two tables. The algorithm and the time complexity analysis are as followed:

**Method 1:** Sort both tables and then use a merge-like intersection, i.e. use a modified version of the method merge from the Merge sort algorithm to find the common values.

**O1(m, n) =**  **+ n + n = n**

**Method 2:** Sort only the larger table, and then use binary search on the resulting table for each value in the smaller table.

**O2(m, n) =** **n + m = n**

**Method 3:** Sort only the smaller table and then use binary search on the resulting table for each value in the larger table.

**O3(m, n)** = **m + n = n**

**Method 4:** Search the larger table for each value in the smaller table. None of the tables should be sorted. Both tables should be accessed sequentially.

**O4(m, n)** = **mn**

Based on the above time complexity analysis, some hypotheses can be concluded as followed:

1. Method 4 basically has the worst performance.
2. Because m is less than n, method 3 performs better than method 2.
3. If m is very close to n, method 1, method 2 and method 3 will have a very similar performance.
4. If n is much larger than m, method 3 will perform better than the other methods.
5. K has no impact on the performances of the four methods.
6. O3 O2 O1 O4

Now, let’s use some cases to test the above hypotheses:

**A screenshot of a cell phone

Description automatically generated**

In the above table, I use 6 cases to test the time that is consumed when finding the intersection.

1. In case 1-3, m is close to n. The consumed time of method 1, method 2 and method 3 is very close. Only method 4 performs much worse.
2. In case 4-6, n is much larger than m. The performance of method 3 is significantly better than the rest of the other methods.
3. K has no impact on the order of the performance of the four methods. That is to say, the expected number of common values does not impact the order of the performance of the four methods.

To sum up, if m is close to n, we can choose anyone of the first three methods, but I recommend method 3. If n is much larger than m, we’d better choose method 3 since it performs best.