Algorithm 1 – Optimal Solution

In main1.cpp, I decided to code for both DFSBnB and IDA* search, and experiment between the two. For the test cases I ran, I observed that IDA* was giving quicker results, though DFSBnB was creating lesser nodes I also wrote A* search, but decided that the memory it was using for larger test cases was too large and I may run into errors due to memory. Hence I went with IDA* search. I tried a few heuristic, with some depending on the frequency of each character in the vocabulary, and some using pairwise dynamic programming solutions to estimate the cost. This turned out to be a weak heuristic, and efforts to improve the heuristic led to it becoming much faster, but also inadmissible for certain cases. Hence it is not used. For the DP heuristic, I kept CC=0 while computing cost to prevent double counting of these cost, as the cost is independent of the K(number of strings). To improve the heuristic, I calculated the minimum number of dashes that would be needed to satisfy the constraint of equal length strings, and then added the cost of inserting this number of dashes. This gave reasonable results and is implemented in our solution.

Algorithm 2 –Sub-Optimal Solution

For the sub-optimal solution, I decided to go with hill climbing with random restart. The initial solution is generated with the aid pseudo random number generators available in C++. I decided to go with a small sized neighborhood function, so the number of neighbors is reasonable even for larger sized problems. I also obtained an initial solution using an algorithm similar to the **ClustalW algorithm**. I observed that it gave reasonable results for smaller K and L, but was not promising for larger cases.