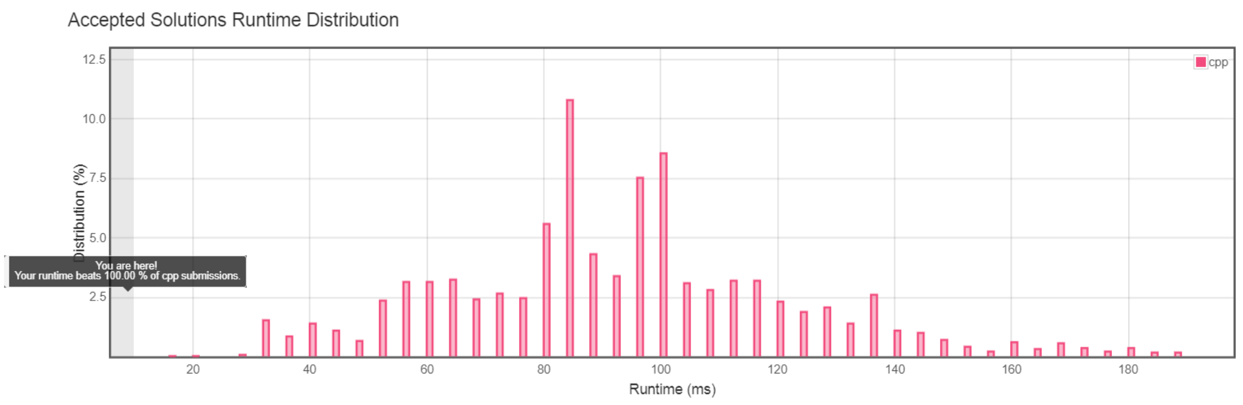
1. DP. $O(n^2)$.
2. We can reduce this to LCS, by computing the LCS between s and its reverse.

LCS algorithms:
1. $O(\frac{nm}{\log^2 n})$ by method of four russians [3], for constant alphabet size. divide the $n \times n$ DP matrix into blocks of size $t \times t$. The LCS DP matrix has a property that neighboring cells can have DP values differ by at most 1, so there are only $2^{\Theta(t)}|\Sigma|^{\Theta(t)}$ possible transition matrices. Set $t = O(\log n)$ is sufficient.
note. The paper originally showed $O(\frac{n^2}{\log n})$, but can actually achieve $O(\frac{n^2}{\log^2 n})$ by compactly representing the DP values, using bit packing. As a result, some references for this paper are inaccurate (e.g. wiki: Longest common subsequence problem).
2. for unbounded alphabet size: $O(\frac{nm \log\log n}{\log^2 n})$, or $O(\frac{nm}{\log^2 n} + r)$, where r is the number of matches in the dynamic programming matrix [2].
3. $O(\frac{nm}{w})$ by bit packing [1].

The following implementation is $O(\frac{n^2}{w})$.



note. if we want to output the subsequence, one pass of LCS can only get half of the solution.

References

[1] Maxime Crochemore, Costas S Iliopoulos, Yoan J Pinzon, and James F Reid. A fast and practical bit-vector algorithm for the longest common subsequence problem. *Information Processing Letters*, 80(6):279–285, 2001.

[2] Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. *Discrete Applied Mathematics*, 212:96–103, 2016.

[3] William J Masek and Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.