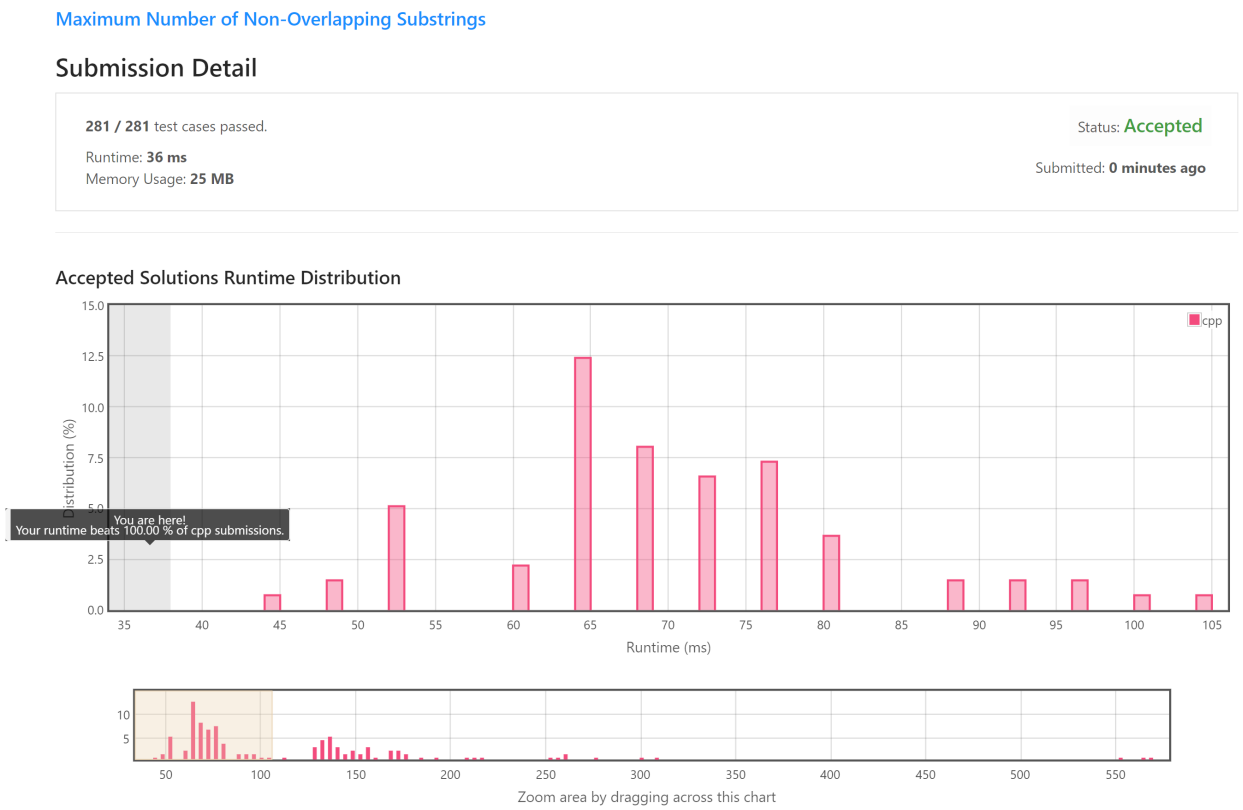
1. add characters from left to right. fix the right endpoint of an interval, we can greedily choose the rightmost valid left endpoint. we can use a stack to maintain the valid left endpoints: when we add a new character c at index i, either it hasn't appeared before, and the best left endpoint is i, or it appeared before, and the current best interval must contain index $i-1$. but then the current best interval must contain the best interval at right endpoint $i-1$, and we can repeatedly apply this argument, and merge the top two intervals in the stack. now we can use DP to compute the optimal solution. let $f[i]$ denote the optimal solution for prefix i, then we either use the last interval in the stack (which contains index i), or not use index i. we can maintain whether the last interval in the stack is valid, in amortized $O(1)$ time. total running time $O(n)$.
2. it's easier to have running time $O(n+\text{poly}(|\Sigma|))$, because we only need to consider $|\Sigma|$ intervals produced by the greedy algorithm (for each character c, let the right endpoint be the rightmost position of c, and greedily select the left endpoint).
3. $O(n|\Sigma|)$ is much easier.



References