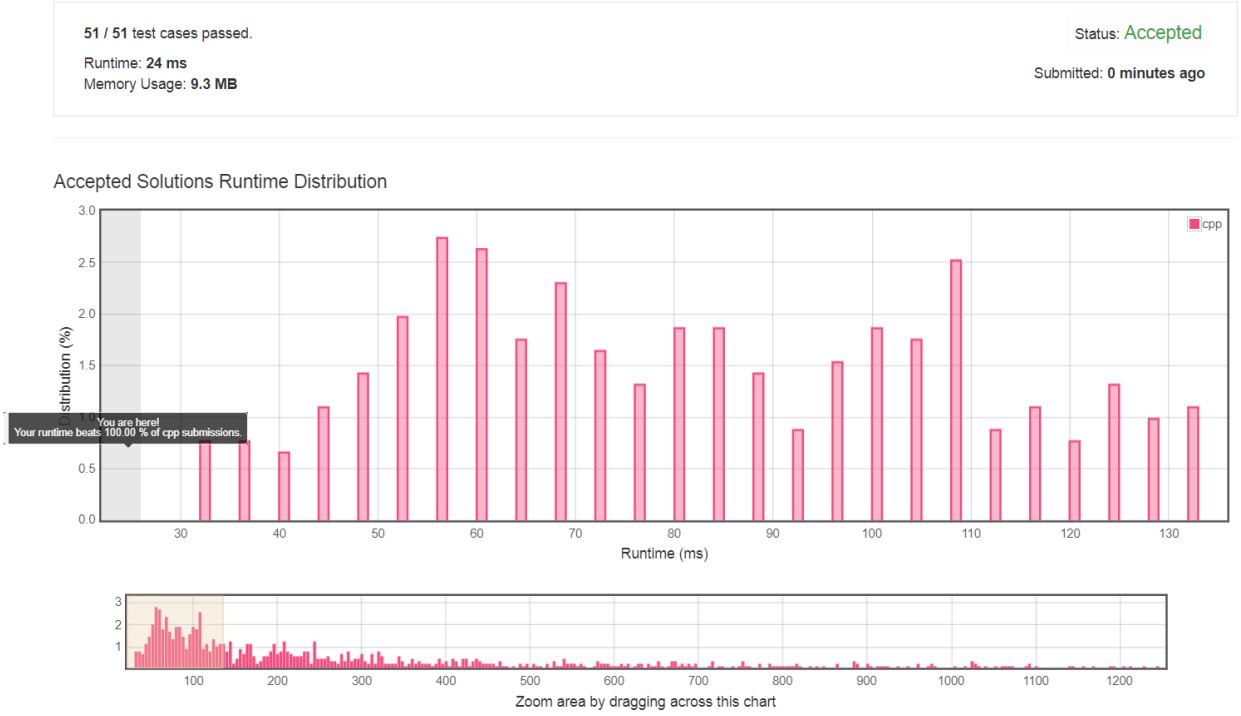
greedy, at each round, suppose we can swap k times, find the minimum number among the first $k+1$ numbers, and delete it (i.e. swap it to the beginning).
1. use segment tree to maintain. $O(n \log n)$.
remark. we can shave a log log factor.
2. use heap. initially put the first $k+1$ elements into the heap. at each round, find the minimum number, delete it, and use rank operation to compute the number of swaps. $O(\text{sort}(n) + n\frac{\log n}{\log w}) = O(n\frac{\log n}{\log w})$. [2]
3. after sorting, reduce to offline orthogonal range counting. $O(\text{sort}(n) + n\sqrt{\log n}) = O(n\sqrt{\log n})$. [1]
remark. on the other hand, for unbounded $|\Sigma|$, we can reduce the problem of counting inversions to this problem (we can sort the array iff $k \geq$ #inversions).

for small $|\Sigma|$, we can obtain better running time:
4. heap can be implemented in $O(|\Sigma|)$ time, and we can also use $|\Sigma|$ counters to support rank. $O(n|\Sigma|)$. we can further improve to $O(n \log |\Sigma|)$ by maintaining a segment tree on those $|\Sigma|$ counters.
5. when $k \geq n$, we can greedily swap the minimum element to the beginning of the array, and we can deal with all elements equal to a fixed value in amortized $O(\frac{n}{w})$ total time, using bit packing. after that we know $k < n$, we can use algorithm 2, but now we support the rank operation using brute force (i.e. doubly linked list), because the total number of swaps is $O(k) = O(n)$. also, heap can be implemented in $O(\lceil \frac{|\Sigma|}{w} \rceil)$ time (using bit packing). $O(n\lceil \frac{|\Sigma|}{w} \rceil)$, which is $O(n)$ for $|\Sigma| = 10$.



References

[1] Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 161–173. Society for Industrial and Applied Mathematics, 2010.

[2] Paul F Dietz. Optimal algorithms for list indexing and subset rank. In *Workshop on Algorithms and Data Structures*, pages 39–46. Springer, 1989.