CS2040S Recitation 3

AY 24/25 Sem 2 — github/omgeta

Q1. (a.) Use a MergeSort algorithm. Our merge step is only $O(n)$ because when we assume smaller subarrays are sorted, we only need at most n cost for reversal. Therefore, our final time complexity is $O(n \log n)$

(b.) Use the binary algorithm as a partition algorithm by considering each element as $<$ or $\geq$ the pivot. We have $T(n) = 2T(\frac{n}{2}) + O(n \log n) = O(n \log^2 n)$

(c.) Implement a 3-way partition using the binary partition twice. Once to split between $<$ and $\geq$. Second time to split between $=$ and $>$.

Q2. (a.) We want permutations which are random. That is, each of the $n!$ permutations must have probability exactly $\frac{1}{n!}$

(b.) Create an new array and for each index, choose a random element in the original. Time complexity $O(n)$. Space complexity $O(n)$.

(c.) No, it does not have a uniform distribution.

(d.) It maintains a prefix of i randomly sorted elements. This produces good permutations.

(e.) Probability of an element remaining in its place $= \frac{(n-1)!}{n!} = \frac{1}{n}$. Expected number of elements in its same position $= n \cdot \frac{1}{n} = 1$

(f.) Not without modifications. If we fail to get elements out of their own position, try again.

(g.) Better algorithm for this situation does not use truly random permutations to ensure expected number of students with their own assignment is 0.