Divide-and-Conquer & Greedy

Sunday, February 14, 2021 7:45 AM

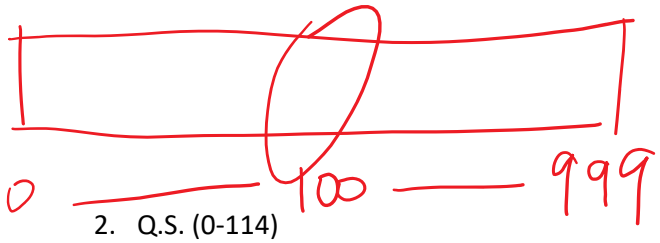
Selection sort:

Pivot element: k -> 0 to n-1
Quick sort:
Any element can be pivot

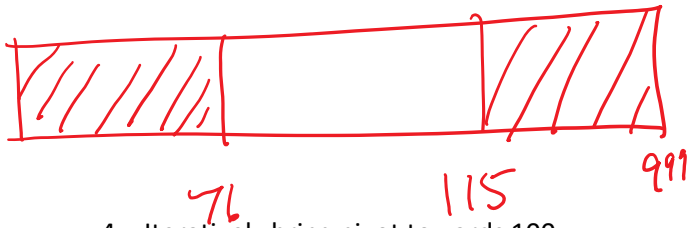
1. If 0 to n-1: O(n^2)
2. If random: O(nlogn)

Assume we want to find the position of 100 in an array of 999 elements

1. Q.S. random pivot 115



2. Q.S. (0-114)
3. Q.S. Random 75



4. Iteratively bring pivot towards 100

Amortized complexity: O(nlogn)

D&C

- Recursive
- Lot of stack space
- If the program stops in between, it must be rerun ()
- Same operation gets repeated multiple time
- All disadvantages of a recursive solution
- Cannot run for bigger solutions

Greedy Approach

- Using a greedy, it finds out the best solution at that stage. May not be the best solution at that stage.
- Simple solutions
- Optimal solution may not always optimal
- O(logN), O(n)

Dynamic

- More complex

- Gives optimal and feasible solution
- O(n^2)

Greedy	Dynamic
Fractional knapsack	0/1 Knapsack

n=3, m=20kg

Item	Price	Weight	p/w
1	25	18	1.3889
2	24	15	1.6
3	15	10	1.5

Maximization problem
- Maximize the profit such that sum of wi <= W

Dynamic programming, you may pick 1 or 0 items. In greedy you can divide the items.

	{item1}	{item2}	{item3}
Dynamic	Optimal	Feasible	Feasible

Greedy:
Maximize pixi such that wixi<= W
Decreasing order of profit/weight

Item2	1.6	15
Item3	1.5	10
Item1	1.38	18

Total available weight = 20kg
15kg of item2 + 5kg of item3
Profit = 24+ (5*1.5) = 31.5