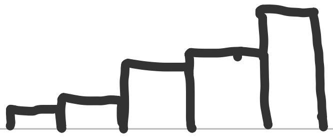
Sorting



① Patience Sorting :-

Dry run for [9, 10, 2, 3, 5, 1, 101, 18]

$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9, & 10, & 2, & 3, & 5, & 1, & 101, & 18 \end{array}$$

[9, 10, 2, 3, 5, 1, 101, 18]
i

vector < linked list>

i = 0 { 9

1 { 9, 10

2 { 2 ↓ 9 , 10

i = 3 { 2 ↓ 9 , 3 ↓ 10

i = 4 { 2 ↓ 9 , 3 ↓ 10 , 5

i = 5 { 1 ↓ 2 ↓ 9 , 3 ↓ 10 , 5

i = 6 { 1 ↓ 2 ↓ 9 , 3 ↓ 10 , 5, 101

i = 7 { 1 ↓ 2 ↓ 9 , 3 ↓ 10 , 5, 18 ↓ 101 }

↳ in partially sorted order

#length of the vector <Lin Lis> = LIS

#algorithm/longest common subsequence (LIS)

partitioning / Quick Sort / Quick Select

pivot (5)

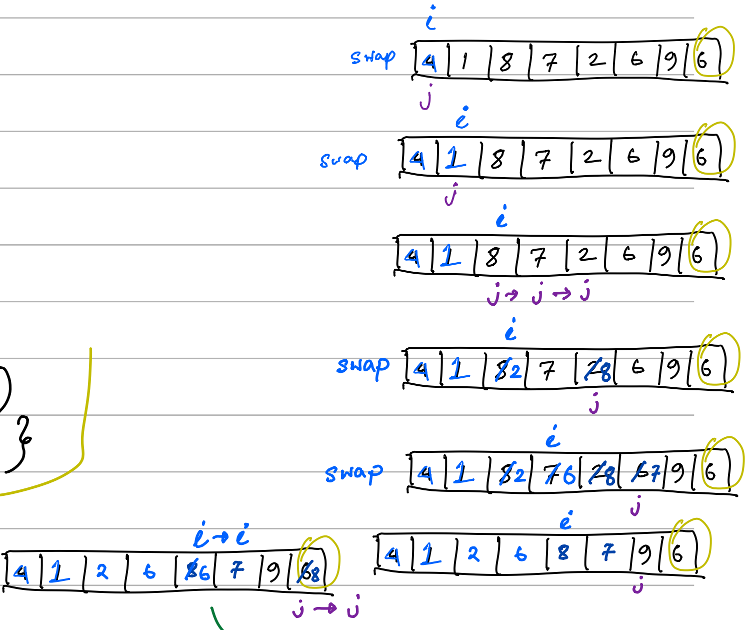
7	8	2	1	10	9	3	12	1

↑ ↑
i j

i to len : unknown
0 to j-1 : ≤ 5
j to i-1 : > 5

similar Questions :-
- ↳ partition odd - even.
- ↳ 0 - 1
- ↳ Zero Non zero partition.

```
> ind partition{arr, L,R}
  { int i = L, j = L;
    int P = arr[R]
  while ( j <= R)
  {   if(arr[j] <= P){
          swap(arr[i], arr[j])
          i++
      }
      j++;  }
  return i-1
  }
```



* return i-1; → index of the pivot.

```
> QuickSort(arr, L, R)
    if (L>=R) return

    int idx = partition(arr, L, R)
    QuickSort(arr, L, idx-1)
    QuickSort(arr, idx+1, R)
```

```
> int QuickSelect(arr, L, R, Target)
      int idx = partition(arr, L, R)
      int piv = arr[idx].
      if ( Target > piv )
            return QuickSelect(arr, idx+1, R, T)
      else if ( Target < piv)
            return QuickSelect(arr, L, idx-1, T)
      else   return idx
```

/or else if Target index given & find the arr[idx].

Counting Sort (stable)

i ←	i ←	i ←	i ←	i ←	i ←	i ←	i ←	i
9_a	3_a	6_a	5_a	3_b	9_b	6_b	5_b	9_c
0	1	2	3	4	5	6	7	8

start from 8.
for int i=n-1 to 0
pos= PS[arr[i]-min]
ans[pos] = arr[i]
PS[arr[i] - min] --;

ans:

3_a	3_b	5_a	5_b	6_a	6_b	9_a	9_b	9_c

	0_3	1_4	2_5	3_6	4_7	5_8	6_9
count	2	0	2	2	0	0	3
prefixSu	2	2	4	6	6	6	9
subtract 1	~~1~~ ~~0~~ -1	1	~~3~~ ~~2~~ 1	~~5~~ ~~4~~ 3	5	5	~~8~~ ~~7~~ ~~6~~ 5

Represents the last index available for that number

✓✓

Radix Sort :

↓		↓		↓		
28	→	321	→	321	→	28
123		123		123		69
321		28		028		123
69		69		069		321

uses stability of Count sort.

Wiggle Sort:

1, 2, 3, 4, 5, 6, 7, 8 . . .

idx $\quad 0 \quad 1 \quad 2 \; \dots$

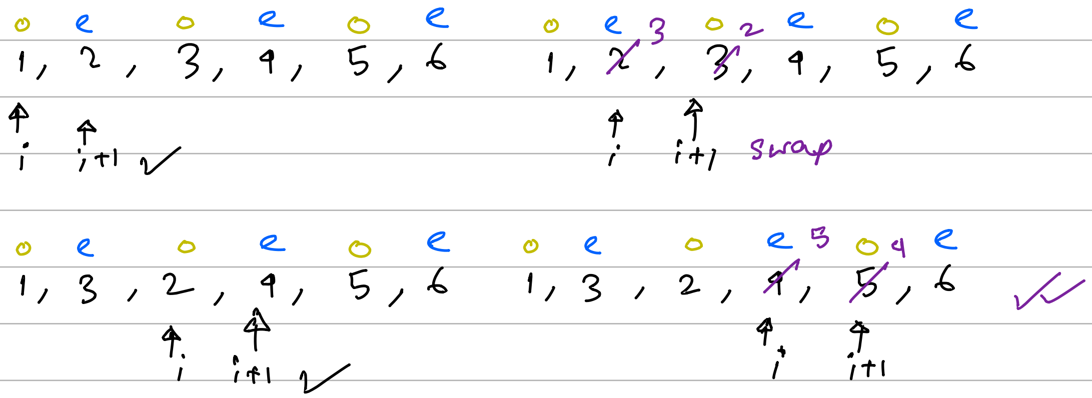
ws = a, b, c, ----

$\overset{0}{a} \le \overset{1}{b} \ge \overset{2}{c} \le \overset{3}{d} \dots$

odd idx → small Element

even idx → larger element.

✓ Just swap according to arr[idx]



Wiggle Sort II:

a < b > c < d $\quad$ strictly < or >.