- Recitation on Saturday - Same time as usual.
- Rajas Nanda will review Stable Matchings + asymptotic notation.
- Zoom link may change. Check Piazza.

Binary Search.

<u>Input</u>: . Array A of distinct integers.
. A is sorted
- int k

<u>Objective</u>: O/P True, if k is in A
false, o.w.

BSearch (A [lo .. hi], k)
<u>if</u> lo > hi <u>then</u>
return false
<u>else</u>
mid $\leftarrow \left\lfloor \frac{lo + hi}{2} \right\rfloor$

```
if A[mid] = k then
    return True
else if A[mid] < k then
    return BSearch(A[mid+1..hi], k)
else
    return BSearch(A[lo..mid-1], k)
```

Runtime recurrence:

$T(n)$: worst case running time of BSearch on an i/p of size n.

$$T(n) = \begin{cases} O(1), & n \leq 1 \\ T(n/2) + O(1), & \text{otherwise.} \end{cases}$$

(assume that n is an exact power of 2).

$T(n) = \Theta(\lg n)$.

InsertionSort (A[1..n])

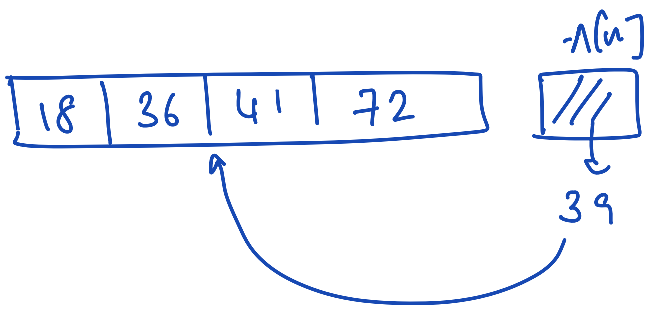
    if $n == 1$ then

        return

    else

        $A' \leftarrow$ InsertionSort ($A[1..n-1]$) $\longrightarrow T(n-1)$

        Insert (A', $A[n]$)



Runtime recurrence:

$$T(n) = \begin{cases} O(1), & n = 1 \\ T(n-1) + O(n), & \text{o.w.} \end{cases}$$

$$T(n) = T(n-1) + cn$$

$$= T(n-2) + c(n-1) + cn$$

$$= T(n-3) + c(n-2) + c(n-1) + cn$$

$$\vdots$$

$$= T(n-k) + c(n-(k-1)) + c(n-(k-2)) + \cdots + cn$$

Recursion bottoms out when $n-k = 1$, i.e., when $k = n-1$

When this happens, we have

$$T(n) = T(1) + c(n + n-1 + \cdots + 1)$$

$$= O(1) + c\left(\frac{n(n+1)}{2}\right)$$

$= O(1) + \Theta(n^2)$

$= \underline{\underline{\Theta(n^2)}}$.

Faster way to do Insert:

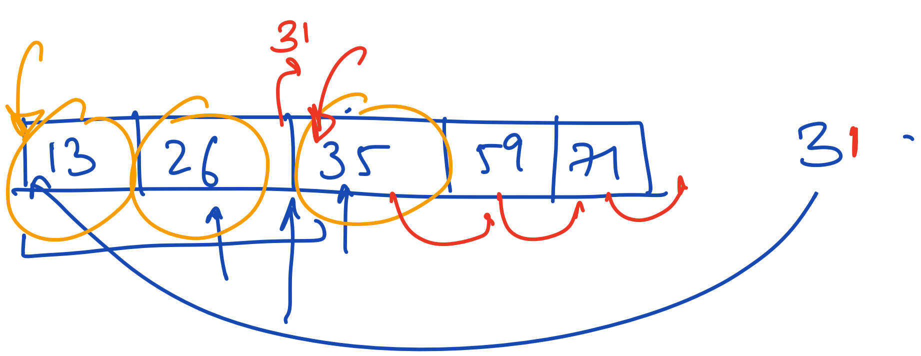
$O(\lg n) \rightarrow$ · Binary search on A to find the loc^n to insert $A[n]$.

$O(1) \rightarrow$ - Add $A[n]$ in the right loc^n in the array.

Runtime recurrence: $T(n) = \begin{cases} O(1), & n = 1 \\ T(n-1) + \lg n, & \text{o.w.} \end{cases}$

n

$T(n) = \Theta(n \lg n)$.



Problem: Even after we find the location of where to insert $A[n]$, we will need to shift the elements in A to the right to make space for $A[n]$. Shifting takes $\Theta(n)$ time in the worst case.

```
IS (A[1..n])
if n == 1 then
    return
else
    Merge
    Insert (IS(A[1..n-1]), IS(A[n..n]))
```

$$\underbrace{\cancel{\text{Insert}}}_{O(n)}\left(\underbrace{IS(A[1..n-1])}_{T(n-1)},\ \underbrace{IS(A[n..n])}_{T(1)}\right)$$

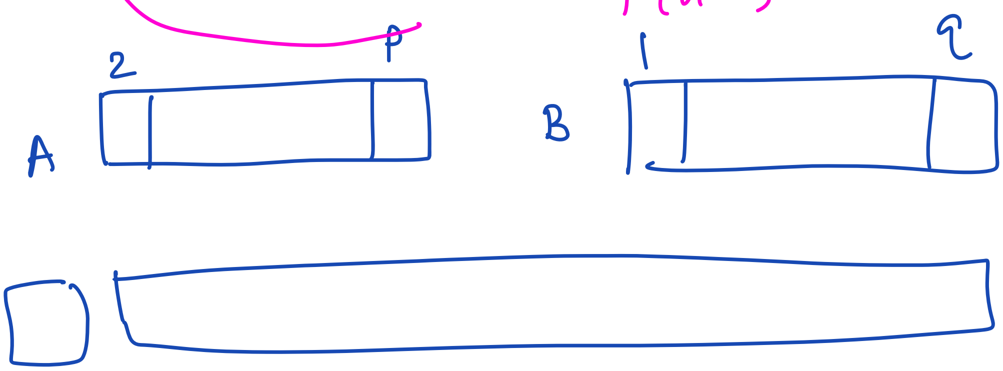
$$T(n) = \begin{cases} O(1), & \text{if } n = 1 \\ T(n-1) + O(n), & \text{o.w.} \end{cases}$$

$$= \Theta(n^2).$$

Merge: merges two sorted arrays into one sorted array.

$$\text{Merge}\left(\overset{\text{Sorted}}{A[1..p]},\ \overset{\text{Sorted}}{B[1..q]}\right) \rightarrow O(n).$$

if $p = 0$ **then**
    return B
else if $q = 0$ **then**
    return A
else if $A[1] < B[1]$ **then**
    $\text{Prepend}\left(A[1],\ \underbrace{\text{Merge}(A[2..p], B[1..q])}_{T(n-1)}\right)$



else

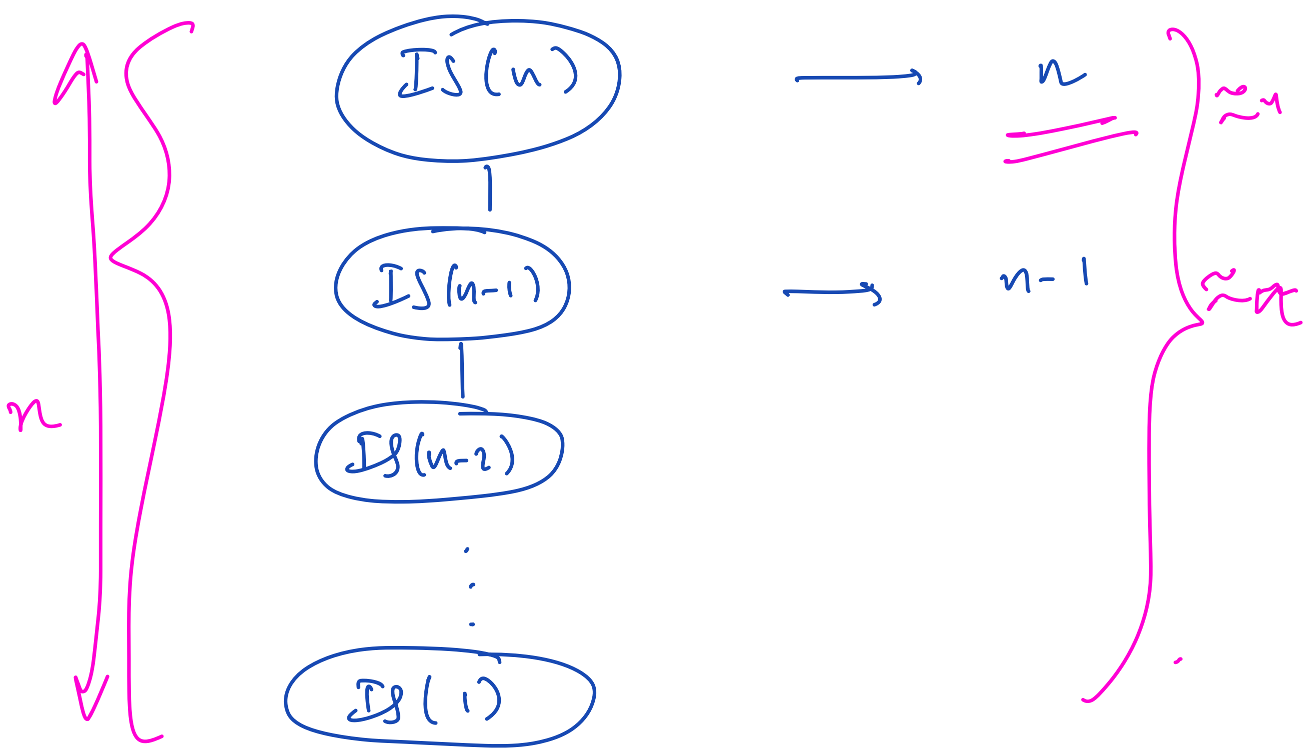
Prepend(B[1], Merge(A[1..p], B[2..q]))

Runtime recurrence of Merge.

If $p+q = n$.

$$T(n) = T(n-1) + c$$

$$= O(n).$$



$MS(A[1..n])$

if $n == 1$ <u>then</u>

<u>return</u> A

<u>else</u>

$$\underbrace{\text{Merge}}_{O(n)}\Big(\underbrace{MS(A[1..\tfrac{n}{2}])}_{T(n/2)},\ \underbrace{MS(A[\tfrac{n}{2}+1..n])}_{T(n/2)}\Big)$$

<u>Runtime recurrence</u>

$$T(n) = \begin{cases} O(1), & n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n), & \text{o.w.} \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$, where c is a const.

$$= 2\left[2T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] + cn$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + cn + cn$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 3cn$$

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + kcn$$

Recursion bottoms out when $n/2^k = 1$, i.e., $k = \lg n$.

When this happens, we get

$$T(n) = 2^{\lg n} T(1) + \lg n \cdot (cn)$$

$= n \cdot O(1) + cn\lg n$

$= \underline{\underline{\Theta(n \lg n)}}$

It is known that the lower bound on the running time of any comparison-based sorting algorithm is $\Omega(n \lg n)$.

Counting Inversions.

Input: Array A of n distinct integers.

Output: # inversions in array A.

A[i] and A[j] are inverted iff $i < j$, but $A[i] > A[j]$.

A: $\quad 26 \quad 14 \quad 39 \quad 3 \quad 31$

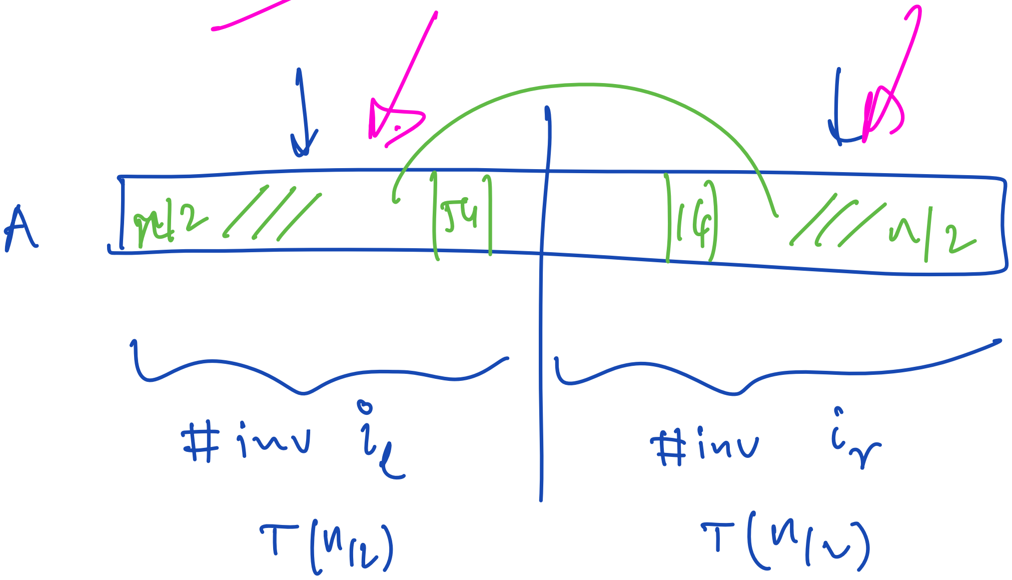
Our output should be 5.

<u>Naive alg</u>: Compare every pair of elements and check if they are inverted.

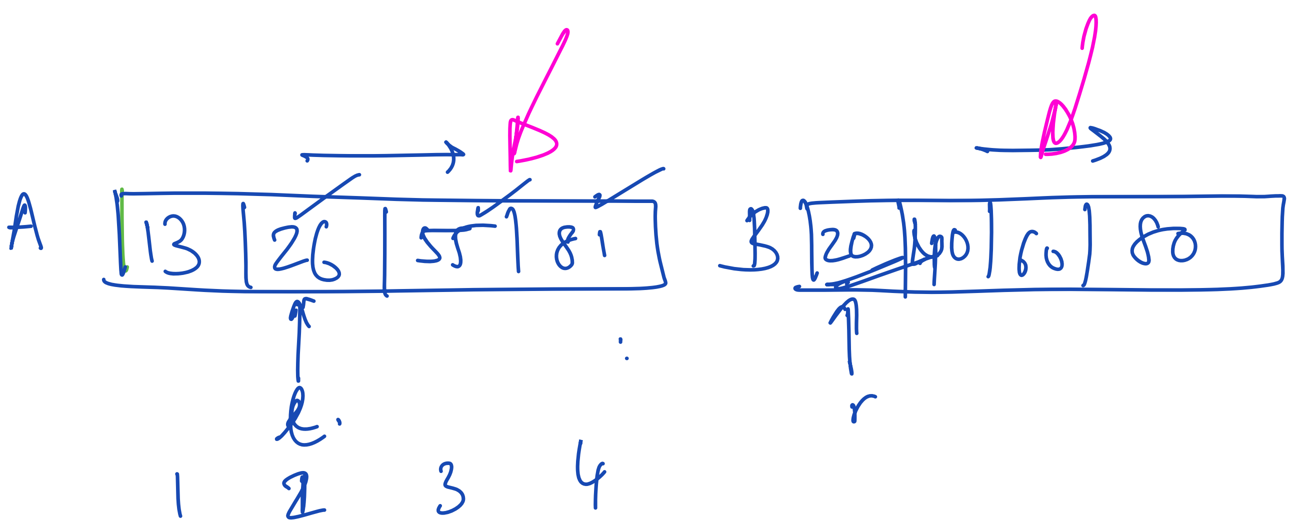
Running time: $\Theta(n^2)$.

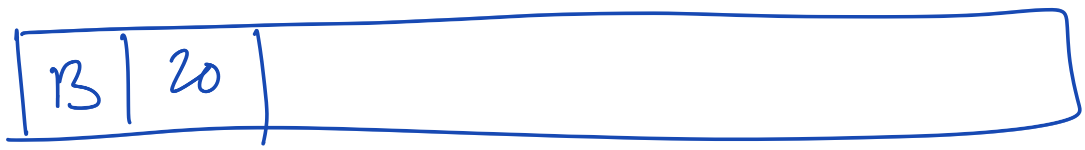
Target runtime: $\Theta(n \lg n)$ (Divide & Conquer)

$$T(n) = \begin{cases} O(1), & \text{if } n=1 \\ 2T\left(\frac{n}{2}\right) + c\cdot n, & \text{o.w.} \end{cases}$$



Our ans: $i_l + i_r$.





$$\#inv = \#inv + \cancel{X}\ |A| - \ell + 1$$