Lecture 21, 28 October 2025

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Lecture 21, 28 Oct 2025

Asymptotic worst-case complexity

Worst Case – reconstruct the most complicated input

$T(n)$ – n input size

$f(n) = O(g(n))$ $\exists c.\ \forall n\ \ f(n) \leq c \cdot g(n)$

$55n^2 + 32n \sim O(n^2)$

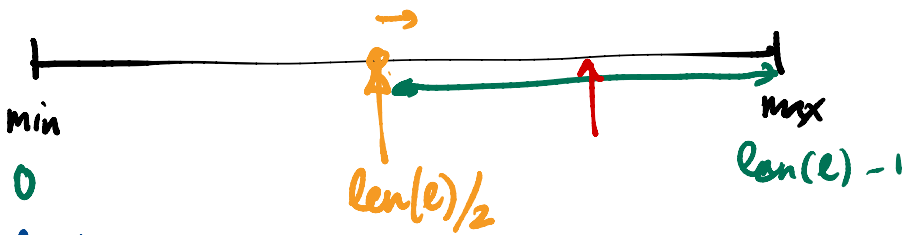
Orders of magnitude

Input size	Values of $t(n)$						
	$\log n$	n	$n \log n$	n^2	n^3	2^n	$n!$
10	3.3	10	33	100	1000	1000	10^6
100	6.6	100	66	10^4	10^6	10^{30}	10^{157}
1000	10	1000	10^4	10^6	10^9		
10^4	13	10^4	10^5	10^8	10^{12}		
10^5	17	10^5	10^6	10^{10}			
10^6	20	10^6	10^7	10^{12}			
10^7	23	10^7	10^8				
10^8	27	10^8	10^9				
10^9	30	10^9	10^{10}				
10^{10}	33	10^{10}	10^{11}				

Searching

Unsorted sequence – $O(n)$ — v in l

Sorted sequence – binary search — $O(\log_2(n))$



log(n) steps → search interval is 1

Searching

$\log_2(n)$ vs $\log_3(n)$

Searching a sorted list — binary search

```
def binarysearch(v,l):
  if l == []:
    return(False)

  m = len(l)//2

  if v == l[m]:
    return(True)

  if v < l[m]:
    return(binarysearch(v,l[:m]))
  else:
    return(binarysearch(v,l[m+1:]))
```

Sorting

Binary search requires a sorted list

Ascending order

Two "intuitive" strategies

Sorting

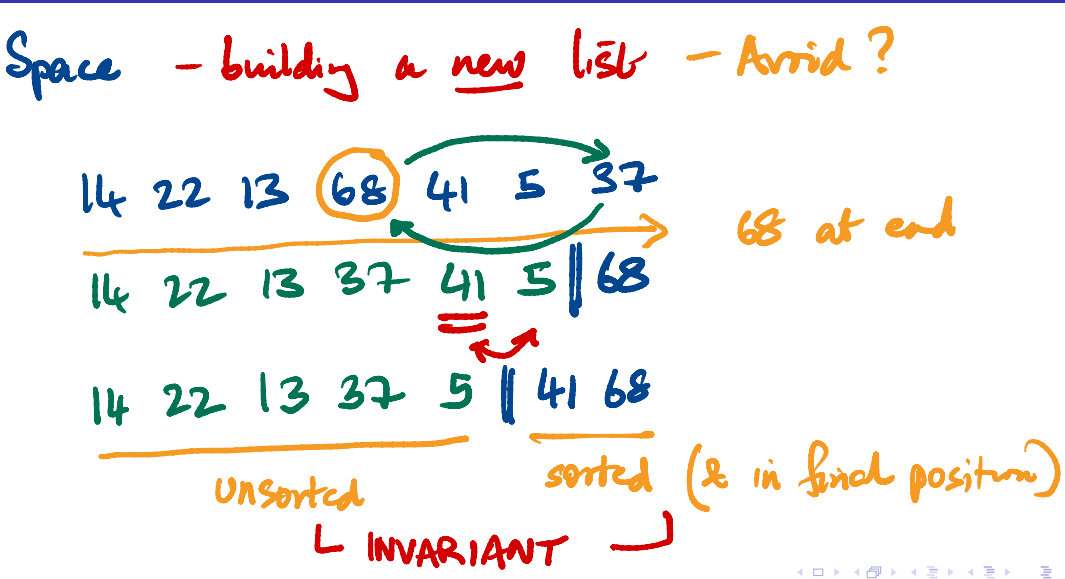
5 13 14 22 37 41 68

Compute min/max in one pass

41 68

"Selection Sort"

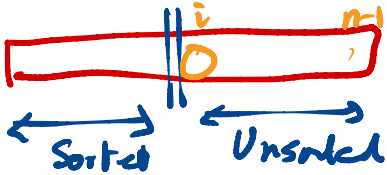
Sorting


Space - building a new list - Avoid ?
14 22 13 68 41 5 37
68 at end
14 22 13 37 41 5 68
14 22 13 37 5 41 68
unsorted
sorted (& in final position)
INVARIANT

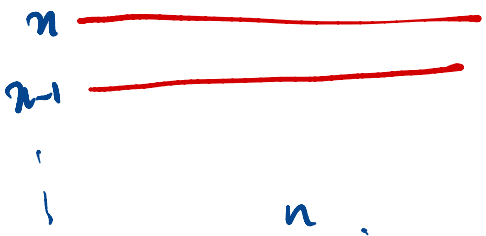
Selection sort

```
def SelectionSort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        # Assume L[:i] is sorted
        mpos = i
        # mpos: position of minimum in L[i:]
        for j in range(i+1,n):
            if L[j] < L[mpos]:
                mpos = j
        # L[mpos] : smallest value in L[i:]
        # Exchange L[mpos] and L[i]
        (L[i],L[mpos]) = (L[mpos],L[i])
        # Now L[:i+1] is sorted
    return(L)
```

Base case



Analysis of Selection Sort?



Find min of $L[0:n]$

Find min of $L[1:n]$

$1+2+\cdots+n$

$$\sum_{i=1}^{n} i$$

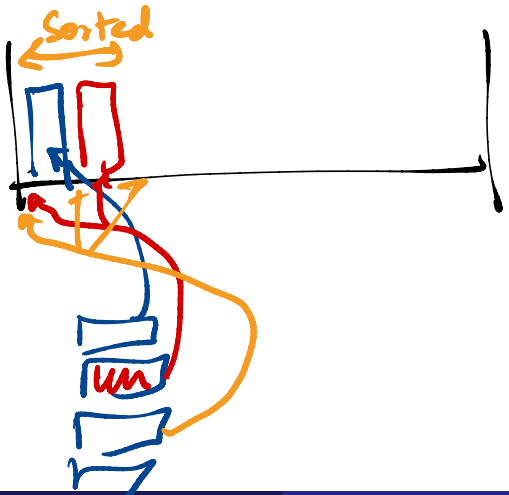
$$\frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = O(n^2)$$

Selection sort always takes $n+(n-1)+(n-2)\cdots+1$ steps

[Check if L is sorted in one pass]

Worst Case applies to every input

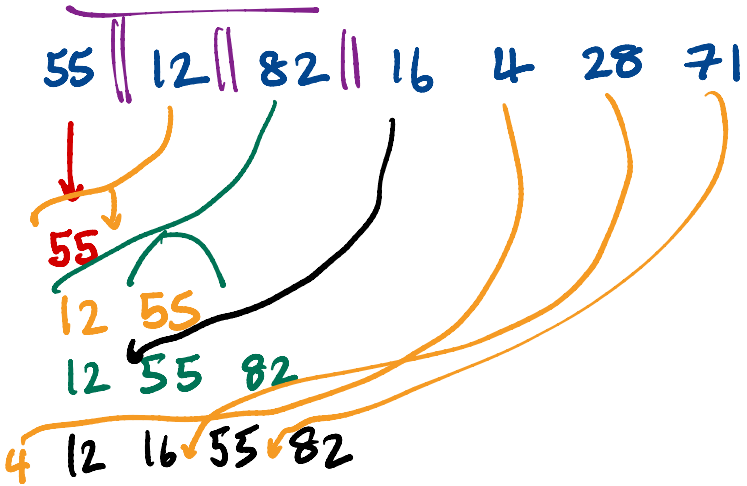
Another natural sort



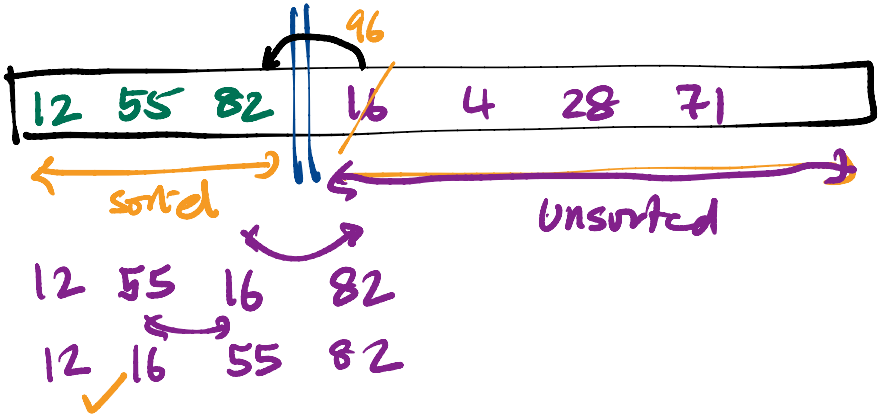
Insert each book in its correct place

Insertion Sort

Sorting



In place

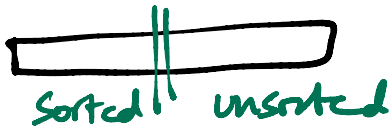


Insertion sort

```
def InsertionSort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        # Assume L[:i] is sorted
        # Move L[i] to correct position in L[:i]
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
        # Now L[:i+1] is sorted
    return(L)
```

Invariant

– Extended sorted segment



Insertion sort

```
def InsertionSort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        # Assume L[:i] is sorted
        # Move L[i] to correct position in L[:i]
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
        # Now L[:i+1] is sorted
    return(L)
```

Analysis of insertion sort

$0+1+2 \ldots + n-1$

$$\frac{n \cdot (n-1)}{2} \Rightarrow O(n^2)$$

Descending order

5 4 3 2 1

4 3

What if input is already sorted?

O(n)

"Almost" sorted lists - good

Insertion sort

```
def InsertionSort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        # Assume L[:i] is sorted
        # Move L[i] to correct position in L[:i]
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
        # Now L[:i+1] is sorted
    return(L)
```

Recursively

Insert operation

insert (l, v) = l + [v] if l[-1] ≤ l[v]

if l[-1] > l[v]

insert (l[:-1], v) + [l[-1]]

left to right

```
if v <= l[0]
    return [v]+l
else
    return [l[0]]+ insert(v, l[1:])
```

Insertion Sort

Sort l[1:]

Insert l[0] into this list

Insertion sort

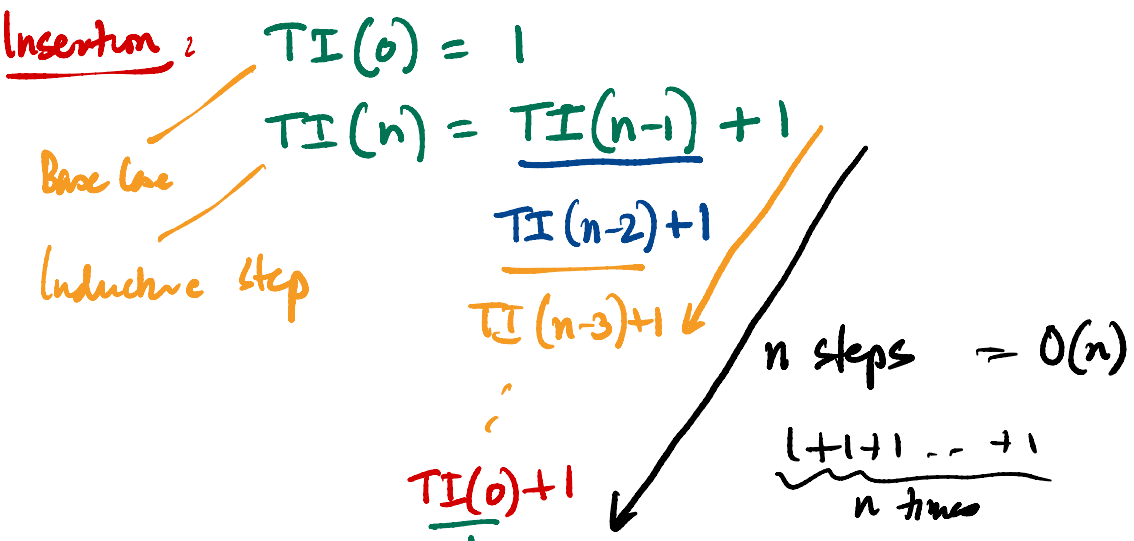
```
def Insert(L,v):
    n = len(L)
    if n == 0:
      return([v])
    if v >= L[-1]:
      return(L+[v])
    else:
      return(Insert(L[:-1],v)+L[-1:])

def ISort(L):
    n = len(L)
    if n < 1:
       return(L)
    L = Insert(ISort(L[:-1]),L[-1])
    return(L)
```

Stick v at end

Analysis ?

TI(n) – Insert v in n elem list

TS(n) – Sort n elements using insertion


Insertion
TI(0) = 1
TI(n) = TI(n-1) + 1
Base Case
Inductive step
TI(n-2)+1
TI(n-3)+1
TI(0)+1
n steps = O(n)
1+1+1 .. +1
n times

ISort.

$TS(0) = 1$

$TS(n) = TS(n-1) + TI(n-1)$

$n-1$

$TS(n-2) + TI(n-2)$

$n-2$

n terms

$TS(0) + TI(0)$

1 1

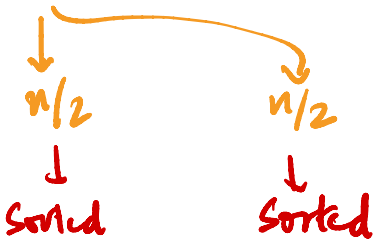
$1+1+2+\dots+(n-2)+(n-1)$

Sorting

2 "naive" sorting algorithms — both $O(n^2)$

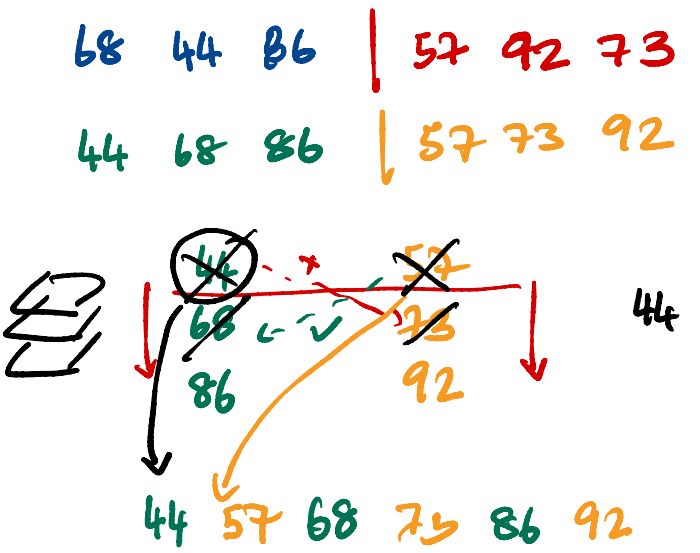
Different strategy

n exam papers, graded, to sort by marks

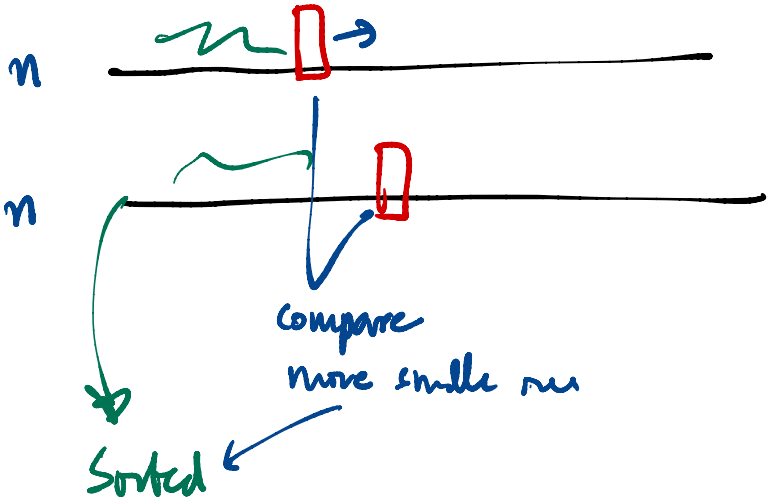


Split the work to 2 TAs

Sorting


68 44 86 | 57 92 73
44 68 86 | 57 73 92
"merge" these sorted lists
44 is smallest overall
44 57 68 73 86 92

Sorting



Each comparison adds one item to sorted output

2n outputs

2n steps

Boundary conditions

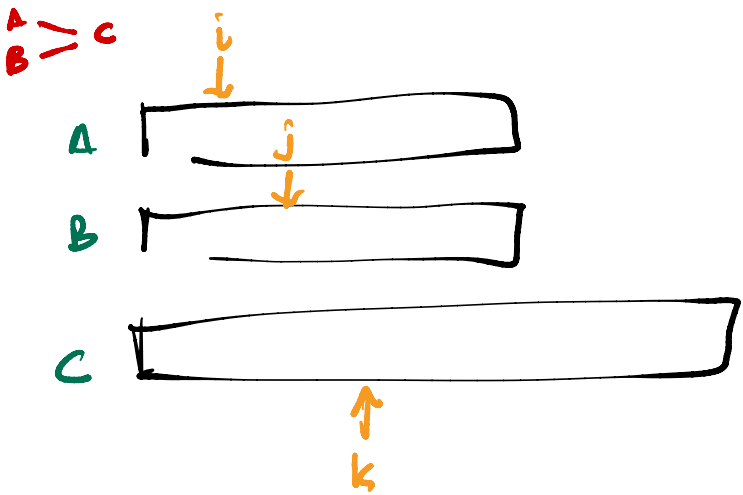
1 2 3 4 X

5 6 7 8

1 2 3 4

Merging sorted lists

```
def merge(A,B):
  (m,n) = (len(A),len(B))
  (C,i,j,k) = ([],0,0,0)
  while k < m+n:
    if i == m:
      C.extend(B[j:])
      k = k + (n-j)
    elif j == n:
      C.extend(A[i:])
      k = k + (m-i)
    elif A[i] < B[j]:
      C.append(A[i])
      (i,k) = (i+1,k+1)
    else:
      C.append(B[j])
      (j,k) = (j+1,k+1)
  return(C)
```

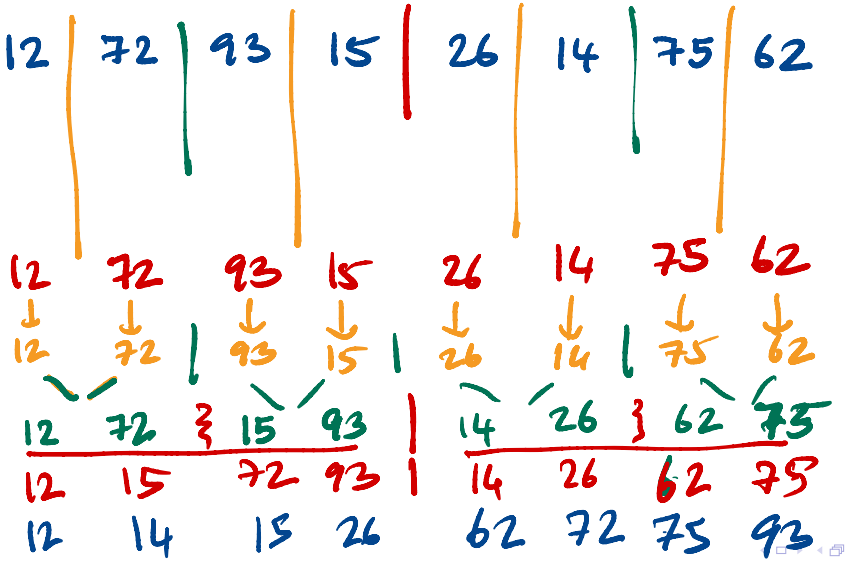


Merge sort

```
def merge(A,B):
  (m,n) = (len(A),len(B))
  (C,i,j,k) = ([],0,0,0)
  while k < m+n:
    if i == m:
      C.extend(B[j:])
      k = k + (n-j)
    elif j == n:
      C.extend(A[i:])
      k = k + (m-i)
    elif A[i] < B[j]:
      C.append(A[i])
      (i,k) = (i+1,k+1)
    else:
      C.append(B[j])
      (j,k) = (j+1,k+1)
  return(C)
```

```
def mergesort(A):
  n = len(A)

  if n <= 1:
    return(A)

  L = mergesort(A[:n//2])
  R = mergesort(A[n//2:])

  B = merge(L,R)

  return(B)
```

Merge sort


12 72 93 15 26 14 75 62
12 72 93 15 26 14 75 62
12 72 15 93 14 26 62 75
12 15 72 93 14 26 62 75
12 14 15 26 62 72 75 93

Merge sort analysis

Divide & conquer

Analysis ?