Unbalanaced Partitioning

- problems based on different

Design Technique

- Instance of problem. you need
  Solve that instance be problem

Application based

- Clear understanding
- Concept

question also asked

time complexity of
algorithm

- knapsack (fractional)
- Merge pattern
- Huffman
- topological sorting
- strongly connected
  compont

$$\underset{100}{T(n)} = \underset{33}{T(n/3)} + \underset{66}{T(2n/3)} + n$$

Right Heavy

an/b

$$n/3 + 2n/3 - n$$

Height - $\log_{n/a} n$

$$n + n + n + \dots + n = n\log n$$

height No. $\log_{3/2} n$

Cost at each level is n or same

Height * cost

2021

$$T(n) = T(n/2) + T(2n/5) + 7n$$

approximate

$$T(n/2) \qquad T(2n/5)$$

$$7 \cdot n/2 \qquad 7 \cdot 2n/5$$

$$7n\left[\frac{1}{2} + \frac{2}{5}\right] = 7n\left[\frac{5+4}{10}\right] = 7n\left(\frac{9}{10}\right)$$

Infinite

$r < 1$

$$7n\left[1 + \left(\frac{9}{10}\right) + \left(\frac{9}{10}\right)^2 + \dots \left(\frac{9}{10}\right)^{k-1}\right] \approx 7n\left(\frac{1}{1 - 9/10}\right)$$

$$\approx 70 \quad \theta(n)$$

$r > 1$ apply formula for GP series & get the bound.

1 question

Unbalanced Partitioning

Suppose partition algorithm splits in size of $\frac{n}{10}$ and $\frac{9n}{10}$ at every stage

10% 90% nLogn

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$

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

quick sort may asymmetric

Recurrence Relation

which side will decide height

$T(n)$ — n

$T(n/10)$ $n/10$ $T(9n/10)$ – $9n/10$ — $\frac{n}{10} + \frac{9n}{10} = \frac{10n}{10} = n$

$T(n/100)$ $T(9n/100)$ $T(9n/100)$ $T(81/100\, n)$ – an/b $9n/10$

$\log_{10/9} n$

Height = $\log_{10/9} n$

Complexity $n \cdot \log_{10/9} n$

Unbalanced Partitioning

Space Complexity of quick sort :

$\frac{n+n}{n}$ $O(n)$

1. Input Size — n
2. Auxiliary Space ×
3. Stack space — $\underline{n}$ $T(0) = T(n-1) + n + T(0)$
4. Constant Space

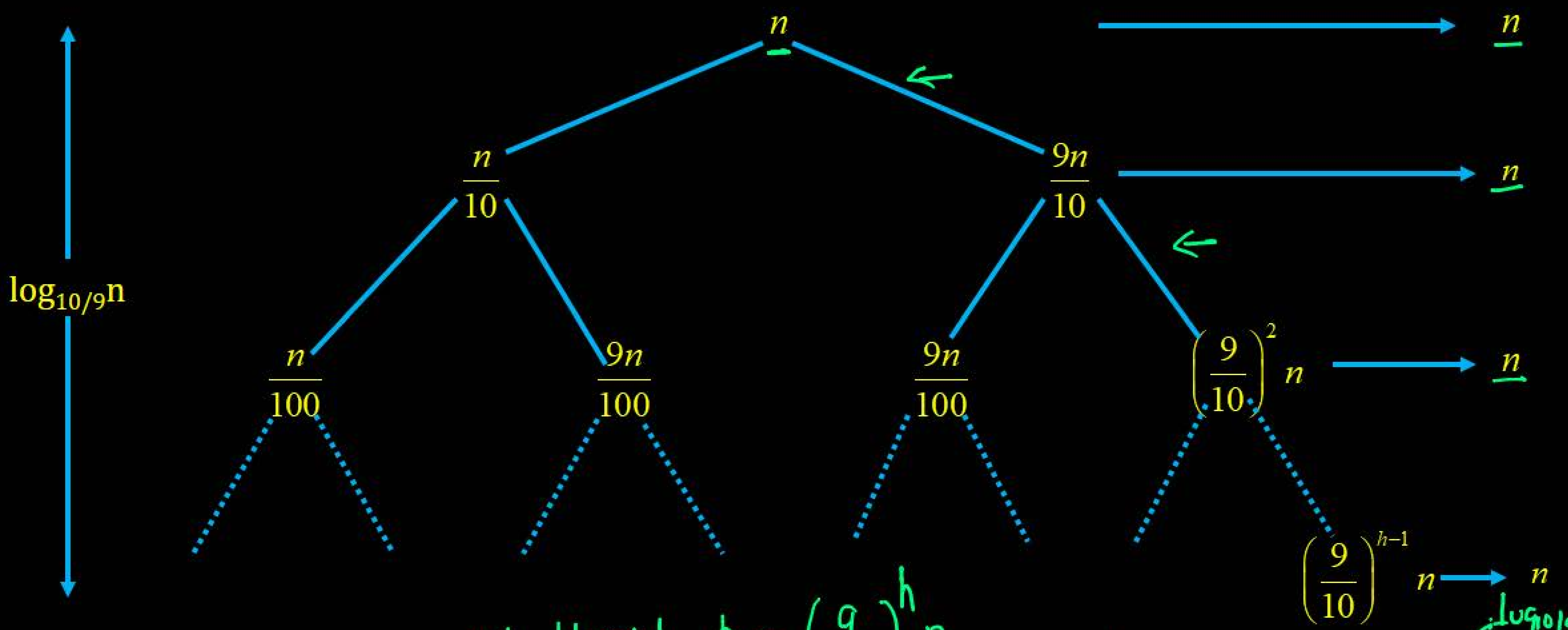
Best case — Input Size - n

Stack space - $\underline{\log_2 n}$ $T(n) = 2\underline{T(n/2) + n}$

$\underline{n + \log n}$

$T(n) = T(n/2) + T(n/2)$

Unbalanced Partitioning



at Height h = $\left(\frac{9}{10}\right)^h n$

$\log_{10/9} n$

$\left(\frac{9}{10}\right)^h n = 1$

$\Rightarrow n = (10/9)^h \Rightarrow h = \log_{10/9} n$

GATE 2009, Question Number 39, 2-Marks

In quick sort, for sorting n elements, the $(n/4)^{th}$ smallest element is selected as pivot using an $O(n)$ time algorithm. What is the worst case time complexity of the quick sort?

(A) $\theta(n)$ (B) $\theta(n \log n)$ (C) $\theta(n^2)$ (D) $\theta(n^2 \log n)$

partition ( )

$T(n) = T(n/4) + T(3n/4) + n + n$

Selection of pivot

time complexity of partition

$T(n) = T(n/4) + T(3n/4) + 2n$

$\theta(n \log n)$

cost will Remain same

Randomized Quick Sort

Randomized-Quicksort: Run the Quicksort algorithm as given above, each time picking a random element in the array as the pivot.

GATE 2001| 1 Mark Question

Randomized quicksort is an extension of quick sort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?

(A) $O(n)$ (B) $O(n \log n)$ (C) $O(n^2)$ (D) $O(n!)$

median of an array

• If Select the median in O(n) time and use it as pivot the what is the complexity of quick Sort?

Sorted

1, 2, 3, 4 5

median

middle element if n is odd

1, 2, $\boxed{3, 4}$, 5, 6

$(3+4)/2 = 7/2 = 3.5$

Lecture -13

Merge Sort

Merge Sort

Merge Sort

- $a[1:10] =$

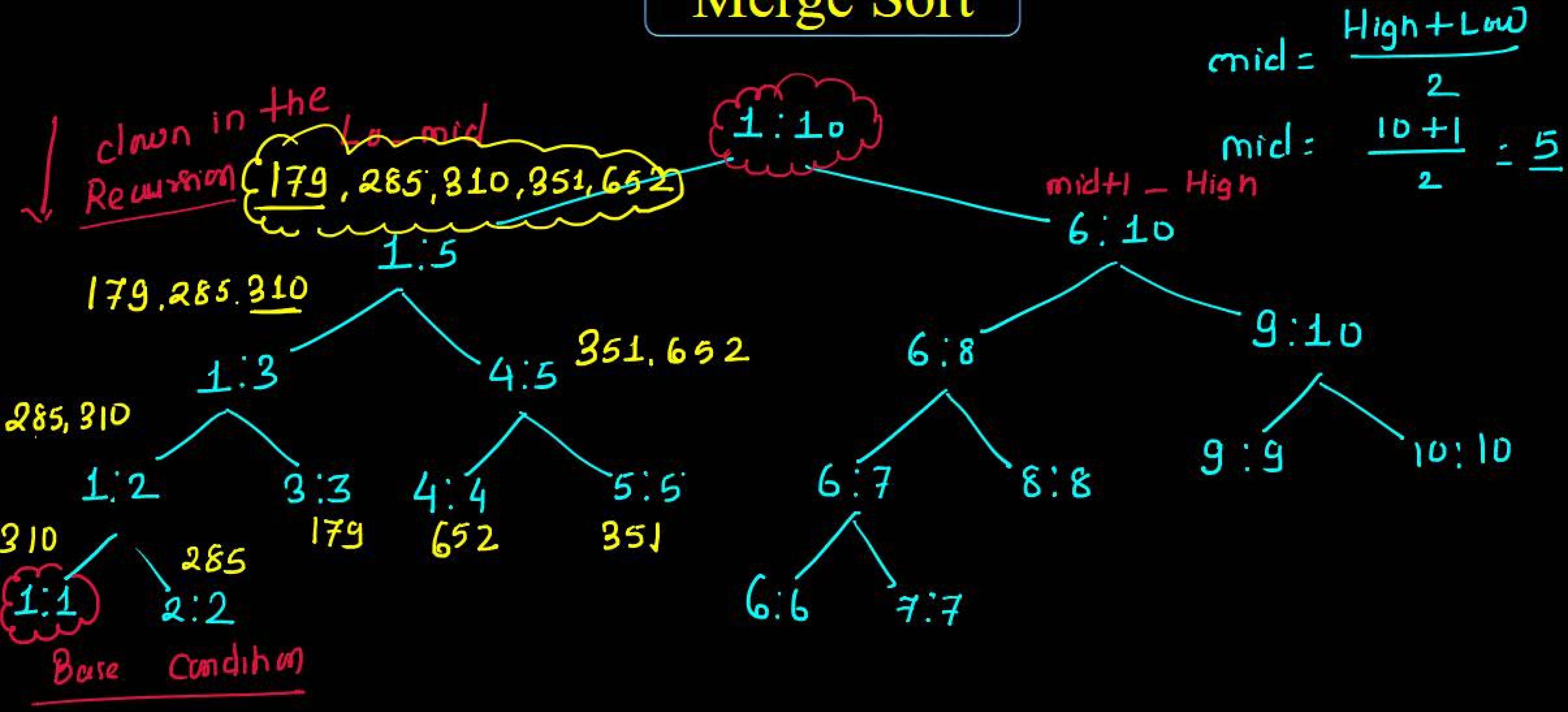
(310, 285, 179, 652,351423. 861, 254, 450, 520).

- divide & Conquer strategy is an approach which divides the array in equal partition. This process is repeated till each partition consists of single element. (Top down) (going down in Recursion).
- On comming out of the Recursion it starts merging elements

question Related
- Heap sort
- Bubble
- Insertion
- Selection
- Radix sort

Seperatly at the end

Merge Sort



- $a[1: 10]$ =

(310, 285, 179, 652,351423. 861, 254, 450, 520).

1 2 3 4 5 6 7 8 9 10

Merge Sort

- Consider the array often elements
- $a[1{:}\ 10] = (310,\ 285,\ 179,\ 652{,}351423.\ 861,\ 254,\ 450,\ 520).$
- Algorithm Merge Sort begins by splitting a[ ] into two subarrays each of size five (a[l : 5] and a[6 : 10]).

Merging Algorithm of two sorted:

mid

high

1	3	5		2	4	6

~~i~~ ~~j~~ $\boxed{j}$ ~~k~~ h h

1, 2 3, 4, 5, 6

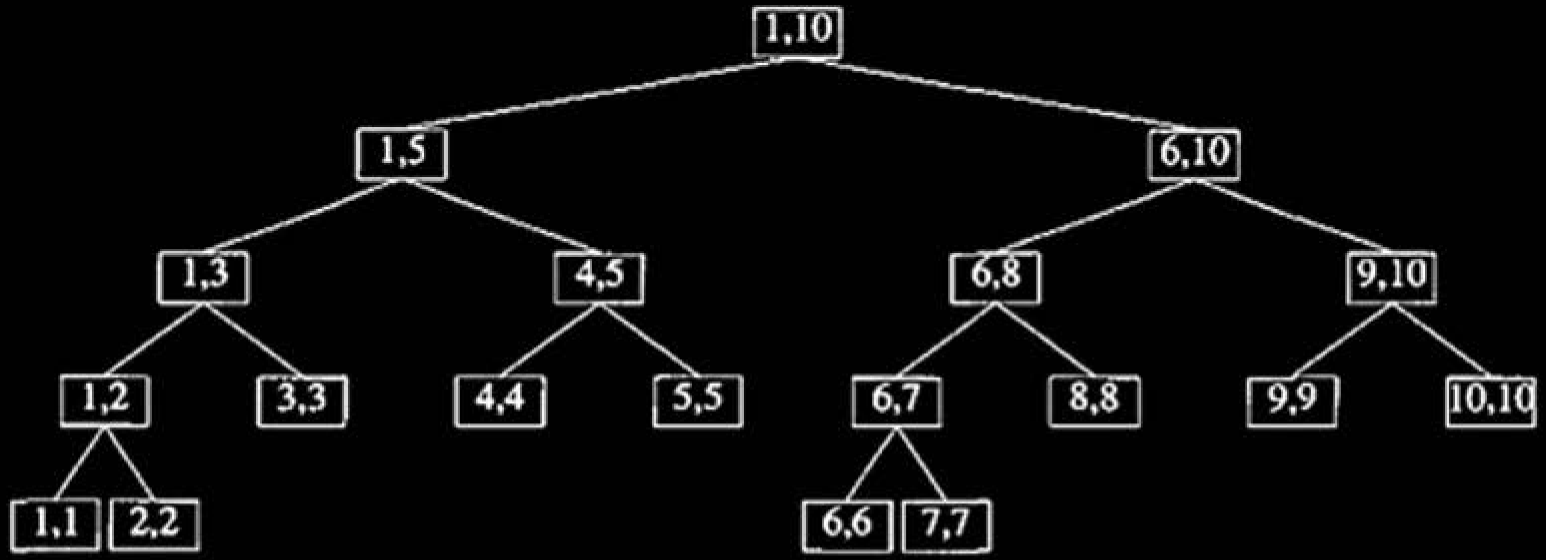
Merging two sorted array into a single sorted array

Merge Sort Partition

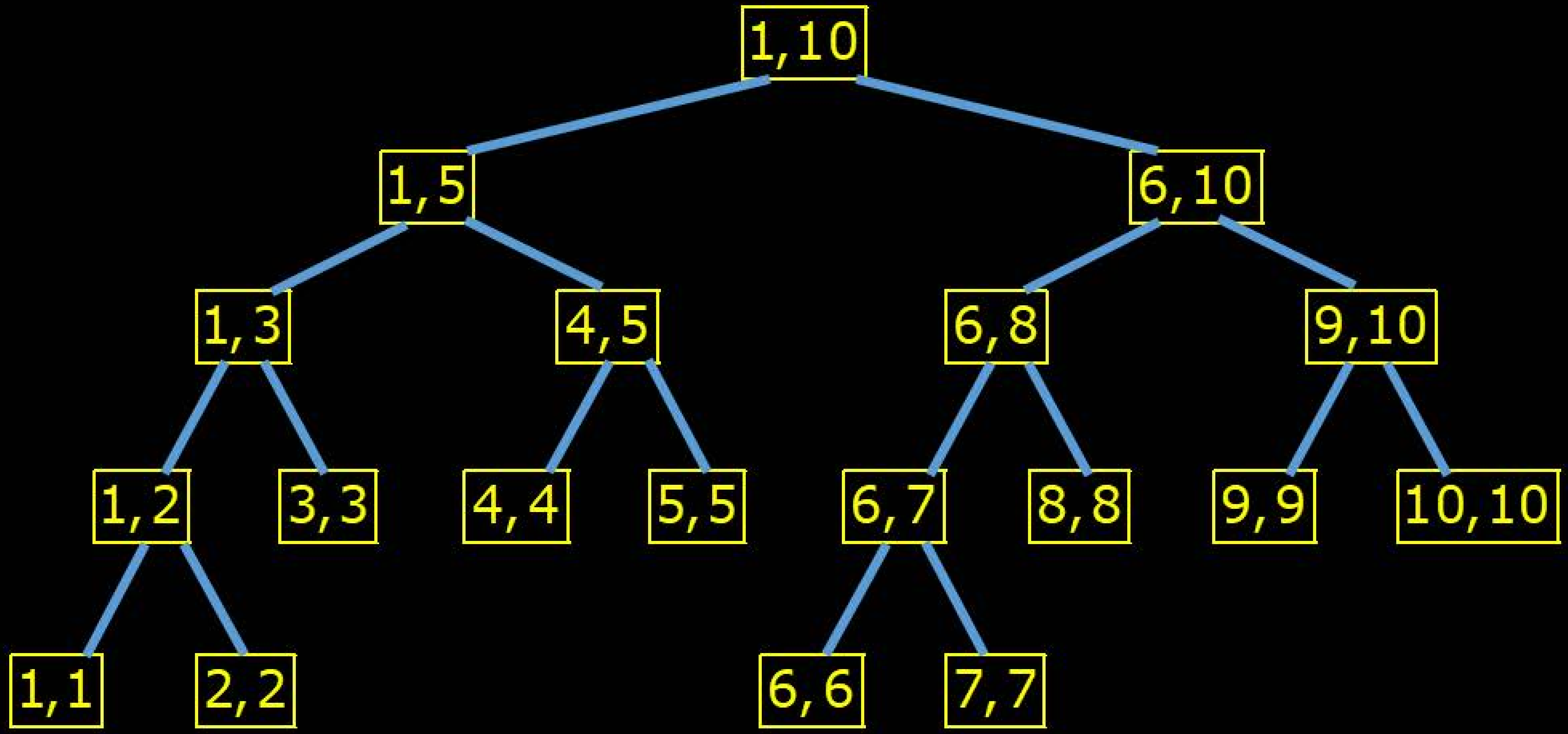
(310 285 179 652, 351 423, 861, 254, 450, 520)

$\boxed{1,10}$

Merge Sort Partition

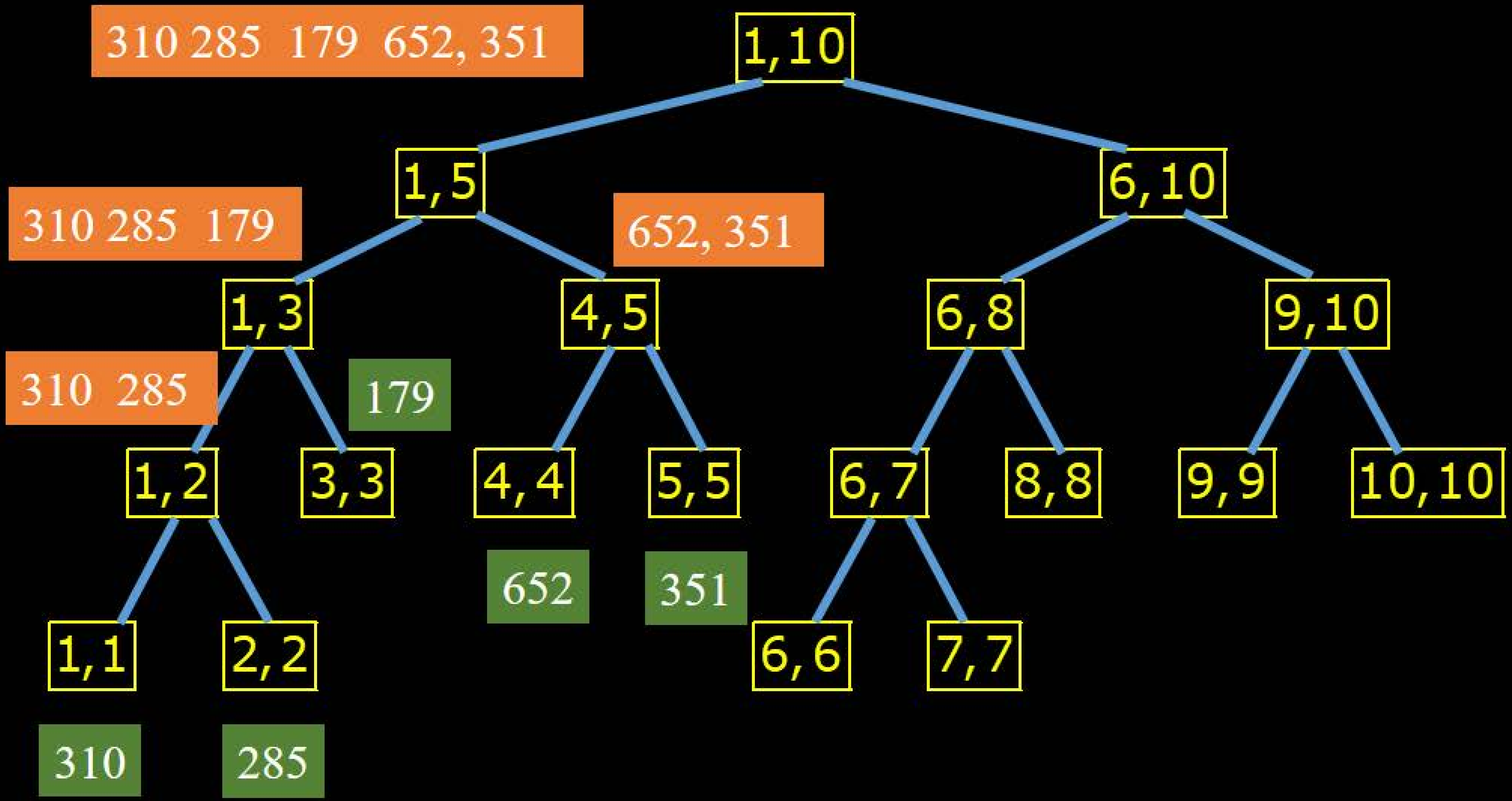


Merge Sort Partition



Merge Sort Partition

(310 285 179 652, 351 423, 861, 254, 450, 520)

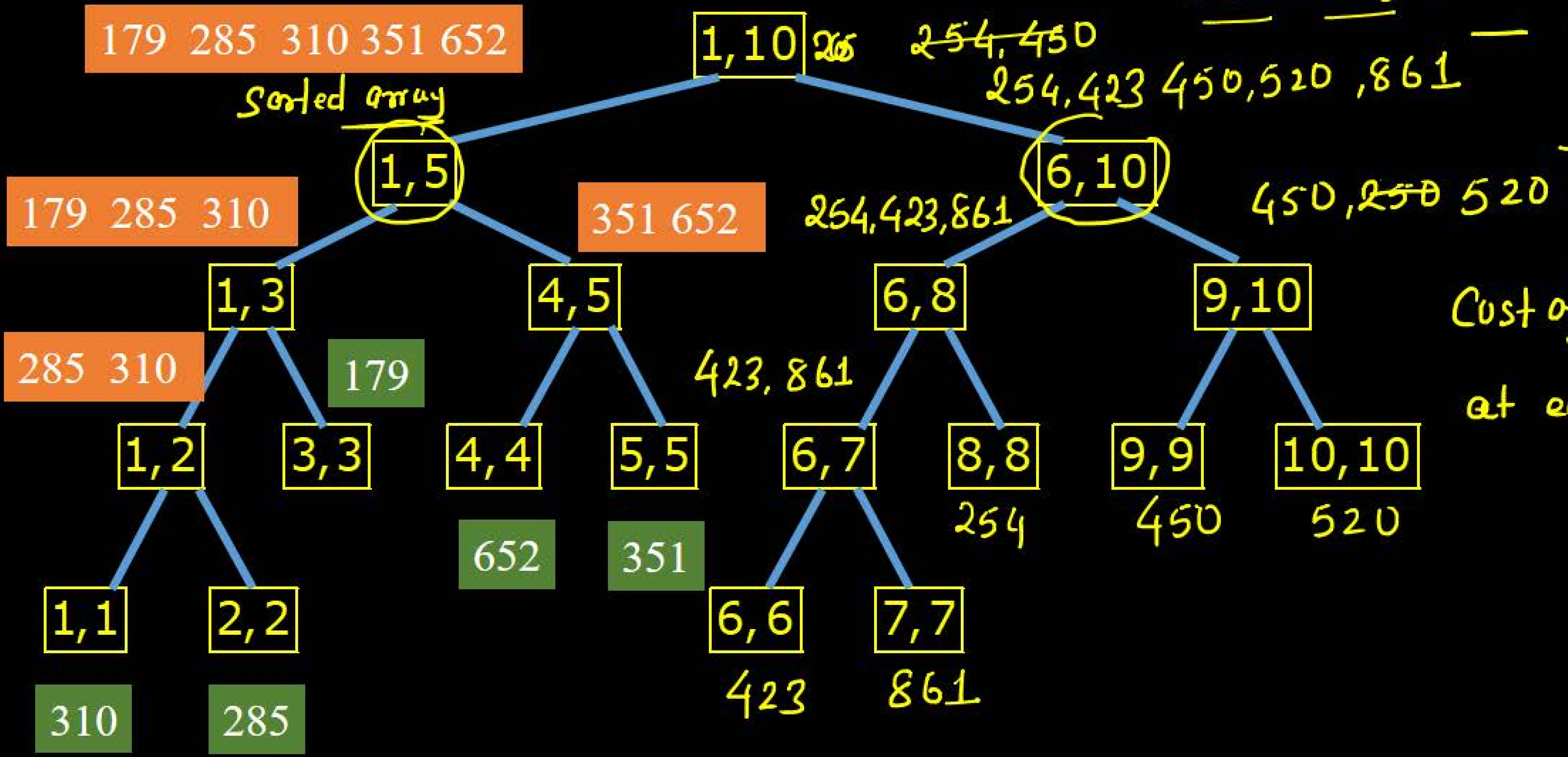


Merge Sort Partition

(310 285 179 652, 351 423, 861, 254, 450, 520)

Low . mid mid+1 — high

b[Low - High]



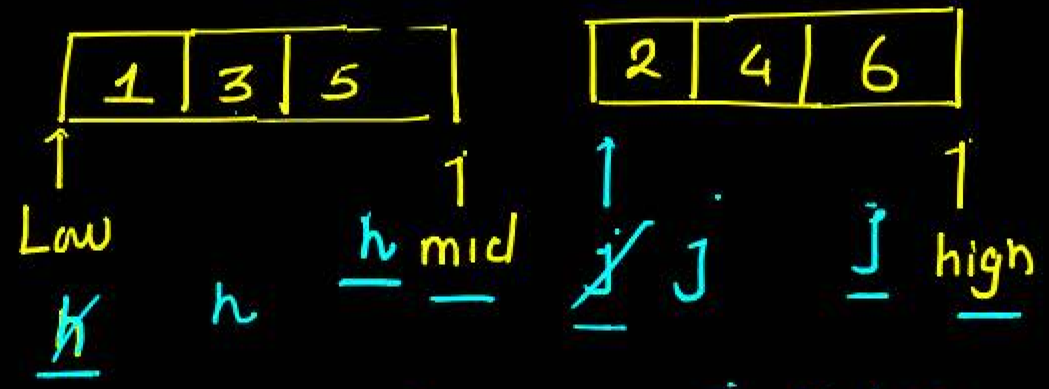
Cost of merging at each level

Algorithm

```
Algorithm MergeSort(a,low, high) {
      if (low < high) {          Bin
            mid := [(low + high)]/2;
      MergeSort(a, low, mid);
      MergeSort(a, mid + 1, high);
      Merge(a, low, mid, high);
              1
      }
```

Merging

```
Algorithm merge(low, mid, high){
h := low; i := low; j := mid + 1;
while ((h ≤ mid) and (j ≤ high)) {
      if (a[h] ≤ a[j]) then
          b[i] :=a[h];
          h:= h + 1;
      else {
          b[i] :=a[j];
          j :=j + 1;
      }
      i:=i + 1;
}
```



auxiliary Spare

i

h ≤ mid

j ≤ high

a[h] ≤ a[j]

b 1 2 3 4 5

i

Merging

```
if (h > mid) then
      for k:= j to high do{
              b[k]:= a[k]; i := i + 1;
      }
else
          for k := h to mid do{
              b[i] := a[k]; i := i + 1;
          }
      for    k := low to high
      a[k] := b[k];
}
```

MergeSort is Not in place.

process of merging

~~B is~~ b[ ] extra space used for holding intermediate result. copied back at the end.

If ~~IV~~ Algorithm required extra space apart from input then that Sorting Algorithm is not in place.

```
Algorithm merge(low, mid, high){
h := low; i := low; j := mid + 1;
while ((h ≤ mid) and (j ≤ high))
{
        if (a[h] ≤ a[j]) then
                b[i] :=a[h];
                h:= h + 1;
        else {
                b[i] :=a[j];
                j :=j + l;
        }
        i:=i + l;
}
if (h > mid) then
        for k:= j to hiqh do
        {
                b[k]:= a[k]; i := i + 1;
        }
else
                for k := h to mid do
                {
                        b[i] := a[k]; i := i + 1;
                }
        for   k := low to high
        a[k] := b[k];
}
```

al

mid

1 3 5.	2, 4, 6, 8, 10

j

elements exausted

1 2 3 4, 5, 6, 8, 10

while (h ≤ mid and h ≤ High)

if (h > mid)

for k = j to High

b[i] = a[k]

Press Esc to show floating meeting controls

Analysis of Merging

$T(n) = 2T(n/2) + n$ ← Loop has to Run from Low — High

cost of merging

Nice

Merge Sort having Best, worst and Average case as $\theta(n \log n)$

Space — n — Input
n — Auxilary
logn — stac

Case(II)

$a = 2$

$b = 2$

$n^{\log_b a} = n^{\log_2 2}$

$f(n)$ is $\underline{n}$

$f(n)$ is $\theta(n^{\log_2 2} \log^0 n)$

$T(n) = \theta(n^{\log_2 2} \log^{0+1} n)$

$= \theta(n \log n)$

Complexity

- $T(n) = \begin{cases} c & if\ n = 1, \\ 2T\left(\frac{n}{2}\right) + cn & if\ n > 1, \end{cases}$

Problem Solving

GATE 1995, Question Number 1.16, 1-Mark

Optimal Merging pattern

For merging two sorted lists of sizes m and n into a sorted list of size m + n, we required comparisons of

time complexity merging O(n+m)

(A) O(m) (B) O(n)

(C) O(m + n) (D) O(logm + logn)

Comparison

Merging is Linear tim

Answer maximum No. of Comparison Required – (m+n−1)

Minimum No. of Comparison Required – min(m,n)

Break

1 . . . . m 1 . . . n

h j

(13) (2,3)

| 1 | 2 |

| 3 | 4 | 5 | 6 | 7 | 8 |

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

3 3

| 1, | 3 | 5 | | 2, | 4, | 6 |

1 2 3 4, 5 6

(1,2)
(3,2)
(3,4)
(5,4)
(5,6)

3+3−1

No. of Comparisons

GATE 2003, Question Number 4, 1-Mark

Let A be a sequence of 8 distinct integers sorted in ascending order. How many distinct pairs of sequences, B and C are there such that (i) each is sorted in ascending order, (ii) B has 5 and C has 3 elements, and (iii) the result of merging B and C gives A?

(A) 2 (B) 30 (C) 56 (D) 256

GATE 2012, Question Number 39, 2-Marks

H.W

individual character

A list of n stings, each of length n, is sorted into lexicographic order using the merge-sort algorithm. The worst case running time of this computation is

(A) $O(n\log n)$

(B) $O(n^2 \log n)$

(C) $O(n^2 + \log n)$

(D) $O(n^2)$

A List of n string to sorted what

abcd
adbc
badb
bcad

Lexicographic ordering

Dictonary ordering

- Legthwise ordering
- Relative ordering among alphabet is there $a < b$

Many instances

unordered array – Linear Search

ordered array – Binary Search

Binary Based on a Single Comparison the Size of problem Reduces to $\underline{1/2}$

Binary Search

Binary Search

Binary Search

```
int binarySearch(int arr[], int lb, int ub, int x){
int mid;
if (lb > ub)
      return -1
else{
      mid = (lb + ub)/2;
      if (arr[mid] == x)  return mid;
      else
      if (x < arr[mid])
            return binarySearch(arr, lb, mid-1, x);
      else
            return binarySearch(arr, mid+1, ub, x);
   }}
```

1 1

program terminating

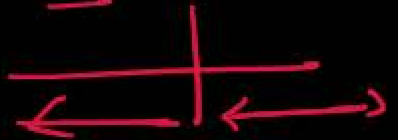
mid ↑ Searching

Lower Bound Binary Search – $\Omega(1)$ –

1 comparison

Comparison with middle element and x

mid is already expu



x<a[mid]

Binary Search

Binary search. Given a key and sorted array a[], find index i such that a[i] = key, or report that no such index exists.

Ex. Binary search for 33.

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Lo ~~*lo*~~ *hi*

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

lo *mid* *hi*

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

lo *hi*

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

lo *mid* *hi*

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

↑ *low* ↑ *hi*

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

↑ ↑ ↑

low ***mid*** ***hi***

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

low

hi

mid

Binary Search

6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

↑

low

hi

mid

stop

Binary Search

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

↑

low

hi

mid

Comparison

↑

worst case time complexity

$$\boxed{T(n) = T(n/2) + 1}$$

$$T(1) = 1$$

for $n = 2^k$

kth term

$$T(n) = T(n/2^k) + k$$

$n = 2^k$

$k = \log_2 n$

$T(1) + \log_2 n$

$1 + \log_2 n$

Number of comparison in worst case

NAT :

mid = $\frac{LB+UB}{2}$

manual approach to find No. of comparison

n

1 comparison — $n/2$

2 comparison — $n/2^2$

⋮

k^{th} comparison — $n/2^k$

$$\frac{n}{2^k} = 1$$

$\Rightarrow n = 2^k$

$\therefore k = \log_2 n$

Time Complexity of Binary Search

Strassen’s Matrix Multiplication

Matrix Addition

$n/2$

```
for (i=1;i<=m;i++)
```

$n/2$

```
    for(j=1;j<=n;j++)
        c[i][j]=a[i][j]+b[i][j]
```

Done - step count

$\underline{m \times n}$

$n/2 * n/2 = n^2/4$

Matrix Multiplication

$$\begin{array}{c} i \\ \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \end{array} \times \begin{array}{c} j=1 \\ \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{array} = \begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} & a_{11} b_{12} + a_{12} * b_{22} \\ a_{21} * b_{11} + a_{22} * b_{21} & a_{21} * b_{12} + a_{22} b_{22} \end{bmatrix}$$

Algorithm for matrix multiplication

Sum of product terms

$i = 1$ - $j = 1$ $k = 1$, $k = 2$

$j = 2$

for $i = 1$ to n

for $j = 1$ to n {

$c[i,j] = 0$

for $k = 1$ to n

$c[i,j] = c[i,j] + a[i,k] * b[k,j]$

Time Complexity

$\Theta(n^3)$

Matrix Multiplication

$$\begin{bmatrix} \underline{a11} & \underline{a12} \\ a21 & a22 \end{bmatrix} \times \begin{bmatrix} \underline{b11} & b12 \\ \underline{b21} & b22 \end{bmatrix} = \begin{bmatrix} a11 * b11 + a12 * b21 & a11 * b21 + a21 * b22 \\ a21 * b11 + a22 * b21 & a21 * b21 + a22 * b22 \end{bmatrix}$$

```
c[i][j]+=a[i][k]*b[k][j]
```

Matrix Multiplication

Matrix Multiplication

```
for (i=1;i<=n;i++)
     for(j=1;j<=n;j++)
          c[i][j]=0
          for(k=1;k<=n;k++)
               c[i][j]+=a[i][k]*b[k][j]
```

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \times \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix} = \begin{bmatrix} a11 * b11 + a12 * b21 & a11 * b12 + a12 * b21 \\ a21 * b11 + a22 * b21 & a21 * b12 + a22 * b22 \end{bmatrix}$$

Number of Addition

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \times \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix} = \begin{bmatrix} a11*b11 + a12*b21 & a11*b12 + a12*b21 \\ a21*b11 + a22*b21 & a21*b12 + a22*b22 \end{bmatrix}$$

Number of Addition = 4

total Addition is 4

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$$= \begin{bmatrix} a11*b11 + a12*b21 + a13*b31 + a14*b41 & \\ & \\ & \end{bmatrix}$$

question

total No. of Addition

No. of position

$n*n\,(n-1) - n^2(n-1)$

$4 \times 4 \times 3 = 48$

Number of Addition = 48

$n^2(n-1)$ – Square ~~Max~~ matrix

$\boxed{n \times p\,(m-1)}$ generalized case $(n \times m)(m \times p)$

Number of Addition

Number of Addition = $(n-1)n^2$

$n \times m$ $(m \times p)$

$np(m-1)$

Number of Multiplication

4

$n \times m$ and $m \times k$ matrix

No. of scalar multiplication

$n \times m \times k$ multiplication required ←

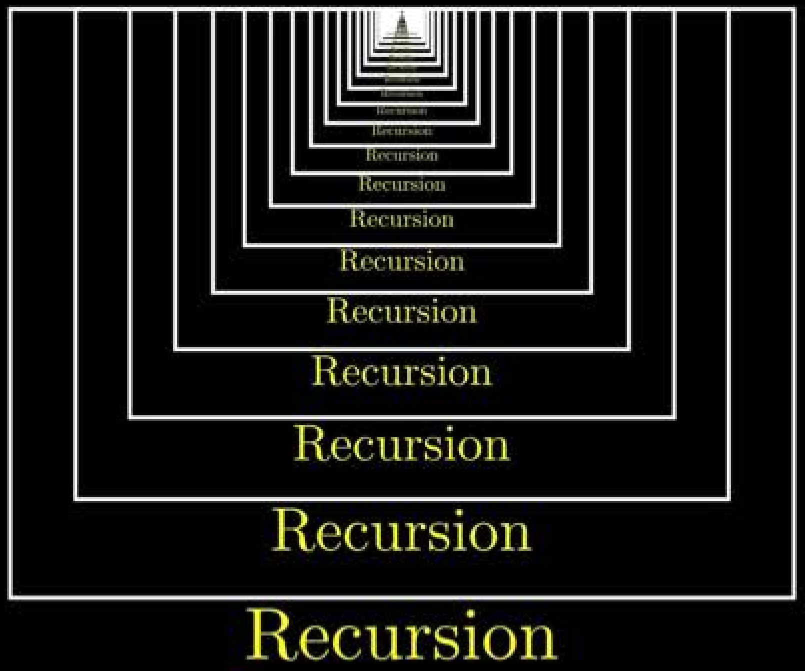
Minimize No. of scalar multiplicat

is Dynamic Programming

Number of Multiplication

$n \times n$ *and* $n \times n$ *matrix*

n^3 *multiplication required*



Recursive Approach

Matrix Multiplication

$\Theta(n^3)$

4x4 - $\boxed{2\times 2}$

$$
\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}
$$

1 2 3 4 5 6 7 8

Break into Smaller instance of problem

n x n squar matrix

1x1 matrix

Divide & Conquer approach

$$
\begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31} + a_{14} * b_{41} & - & - & - \\ 1 * 5 \quad + \quad (2 * 7) & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}
$$

2x2 2x2

1 * 6 + 2 * 8

3 * 5 + 4 * 7

3 * 6 + 4 * 8

Matrix Multiplication

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$$= \begin{bmatrix} a11 * b11 + a12 * b21 + a13 * b31 + a14 * b41 & .. & .. & .. \\ .. & .. & .. & \\ & & & \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$$\times \qquad + \qquad \times$$

Matrix Multiplication

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$\underline{n \times n}$

4×4 – 2×2

$n \times n$ – $\underline{\left(n/_2 \times n/_2\right)}$

$\times \qquad + \qquad \times$

$$\begin{bmatrix} a11*b11+a12*b21 & a11*b12+a12*b22 \\ a21*b11+a22*b21 & a21*b12+a22*b22 \end{bmatrix} + \begin{bmatrix} a13*b31+a14*b41 & a13*b32+a14*b42 \\ a23*b31+a24*b41 & a23*b32+a24*b42 \end{bmatrix}$$

$$\begin{bmatrix} a11 * b11 + a12 * b21 + a13 * b31 + a14 * b41 & .. \\ .. & .. \end{bmatrix}$$

Matrix Multiplication

$A11 \quad A12 \qquad B11 \quad B21$

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$\times \qquad + \qquad \times$

$$\begin{bmatrix} a11*b11+a12*b21 & a11*b12+a12*b22 \\ a21*b11+a22*b21 & a21*b12+a22*b22 \end{bmatrix} + \begin{bmatrix} a13*b31+a14*b41 & a13*b32+a14*b42 \\ a23*b31+a24*b41 & a23*b32+a24*b42 \end{bmatrix}$$

$$\begin{bmatrix} a11*b11+a12*b21+a13*b31+a14*b41 & .. \\ .. & .. \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a41 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix}$$

$\times \qquad + \qquad \times$

Matrix Multiplication

$$\left[\begin{array}{cc|cc} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ \hline a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{array}\right] \times \left[\begin{array}{cc|cc} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ \hline b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{array}\right]$$

A11 A12 A21 A22

B11 B12 B21 B22

$\frac{n}{2} \times \frac{n}{2}$

Multiple(A,B,n) {

total - $n/2$ is 8 times called with size $n/2$

Multiply(A11, B11, n/2) + Multiply(A12, B21, n/2) $- n^2/4$

Multiply(A11, B12, n/2) + Multiply(A12, B22, n/2) $- n^2/4$ addition of mam,

Multiply(A21, B11, n/2) + Multiply(A22, B21, n/2) $- n^2/4$

Multiply(A21, B12, n/2) + Multiply(A22, B22, n/2) $- n^2/4 =$

$4 * n^2/4$

Recursive Matrix Multiplication

Input: A, B ∈ $R^{n \times n}$

Output: AB

```
    function M(A,B, n)
      if A,B is 2 × 2 then
```

$$\text{return} \begin{bmatrix} a11 \times b11 + a12 \times b21 & a11 \times b12 + a12 \times b22 \\ a21 \times b11 + a22 \times b21 & a21 \times b12 + a22 \times B22 \end{bmatrix}$$

Base condition

```
      end if
```

Recursive Matrix Multiplication

$$C_{11} = M(A_{11}, B_{11}, n/2) + M(A_{12}, B_{21}, n/2) - n^2/4 = 4 \times \frac{n^2}{4}$$

$$C_{12} = M(A_{11}, B_{12}, n/2) + M(A_{12}, B_{22}, n/2) - n^2/4$$

$$C_{12} = M(A_{21}, B_{11}, n/2) + M(A_{22}, B_{21}, n/2) - n^2/4$$

$$C_{12} = M(A_{21}, B_{12}, n/2) + M(A_{22}, B_{22}, n/2) - n^2/4$$

return $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

end function

$$T(n) = 8T(n/2) + n^2$$

Recursive approach ↑ Addition

Master Method ?

$a = 8$

$b = 2$

$n^{\log_2 8} = n^3$

$f(n) - n^2$

$$T(n) = \Theta(n^3)$$