

Characteristics of Algorithm

Algorithm :- An Algo. is a finite set of instructions that if followed accomplishes the task.

Criteria:-

- ① Input :- one/many o/many.
- ② Output :- one/many.
- ③ Finiteness :- It should Terminate.
- ④ Definiteness :- Eng. but not code. // e.g. Add 5 or 6 to x
unambiguous 5/0 Faulty!
- ⑤ Effectiveness :- Should be on pen/pencil & paper.

Steps to Write a Good Algorithm

- ① Indentation includes Block Structures.
- ② The looping constructs have interpretations similar to C & Java.
e.g. `for (i=0; i < n; i++)` \Rightarrow for $i=0$ to n , step size 2
- ③ // symbol indicates comments.
- ④ Multiple Assignments

$i=j=e$ assigns to both i, j the value of expression e .

Variables such as i, j are local to a given procedure. We should not use global variables without explicit statements.

e.g. global i ;

Accessing Specific elements can be done in the same way as we do it traditionally.

For ~~any~~ any attributes no brackets are required.

We organise compound data into objects which are composed of attributes.

We Pass ~~attr~~ parameter by ~~a prob prob~~ through a procedure by value.

A return statement immediately transfers control back to the point of call in the calling procedure.

The keyword error indicates that an error occurred because the conditions were wrong for the procedure to be called.

Algorithm SumofArray (A, n)

```
// finding sum of elements in the array
1 Sum = 0
2 for i = 1 to i = n
3     sum = sum + A[i]
4 print (sum)
```

	# no. of times	cost of each instr.	Total
1	1	C_1	C_1
2	$n+1$	C_2	nC_2
3	n	C_3	nC_3
4	1	C_4	C_4
			$2n + 3$

$O(n)$

$$C_1 \cdot 1 + C_2(n+1) + C_3(n) + C_4 \cdot 1$$

$$C_1 + C_2n + C_2 + C_3n + C_4$$

$$(C_2 + C_3)n + (C_1 + C_2 + C_4)$$

$$\alpha n + \beta$$

$\alpha n + \beta$

$O(n)$

Algorithm - Sum of Matrices (A, B, C)

```
1 for i = 1 to m           A[m][n]   B[m][n]   m+1  
2     for j = 1 to n          m(n+1)  
3         c[i][j] = A[i][j] + B[i][j]    m·n  
4         print C[i][j];            m·n  
5     print (newline).           m.  
m+1 + m(n+1) + mn + mp + m.
```

$$3mn + \cancel{3m} + \cancel{1}$$

$$= O(mn)$$

Write an algorithm for Selection Sort & derive its time complexity

↓
(Best & Worst Case).

Worst Case Scenario.

Algorithm SelectionSort (A, n)

times.

cost

```
1 for i=1 to n-1 .....  
2    imin = i .....  
3     for j=i+1 to n .....  
4         if ( A[imin] > A[j] ) ...  
5             imin = j .....  
6     temp = A[imin] .....  
7     A[imin] = temp A[i] .....  
8     A[i] = temp .....  
.....
```

n.

n-1

$$\sum_{i=2}^n i = \frac{n^2 + n}{2} - 1$$

\sum_j

$$\left\{ \begin{array}{l} \sum_{i=1}^{n-1} i \\ \sum_{i=1}^{n-1} i \end{array} \right\} \frac{n^2 - n}{2}$$

c₁

c₂

c₃

c₄

c₅

c₆

c₇

c₈

$$n c_1 + c_2(n-1) + c_3 \left(\frac{n^2 + n}{2} - 1 \right) + c_4 \left(\frac{n^2 - n}{2} \right) + c_5 \left(\frac{n^2 - n}{2} \right) \\ + c_6(n-1) + c_7(n-1) + c_8(n-1).$$

$$\Rightarrow an^2 + bn + c$$

$\Theta(n^2)$.

In Best Case ignore c₅

As we assume the array imin is always smaller than j ∵ imin = j never occurs.

Worst Case / for while loop consider & shifting + termination.

for CS & CC
or only consider shifting.

Algorithm Insertion Sort (A, n)

# Times	Cost
n	C_1
$n-1$	$C_2(n-1)$
$n-1$	$C_3(n-1)$
$\sum_{i=2}^n i$	$C_4(n-1)$
$\sum_{i=2}^n i$	$C_5 \left(\frac{n(n+1)}{2} - 1 \right)$
$\sum_{i=2}^n i$	$C_6 \left(\frac{n^2+n}{2} - 1 \right)$
$n-1$	$C_7(n-1)$

while ($i \geq 1$ and $A[i] > \text{key}$) .

$A[i+1] = A[i]$.

$$\sum_{i=2}^n i$$

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

$$\frac{n(n+1)}{2} - 1$$

$$\frac{n(n-1)}{2}$$

$$C_1 n + C_2 + C_4 \left(\frac{n^2}{2} + \frac{n-1}{2} \right) + C_5 \left(\frac{n^2}{2} - \frac{n}{2} \right) + C_6 \left(\frac{n^2}{2} - \frac{n}{2} \right)$$

$$C_1 n + C_2 n - C_2 + C_3 n - C_3 + C_4 \frac{n^2}{2} + C_4 \frac{n}{2} - C_4 + C_5 \frac{n^2}{2} - C_5 \frac{n}{2} + C_6 \frac{n^2}{2} - C_6 \frac{n}{2}$$

$$+ C_7 n - C_7$$

$$(C_1 + C_2 + C_3 + C_4 + C_5) n + (C_4 + C_5 + C_6 + C_7)$$

$$\left(\frac{C_4 + (C_5 + C_6)}{2} \right) n^2 + (C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7) n$$

$$+ (C_2 + C_3 + C_4 + C_7)$$

$$an^2 + bn - d$$

$$\Theta(n^2)$$

for Best Case :- Only consider the termination of the while loop.

$$C_4 = n-1$$

$$C_5 \quad 0 \\ C_6 \quad 0$$

$$C_{1,n} + C_2(n-1) + C_3(n-1) + C_4(n-1) + C_5 \cdot 0 + C_6 \cdot 0 + C_7(n-1)$$
$$(C_1 + C_2 + C_3 + C_4 + C_7) n - (C_2 + C_3 + C_4 + C_7)$$

Average Case :-

The while loop will get executed on an average of n^2 times

~~avg case~~ The Average Case Time Complexity will be $O(n^2)$.

Asymptotic Notation:

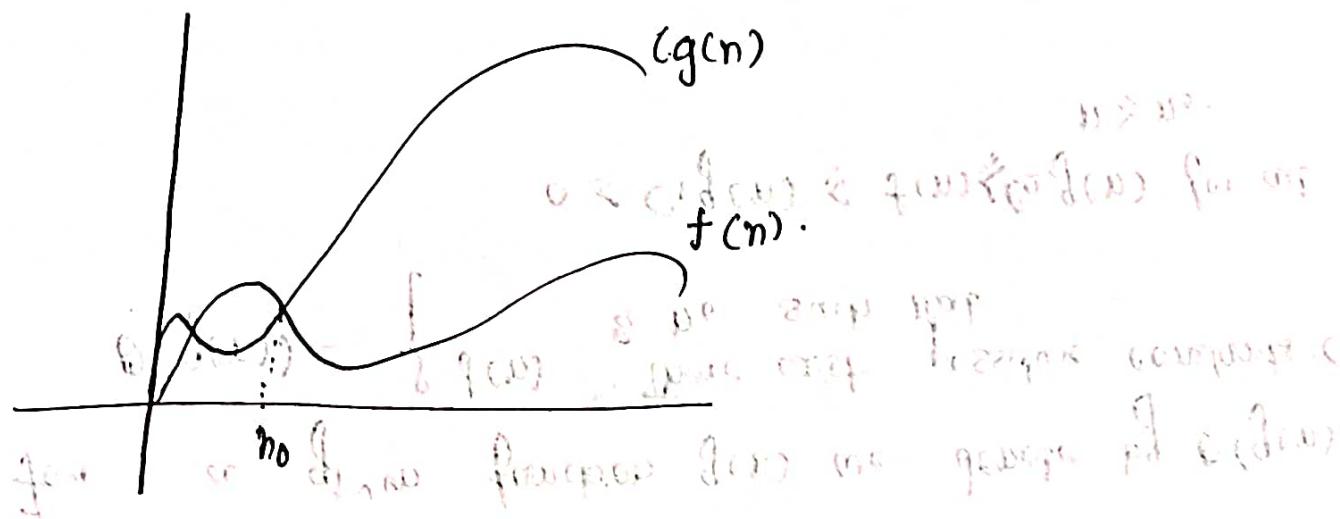


O-notation (Big Oh)

For a given function $g(n)$ we denote by $O(g(n))$

$O(g(n)) = \{ f(n) : \text{There exist positive constants } C \text{ & } n_0 \text{ such that}$

$$0 \leq f(n) \leq g(n) \text{ for all } n \geq n_0.$$



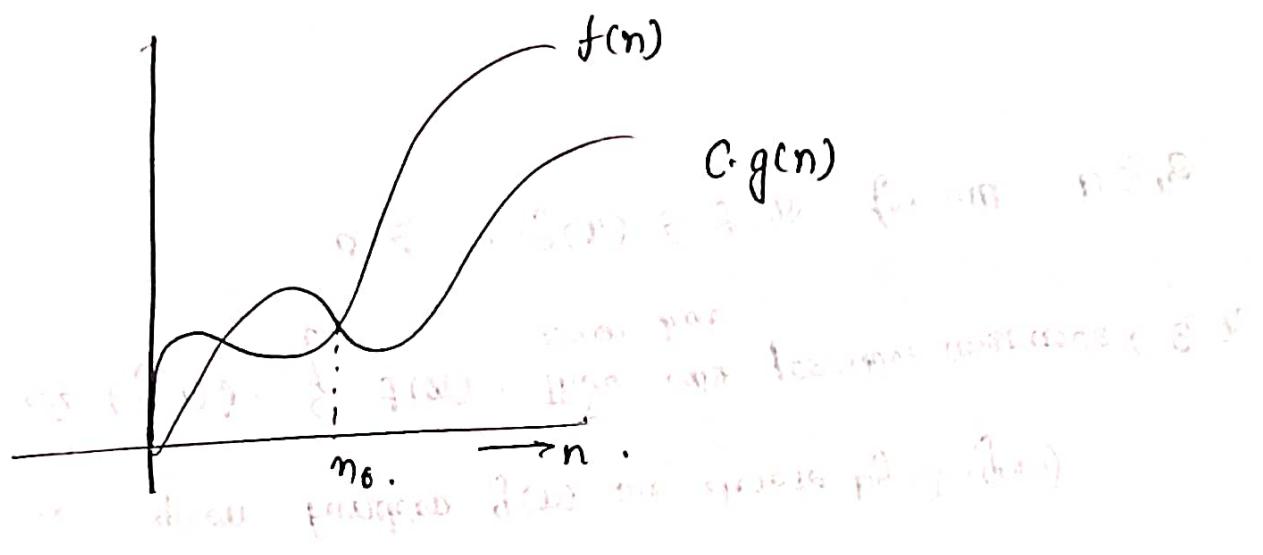
Ω (Omega)

for a given function $g(n)$ we denote by $\Omega(g(n))$

$\Omega(g(n)) = \{ f(n) : \text{There exist positive constants } C \text{ & } n_0 \text{ such that}$

$$0 \leq c.g(n) \leq f(n) \text{ for all } n \geq n_0$$

$g(n)$

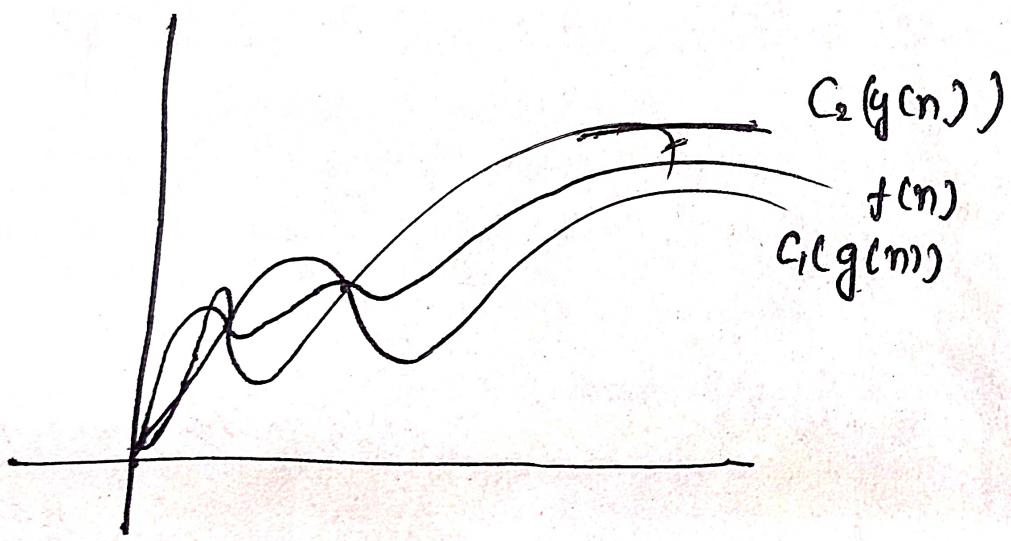


Θ Notation (Theta)

for a given function $g(n)$ we denote by $\Theta(g(n))$

$\Theta(g(n)) = \{ f(n) : \text{There exist positive constants } C \text{ & } n_0 \text{ such that}$

$$0 \leq C_1 g(n) \leq f(n) \leq C_2 g(n) \text{ for all } n \geq n_0.$$



Binary search (A, low, high, x)

if (low == high)

if A[low] == x

return low.

else

return false. (-1)

} ①

else

$$\text{mid} = (\text{low} + \text{high}) / 2$$

if (A[mid] == x)

return mid

else if (A[mid] > x)

BinarySearch (A, low, mid-1, x)

else

BinarySearch (A, mid+1, high, x)

Recurrence Relation.

(1) Masters Theorem.

(2) Recursion Tree.

(3) Substitution Method -

$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

solution. (no. of comparisons made for the terminating condition) i.e ①.

↑
no. of subproblems.

ratio of original problem to subproblems.

recurrence relation of revised question

Recurrence relation of Binary Search.

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

Master's Theorem:

Let $a > 1$ & $b > 1$ be constants; let $f(n)$ be a function & let $T(n)$ be defined on the non-negative integers by the recurrence.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

(i) ~~if~~ if $f(n) = O\left(n^{\log_b a - \epsilon}\right)$; $\epsilon > 0$.

$$\text{then } T(n) = \Theta\left(n^{\log_b a}\right)$$

$$\text{② if } f(n) = \Theta\left(n^{\log_b a}\right) \text{ then } T(n) = \Theta\left(n^{\log_b a} \log n\right)$$

$$\text{③ if } f(n) = \Omega\left(n^{\log_b a + \epsilon}\right); \epsilon > 0 \text{ & } a < b.$$

if $\alpha f\left(\frac{n}{b}\right) \leq c \cdot f(n)$ for some constant $0 < c < 1$
then $T(n) = \Theta(f(n))$

if $\alpha f\left(\frac{n}{b}\right) = \infty$

then $T(n) = \Theta(\log n)$

$$T(n) = T(n/2) + 1$$

$$a = 1 \quad b = 2.$$

$$n^{\log_b^a} = n^{\log_2^1} = n^0 = \underline{\underline{1}}.$$

$f(n) = \underline{\underline{1}}$
The recurrence relation falls in the case II.

$$\therefore T(n) = \Theta(n^{\log_b^a} \log n)$$

$$\boxed{T(n) = \Theta(\log n)}.$$

~~Recursion~~

Tree :

Date _____
Page _____

basically, $2T(n/2)$ means we are DIVIDING the problem into "2" sub-problems where each is half of the original problem.

So,

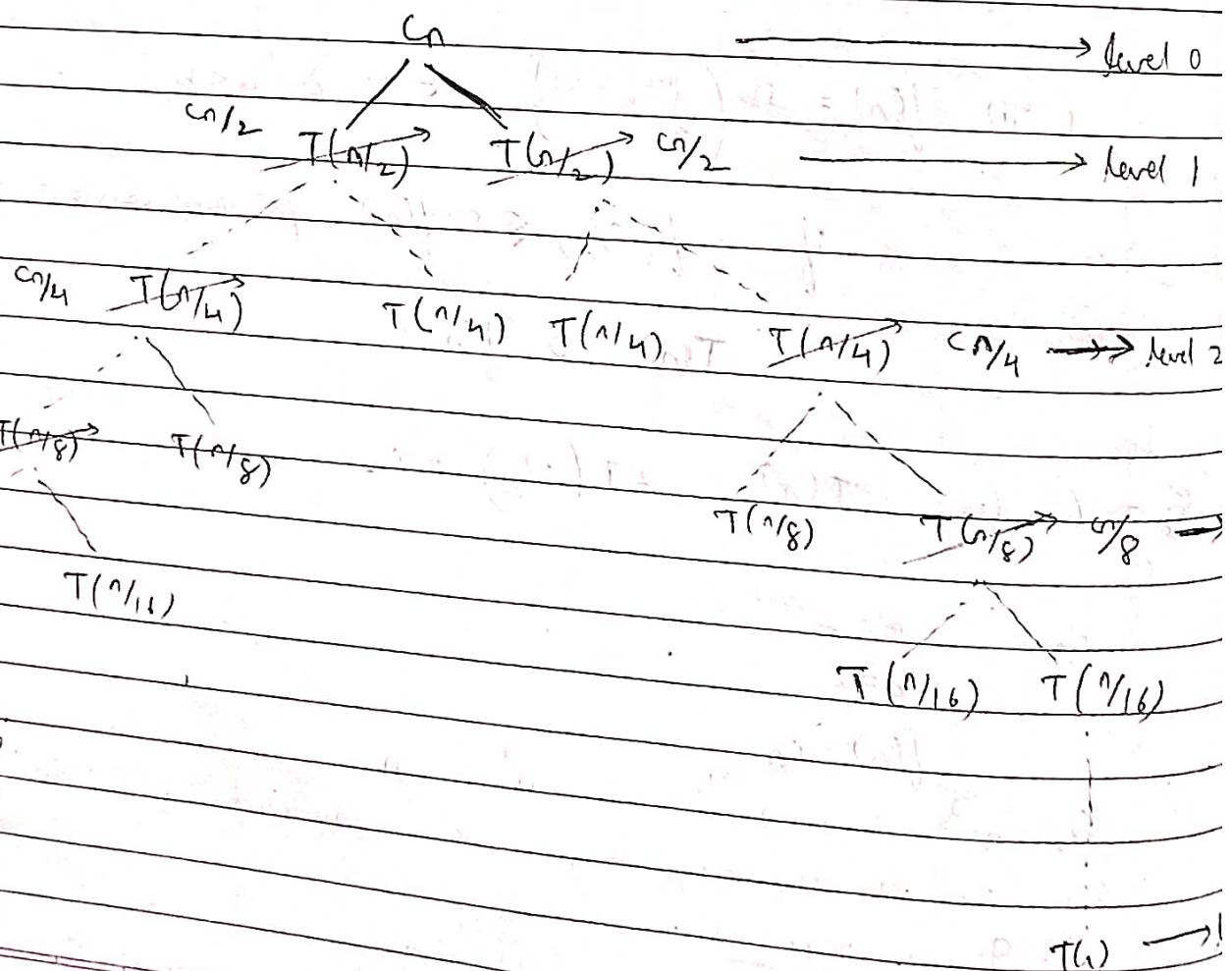
$$T(n) = \left\{ \begin{array}{l} cn \\ + \\ T(n/2) \quad T(n/2) \end{array} \right\}$$

(Combining w/t)

To get $T(n/2)$, substitute $n/2$ in $T(n) = 2T(n/2) + cn$

So, $T(n/2) = 2T(n/4) + cn/2$

So,



Level	No. of terms at each level	cost at each level
-------	----------------------------	--------------------

0	1	c_n
---	---	-------

1	2	c_n
---	---	-------

2	4	c_n
---	---	-------

3	8	c_n
---	---	-------

4	16	c_n
---	----	-------

5	32	c_n
---	----	-------

6	64	c_n
---	----	-------

7	128	c_n
---	-----	-------

8	256	c_n
---	-----	-------

9	512	c_n
---	-----	-------

10	1024	c_n
----	------	-------

11	2048	c_n
----	------	-------

12	4096	c_n
----	------	-------

13	8192	c_n
----	------	-------

14	16384	c_n
----	-------	-------

15	32768	c_n
----	-------	-------

16	65536	c_n
----	-------	-------

17	131072	c_n
----	--------	-------

18	262144	c_n
----	--------	-------

19	524288	c_n
----	--------	-------

20	1048576	c_n
----	---------	-------

21	2097152	c_n
----	---------	-------

22	4194304	c_n
----	---------	-------

23	8388608	c_n
----	---------	-------

24	16777216	c_n
----	----------	-------

25	33554432	c_n
----	----------	-------

26	67108864	c_n
----	----------	-------

27	134217728	c_n
----	-----------	-------

28	268435456	c_n
----	-----------	-------

29	536870912	c_n
----	-----------	-------

30	1073741824	c_n
----	------------	-------

31	2147483648	c_n
----	------------	-------

32	4294967296	c_n
----	------------	-------

33	8589934592	c_n
----	------------	-------

34	17179869184	c_n
----	-------------	-------

35	34359738368	c_n
----	-------------	-------

36	68719476736	c_n
----	-------------	-------

37	137438953472	c_n
----	--------------	-------

38	274877906944	c_n
----	--------------	-------

39	549755813888	c_n
----	--------------	-------

40	1099511627776	c_n
----	---------------	-------

41	2199023255552	c_n
----	---------------	-------

42	4398046511104	c_n
----	---------------	-------

43	8796093022208	c_n
----	---------------	-------

44	17592186044416	c_n
----	----------------	-------

45	35184372088832	c_n
----	----------------	-------

46	70368744177664	c_n
----	----------------	-------

47	140737488355328	c_n
----	-----------------	-------

48	281474976710656	c_n
----	-----------------	-------

49	562949953421312	c_n
----	-----------------	-------

50	1125899906842624	c_n
----	------------------	-------

51	2251799813685248	c_n
----	------------------	-------

52	4503599627370496	c_n
----	------------------	-------

53	9007199254740992	c_n
----	------------------	-------

54	18014398509481984	c_n
----	-------------------	-------

55	36028797018963968	c_n
----	-------------------	-------

56	72057594037927936	c_n
----	-------------------	-------

57	144115188075855872	c_n
----	--------------------	-------

58	288230376151711744	c_n
----	--------------------	-------

59	576460752303423488	c_n
----	--------------------	-------

60	1152921504606846976	c_n
----	---------------------	-------

61	2305843009213693952	c_n
----	---------------------	-------

62	4611686018427387904	c_n
----	---------------------	-------

63	9223372036854775808	c_n
----	---------------------	-------

64	18446744073709551616	c_n
----	----------------------	-------

65	36893488147419103232	c_n
----	----------------------	-------

66	73786976294838206464	c_n
----	----------------------	-------

67	147573952589676412928	c_n
----	-----------------------	-------

68	295147905179352825856	c_n
----	-----------------------	-------

69	590295810358705651712	c_n
----	-----------------------	-------

70	1180591620717411303424	c_n
----	------------------------	-------

71	2361183241434822606848	c_n
----	------------------------	-------

72	4722366482869645213696	c_n
----	------------------------	-------

73	9444732965739290427392	c_n
----	------------------------	-------

74	18889465931478580854784	c_n
----	-------------------------	-------

75	37778931862957161699568	c_n
----	-------------------------	-------

76	75557863725914323399136	c_n
----	-------------------------	-------

77	15111572745828646698272	c_n
----	-------------------------	-------

78	30223145491657293396544	c_n
----	-------------------------	-------

79	60446290983314586793088	c_n
----	-------------------------	-------

80	120892581966629173586176	c_n
----	--------------------------	-------

81	241785163933258347172352	c_n
----	--------------------------	-------

82	483570327866516694344704	c_n
----	--------------------------	-------

83	967140655733033388689408	c_n
----	--------------------------	-------

84	1934281311466066777378816	c_n
----	---------------------------	-------

85	3868562622932133554757632	c_n
----	---------------------------	-------

86	7737125245864267109515264	c_n
----	---------------------------	-------

87	15474250491728534219030528	c_n
----	----------------------------	-------

88	30948500983457068438061056	c_n
----	----------------------------	-------

89	61897001966914136876122112	c_n
----	----------------------------	-------

90	12379400393382827375224424	c_n
----	----------------------------	-------

91	24758800786765654750448848	c_n
----	----------------------------	-------

92	49517601573531309500897696	c_n
----	----------------------------	-------

93	99035203147062619001795392	c_n
----	----------------------------	-------

94	198070406294125238003590784	c_n
----	-----------------------------	-------

95	396140812588250476007181568	c_n
----	-----------------------------	-------

96	792281625176500952014363136	c_n
----	-----------------------------	-------

97	1584563250353001840287326272	c_n
----	------------------------------	-------

98	3169126500706003680574652544	c_n
----	------------------------------	-------

99	6338253001412007361149305088	c_n
----	------------------------------	-------

100	12676506002824014722296600176	c_n
-----	-------------------------------	-------

101	25353012005648029444593200352	c_n
-----	-------------------------------	-------

102	50706024011296058889186400704	c_n
-----	-------------------------------	-------

103	101412048022592117778372801408	c_n
-----	--------------------------------	-------

104	202824096045184235556745602816	c_n
-----	--------------------------------	-------

105	405648192090368471113491205632	c_n
-----	--------------------------------	-------

106	811296384180736942226982411264	c_n
-----	--------------------------------	-------

107	1622592768361473884453964822528	c_n
-----	---------------------------------	-------

108	3245185536722947768907929645056	c_n
-----	---------------------------------	-------

109	6490371073445895537815859290112	c_n
-----	---------------------------------	-------

110	1298074214689179107563171858024	c_n
-----	---------------------------------	-------

111	2596148429378358215126343716048	c_n
-----	---------------------------------	-------

112	5192296858756716430252687432096	c_n
-----	---------------------------------	-------

113	1038459371751343286050537486192	c_n
-----	---------------------------------	-------

114	2076918743502686572101074972384	c_n
-----	---------------------------------	-------

115	4153837487005373144202149944768	c_n
-----	---------------------------------	-------

116	8307674974010746288404299889536	c_n
-----	---------------------------------	-------

117	1661534954802149257680859977908	c_n
-----	---------------------------------	-------

118	3323069909604298515361719955816	c_n
-----	---------------------------------	-------

119	6646139819208597030723439911632	c_n
-----	---------------------------------	-------

120	1329227963841719406144687982324	c_n
-----	---------------------------------	-------

121	2658455927683438812289375964648	c_n
-----	---------------------------------	-------

122	5316911855366877624578751929296	c_n
-----	---------------------------------	-------

123	1063382371073375524915750385892	c_n
-----	---------------------------------	-------

124	21267647421467510498315007717
-----	-------------------------------

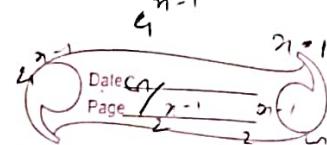
$$\frac{cn}{2^n}$$

$$T(1)$$

$$2^n \cdot cn \cdot T(1)$$

$$2^{n-1} = \frac{cn}{2} \cdot T(1)$$

$$cn + 2cn \left[1 + 2 + 3 + 4 + \dots + 2^{n-1} \right]$$



$$(S_1) \quad T(n) = 4T\left(\frac{n}{2}\right) + cn$$

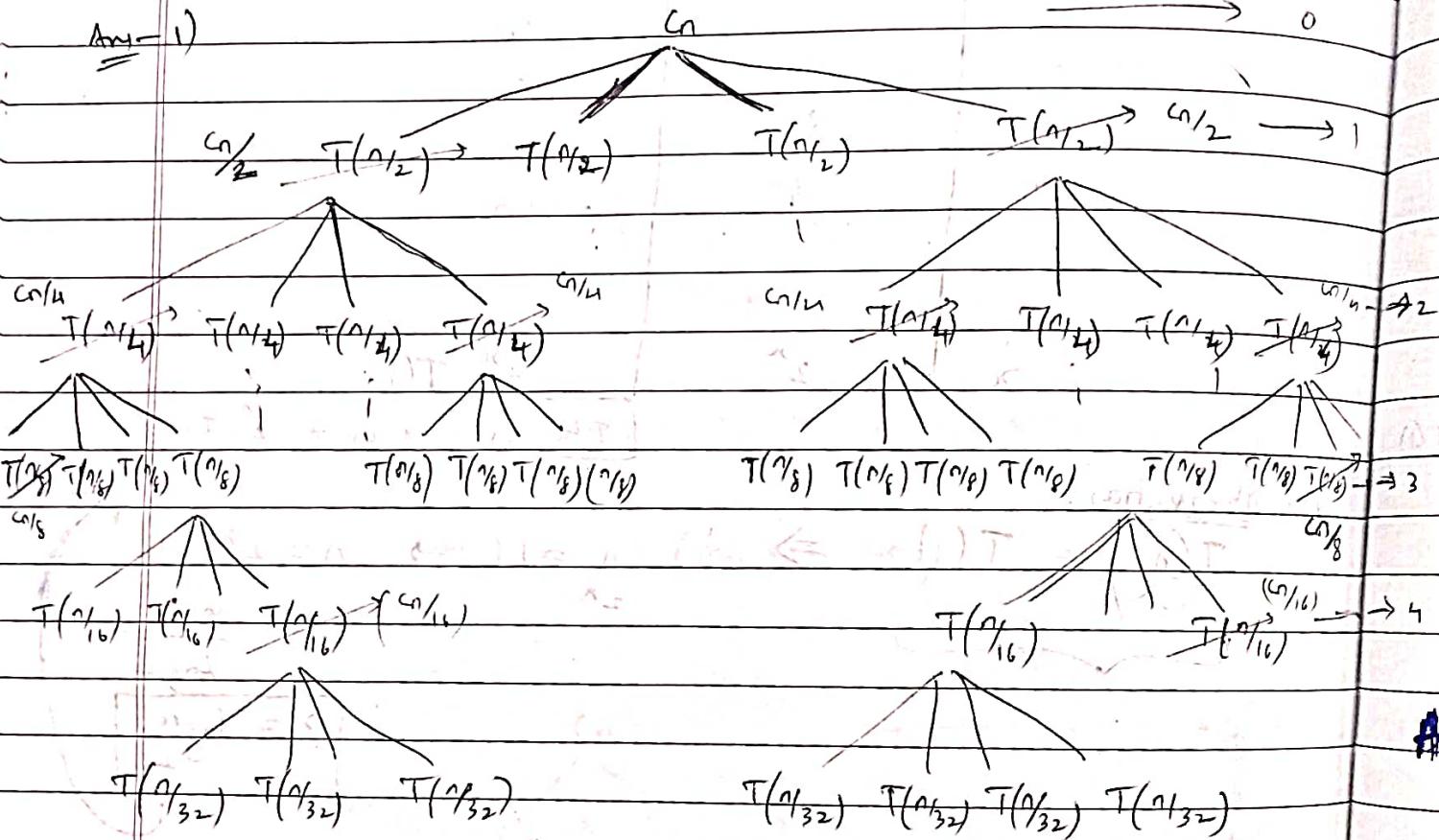
$$(S_2) \quad T(n) = 2T\left(\frac{n}{2}\right) + cn^2$$

$$(S_3) \quad T(n) = 3T\left(\frac{n}{3}\right) + cn^2$$

Ans - 1)

cn

0



Level

No. of terms at each level

cost

0

cn

1

~~2~~ $2cn$

2

~~4~~ $4cn$

3

~~8~~ $8cn$

4

~~16~~ $16cn$

5

:

x

4^x

$$4^n T(1) \sum_{i=0}^{x-1} 2^i cn$$

$$\text{Total cost} = (n^2 - 3n + 3) cn + 4^n T(1)$$

$$T\left(\frac{n}{2^n}\right) = T(1) \Rightarrow \frac{n}{2^n} = 1 \Rightarrow n = 2^n$$

$$n = \frac{\log n}{\log 2} = \boxed{\log_n}$$

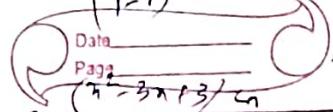
$$S = 2cn(1 + L + S + \dots)$$

$$S = 2cn[1 + 2 + 3 + \dots]$$

$$S = 2cn \frac{(x-2)(n-1)}{2}$$

$$\frac{a(1-r)}{1-r}$$

$$\frac{1(2-1)}{1}$$



$$\text{So, Total cost} = 4^x \cdot T(1) + \sum_{i=2}^{x-1} 2^i \cdot cn$$

$$= 4^{\log_2^n} \cdot T(1) + cn(2^{x-1})$$

$$= n^2 \cdot T(1) + cn\left(\frac{\log_2^n}{2} - 1\right)$$

$$= n^2 \cdot T(1) + cn(n-1)$$

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

$$= n^2 \cdot (1) + \cancel{cn^2 - cn}$$

$$\cancel{cn^2} - cn$$

$$\Rightarrow O(n^2)$$

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

$$\frac{cn^2}{4} \quad T(n/2) \rightarrow \frac{cn^2}{4} \quad T(n/2) \rightarrow \frac{cn^2}{4} \rightarrow ①$$

$$\frac{cn^2}{16} \quad T(n/4) \rightarrow \frac{cn^2}{16} \quad T(n/4) \rightarrow \frac{cn^2}{16} \rightarrow ②$$

$$\frac{cn^2}{64} \quad T(n/8) \rightarrow \frac{cn^2}{64} \quad T(n/8) \rightarrow \frac{cn^2}{64} \rightarrow ③$$

$$\frac{cn^2}{256} \quad T(n/16) \rightarrow \frac{cn^2}{256} \quad T(n/16) \rightarrow \frac{cn^2}{256} \rightarrow ④$$

$$T(n/32) \quad T(n/32) \quad T(n/32) \quad T(n/32)$$

$$T(1) \rightarrow ⑤$$

Level	No. of terms at each level	cost
0	1	cn^2
1	2	$cn^2/2$
2	4	$cn^2/4$
3	8	$cn^2/8$
4	16	$cn^2/16$
\vdots	2^n	$2^n \cdot T(1)$
n	2^n	Total = $\sum_{i=0}^{n-1} \frac{cn^2}{2^i} + 2^n \cdot T(1)$

$$T\left(\frac{n}{2^n}\right) = T(1) \Rightarrow \frac{n}{2^n} = 1 \Rightarrow n = \log_2 2^n$$

$$\text{Total cost} = 2^{\log_2 n} \cdot T(1) + \binom{n}{2} \sum_{i=0}^{n-1} \binom{i}{2}$$

$$= n \cdot T(1) + (cn^2) \frac{(2^n - 1)}{(1/2) \cdot (2^n)}$$

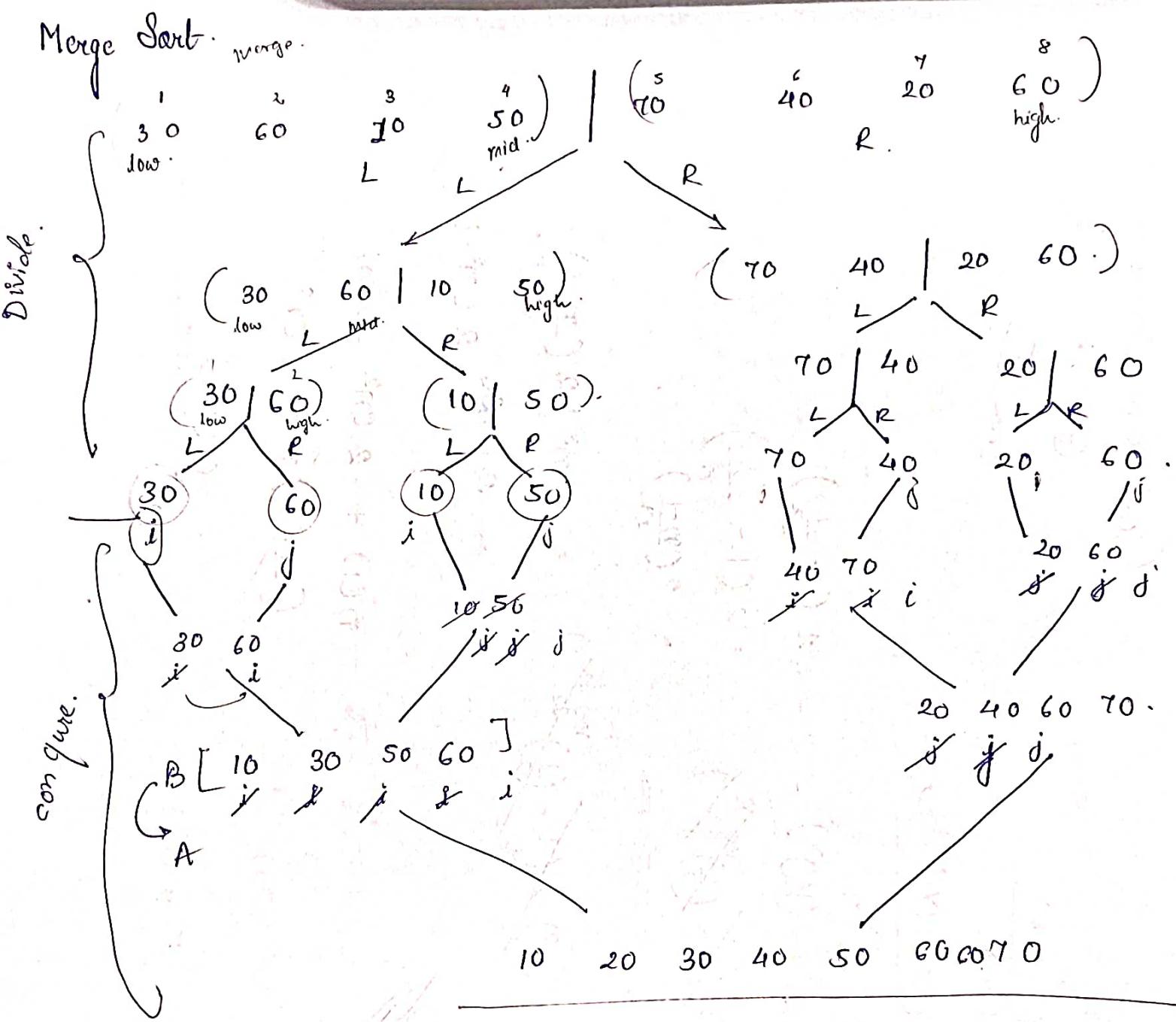
$$= n \cdot (1) + 2cn^2 \frac{(2^n - 1)}{(2^n)}$$

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

$$= cn^2 + 2cn^2 - 2cn$$

$$= 2cn^2 - n(2n-1)$$

$$= O(n^2)$$



Merge Sort (A, low, high)

{ if ($low < high$)

{ mid = ($low + high$) / 2 // left

Merge Sort (A, low, mid) // right // left

Merge Sort (A, mid+1, high) // Right

Merge (A, mid+1, high) combine.

Merge (A, low, mid, high)

}

Merge (A, low, mid, high)

{

i = low

j = mid+1

k = low // temporary array

while (i ≤ mid & j ≤ high)

// B is a temporary array.

if A[i] ≤ A[j]

B[k] = A[i]

i++

else

B[k] = A[j].

j++

if (i > mid) k++ }

if (j > mid)

while (j ≤ high)

B[k] = A[j]

j++

k++

~~if $j > \text{high}$~~
~~if $i > \text{high}$~~ if i

while $i \leq \text{mid}$

$$B[k] = A[i]$$

$i++$

$k++$

for $i = \text{low}$ to high .
 $A[i] = B[i]$

}

Merge Sort time Complexity.

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

$a = 2$

$b = 2$

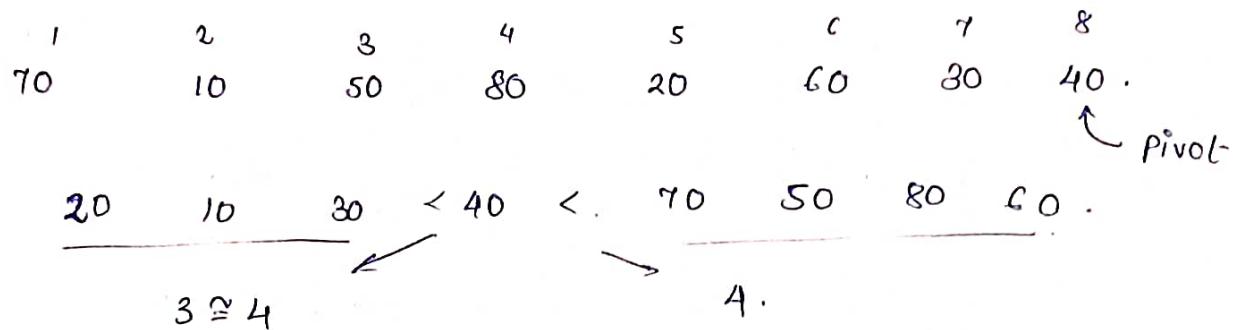
$$f(n) = Cn$$

$$n \log_2 2 = n$$

$$f(n) = Cn$$

$$T(n) = O(n \log n)$$

case II



$$T(n) = 2T\left(\frac{n}{2}\right) + \mathcal{O} \quad \text{Partition.}$$

Quick Sort: $(A, low, high)$

if $low < high$

$p = \text{partition}(A, low, high)$

Quicksort $(A, low, p-1)$

Quicksort $(A, p+1, high)$

Partition $(A, low, high)$

$\text{pivot} = A[high]$

$i = low$

$j = low$

while ($i < high$)

{ if $A[i] < \text{pivot}$

swap $(A[i], A[j])$

$j++$

$i++$

swap $(A[j], A[high])$

swap (int a, int b)

{ int temp;

temp = a

a = b

b = temp.

∴ Best Case Time Recurrence Relation. (Average Case).

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

$$\underline{O(Cn/2)} \quad O(Cn \log n).$$

For worst case:— \longrightarrow The pivot element is at the end. (correctly placed)

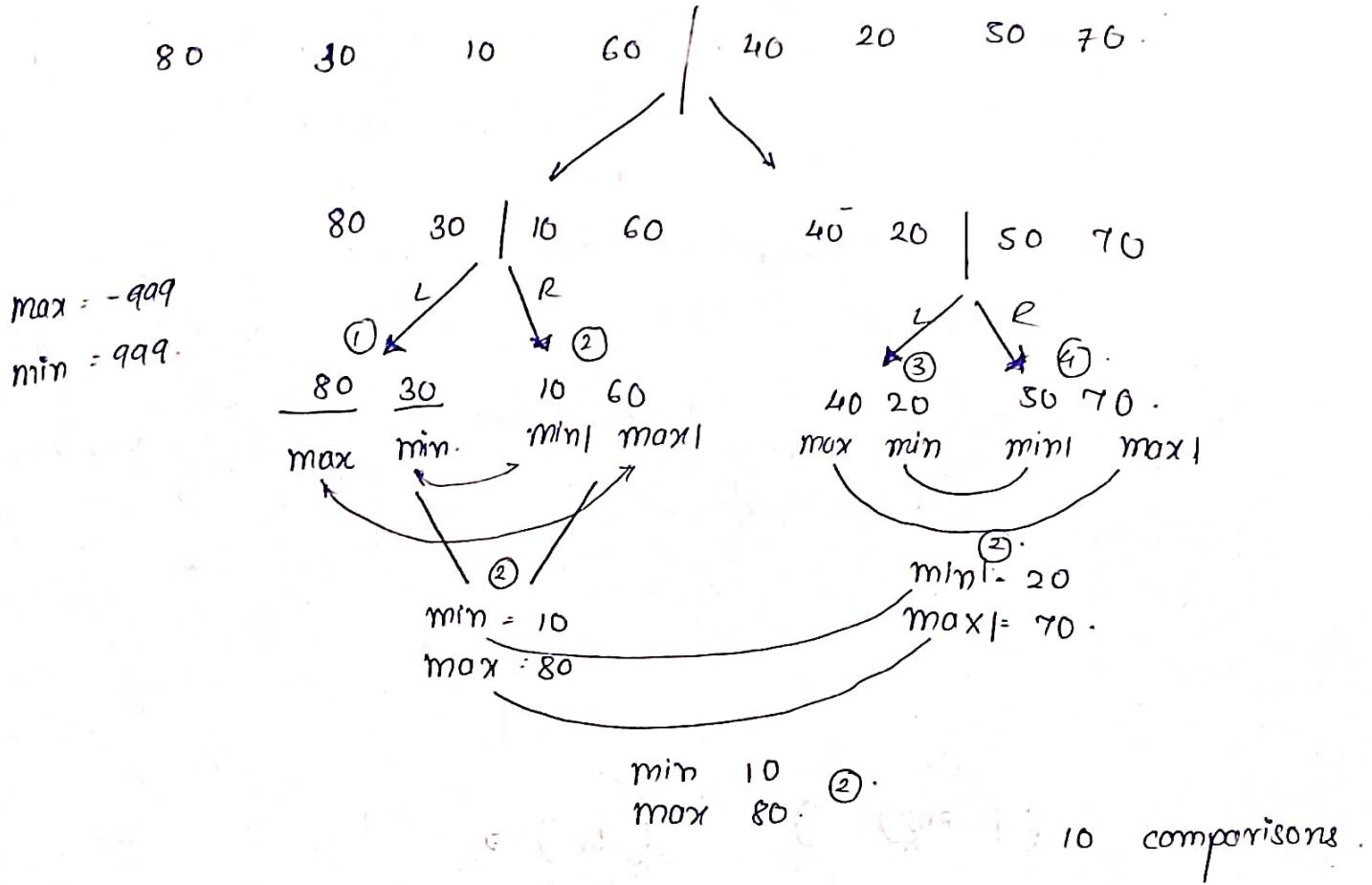
$$T(n) = T(n-1) + Cn.$$

Eg 10 20 30 40 50 | 60.

$$T(60) = T(55) + Cn$$

$$\underline{O(Cn^2)}.$$

OR
Array might be sorted
in Descending
Order.



Time Complexity $O(n)$

Algorithm ~~Maximin~~ ($A, i, j, \text{max}, \text{min}$)

if $i == j$
 $\text{max} = \text{min} = A[i]$ // For only one element.

else if $i == j - 1$
 if $A[i] < A[j]$
 $\text{min} = A[i]$ // for two elements.
 $\text{max} = A[j]$

else
 if $i < j$
 $\text{max} = A[i]$
 $\text{min} = A[j]$.

else

$$\text{mid} = (i+j)/2$$

$\text{Maxmin}(A, i, \text{mid}, \text{max}, \text{min})$

$\text{Maxmin}(A, \frac{i+\text{mid}}{2}, j, \text{maxl}, \text{minl})$

if $\text{maxl} > \text{max}$.

$$\text{max} = \text{maxl}$$

if $\text{minl} < \text{min}$.

$$\text{min} = \text{minl}$$

Ambiguous
in Practical
Code.

Time Complexity

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

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

$$f(n) = 2^n$$

$$\therefore \Theta(n) \quad (\text{Case 1}).$$