ADA Lab

1) Selection Sort (no sleep)

n	t	≈t
10000	0.151890	0.2
20000	0.572826	0.6
30000	1.267554	1.3
40000	2.263620	2.3
50000	3.564726	3.6
60000	5.077233	5.1
70000	6.969399	7.0
80000	9.228300	9.2
90000	11.428874	11.4
100000	14.023419	14

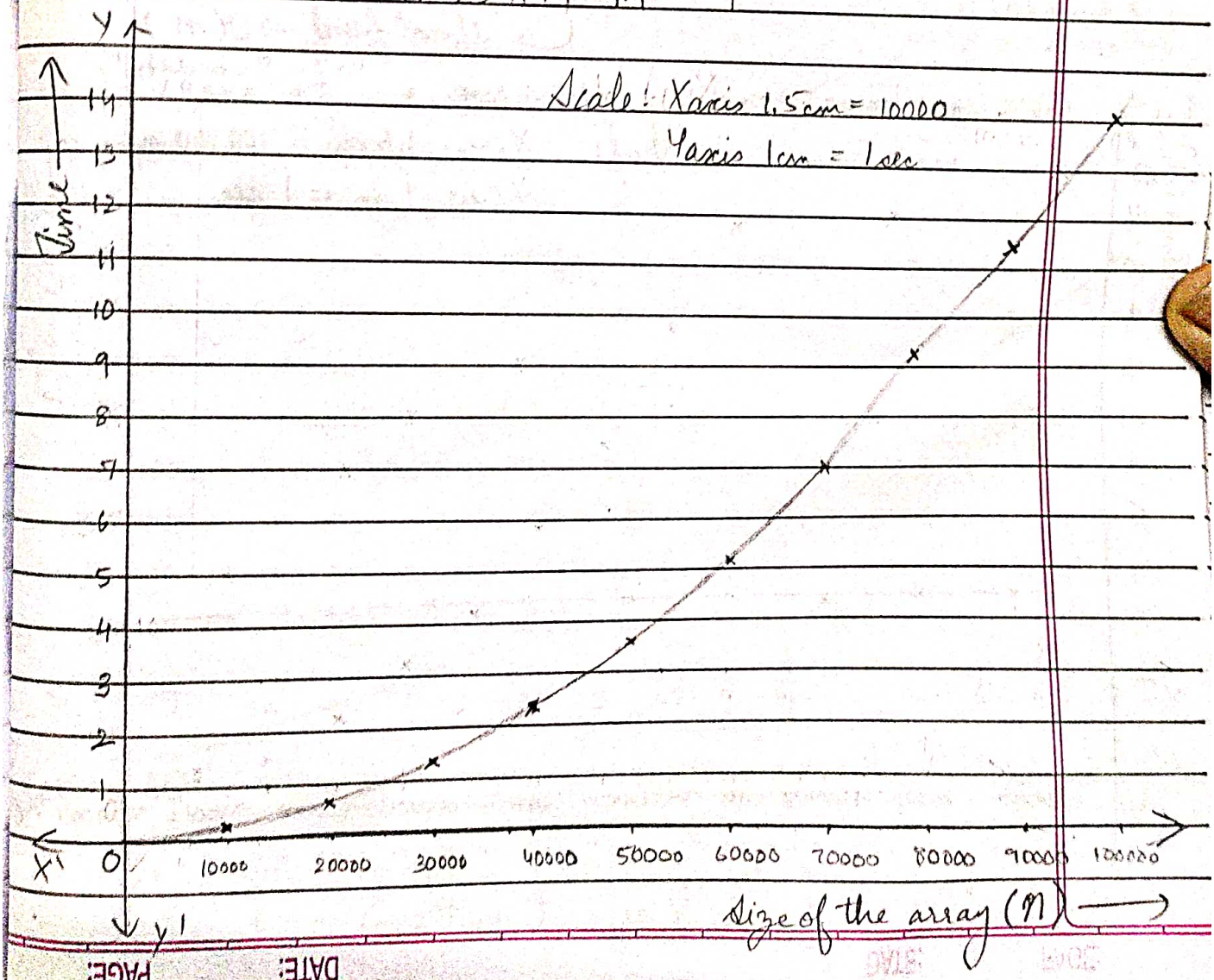
Time:

Case Comparisons, Swaps

Best	$O(n^2)$	$O(1)$
Avg	$O(n^2)$	$O(n)$
Worst	$O(n^2)$	$O(n)$

Space:

Worst $O(1)$ auxiliary.



Scale: Xaxis 1.5cm = 10000
Yaxis 1cm = 1sec

PAGE:
DATE:

2) Brute force string match (sleep(0.1) in inner while loop)

(n) = nt = 1,00,00,000 (1 crore)

(m)

np	t	≈t
100000	3.495001	3.5
200000	3.370467	3.4
300000	3.338601	3.3
400000	3.366675	3.4
500000	3.294913	3.3
600000	3.267052	3.3
700000	3.254268	3.3
800000	3.186548	3.2
900000	3.176283	3.2
1000000	3.096119	3.1

If we take nt v/s time then graph will increase maybe for np v/s time it'll decrease.

Worst → O(nm)

Avg → O(n+m) in the beg.

Best → if pattern found → O(m)

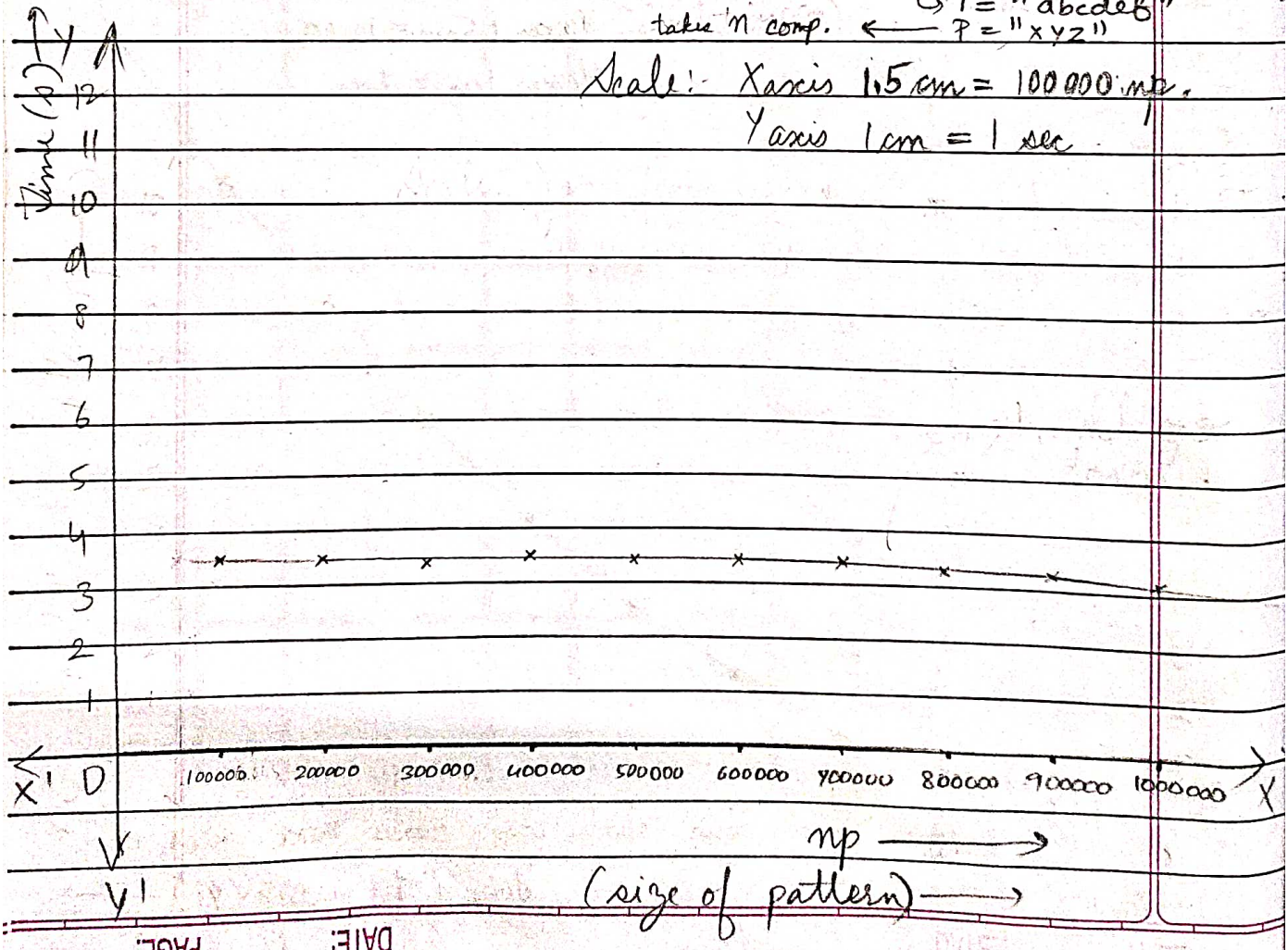
↳ if not found → O(n)

↳ T = "abcdefb"

take 'n' comp. ← P = "xyz"

Scale: X axis 1.5 cm = 100000 np.

Y axis 1 cm = 1 sec



PAGE: DATE:

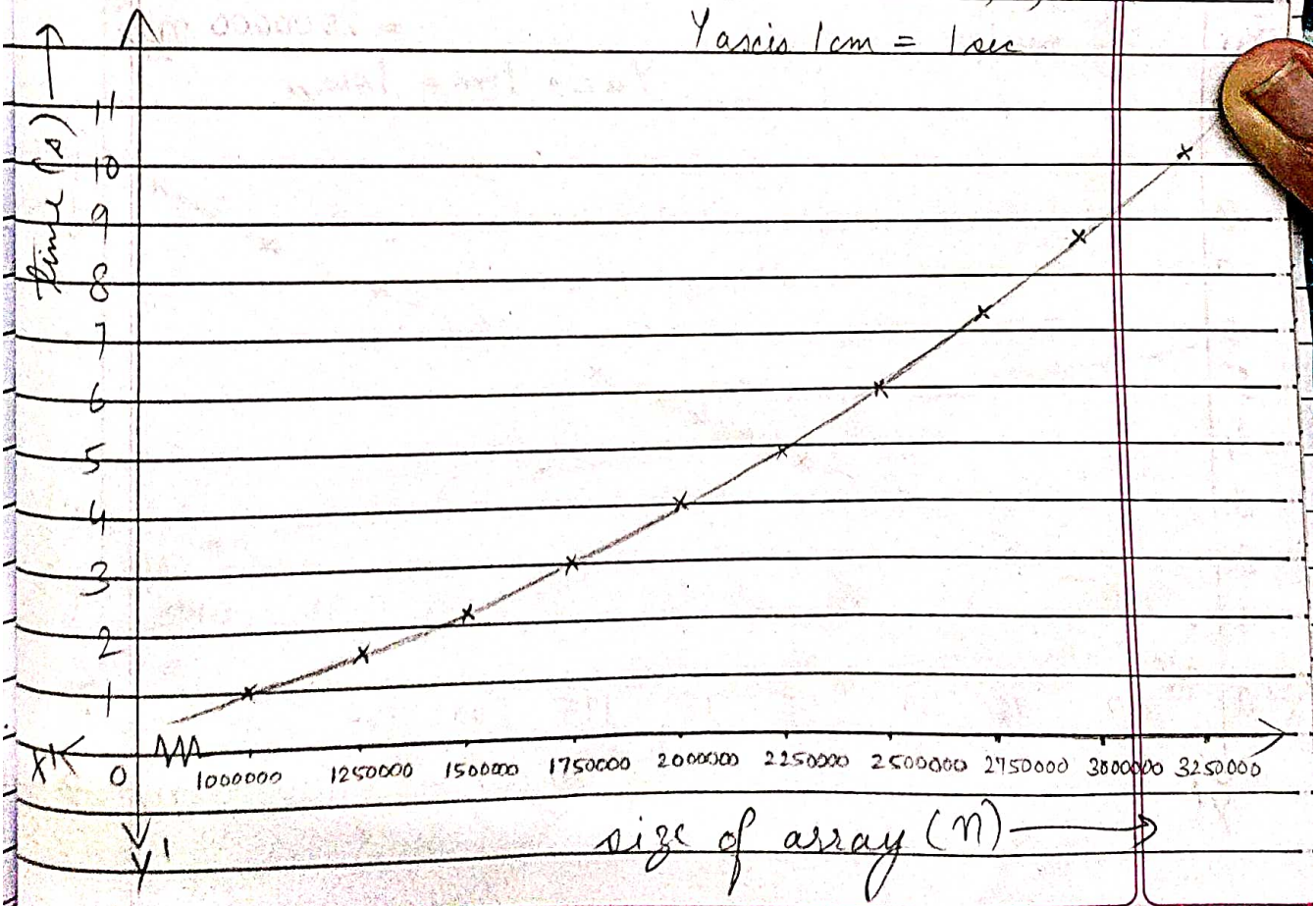
3) Quicksort (use sleep) — Case performance

n	t	≈t
10 00 000	1.015733	1.0
12 50 000	1.558347	1.6
15 00 000	2.209464	2.2
17 50 000	2.993569	3.0
20 00 000	3.912658	4.0
22 50 000	4.899253	4.9
25 00 000	6.023231	6.0
27 50 000	7.281868	7.3
30 00 000	8.705107	8.7
32 50 000	10,091069	10.1

Best O(nlogn)
Avg O(nlogn)
Worst $O(n^2)$

here we did sand() % 1000 so many copies of same no., (use quicksort 3 to avoid this), that's why there's a difference in the no. of ele quick & merge sort in next page.

Scale: Xaxis 1.5cm = 2,50,000 n
Yaxis 1cm = 1sec

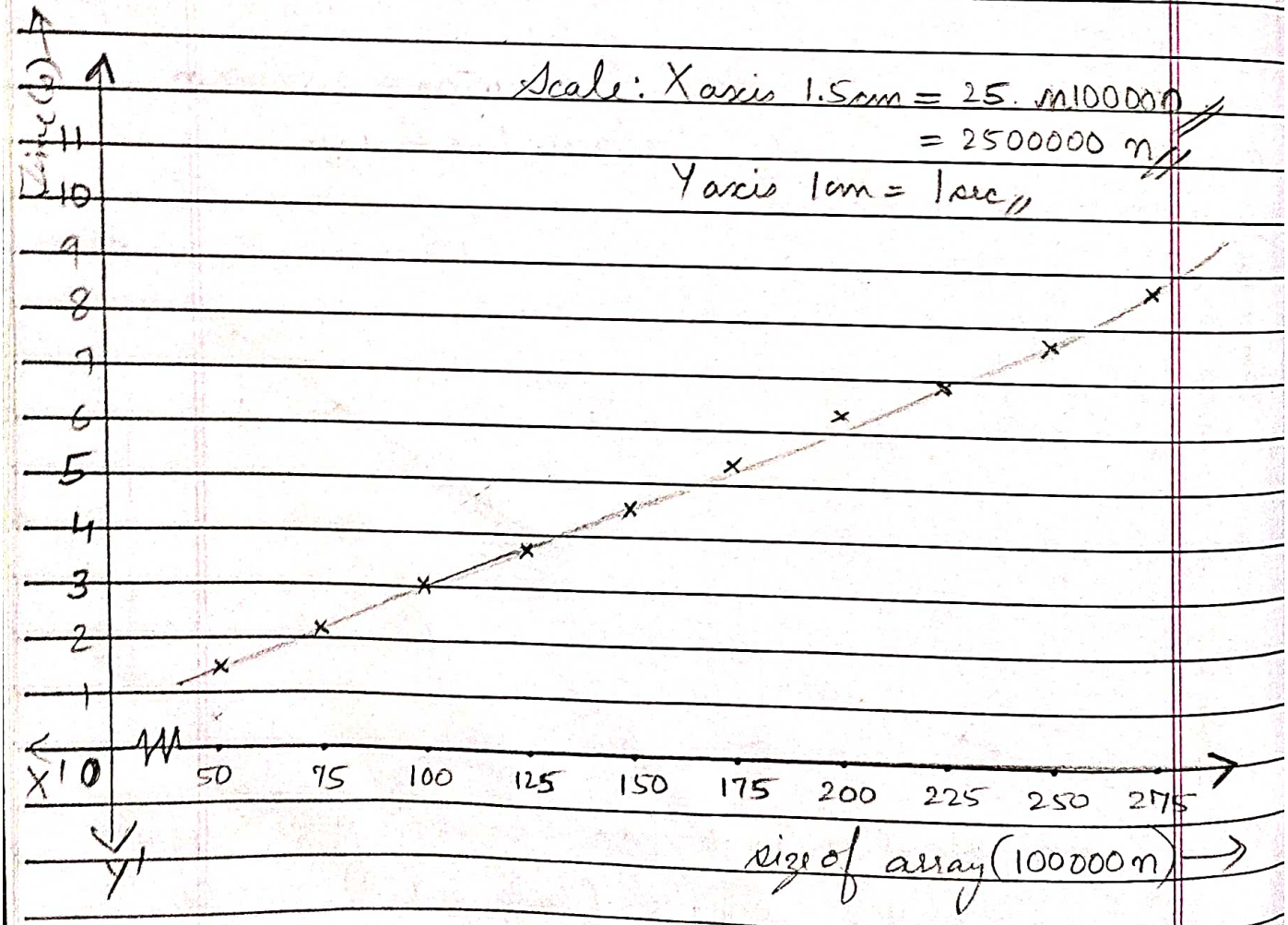


PAGE: DATE:

4) Mergesort (no sleep)

Case Performance
B, A, W O(n log n)

n	t	≈t	
50 00 000	1.481491	1.5	
75 00 000	2.211617	2.1	O(n) Space Comp.
100 00 000	3.003953	3.0	
125 00 000	3.718443	3.7	
150 00 000	4.497951	4.5	
175 00000	5.312698	5.3	
200 00 000	6.211526	6.2	
225 00 000	6.879394	6.9	
250 00 000	7.584053	7.6	
275 00 000	8.576077	8.6	

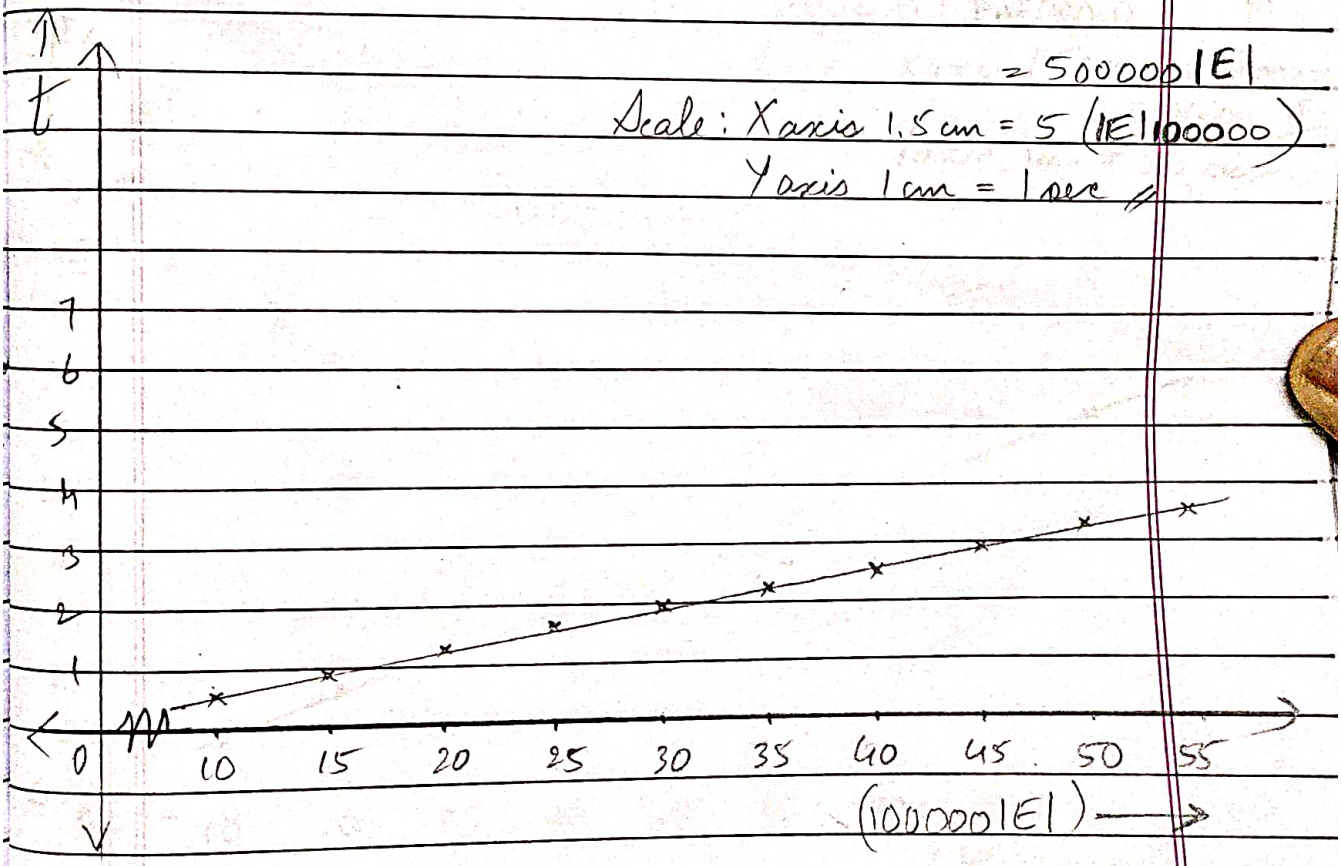


DATE: PAGE:

5) Kruskal's Algorithm (no sleep)
$|V| = 1000000$ (constant)

$\|E\|$	t	$\simeq t$
1000000	0.607917	0.6
1500000	0.932571	0.9
2000000	1.260241	1.3
2500000	1.575254	1.6
3000000	1.871444	1.9
3500000	2.236791	2.2
4000000	2.523163	2.5
4500000	2.865025	2.9
5000000	3.216166	3.2
5500000	3.511158	3.5

$O(E \log E)$ // $O(E \log V)$ //



ElogE / ElogV.

DATE: PAGE:

6) BFS (no sleep)

$|V| = 10000$ (i.e rand()%10000)

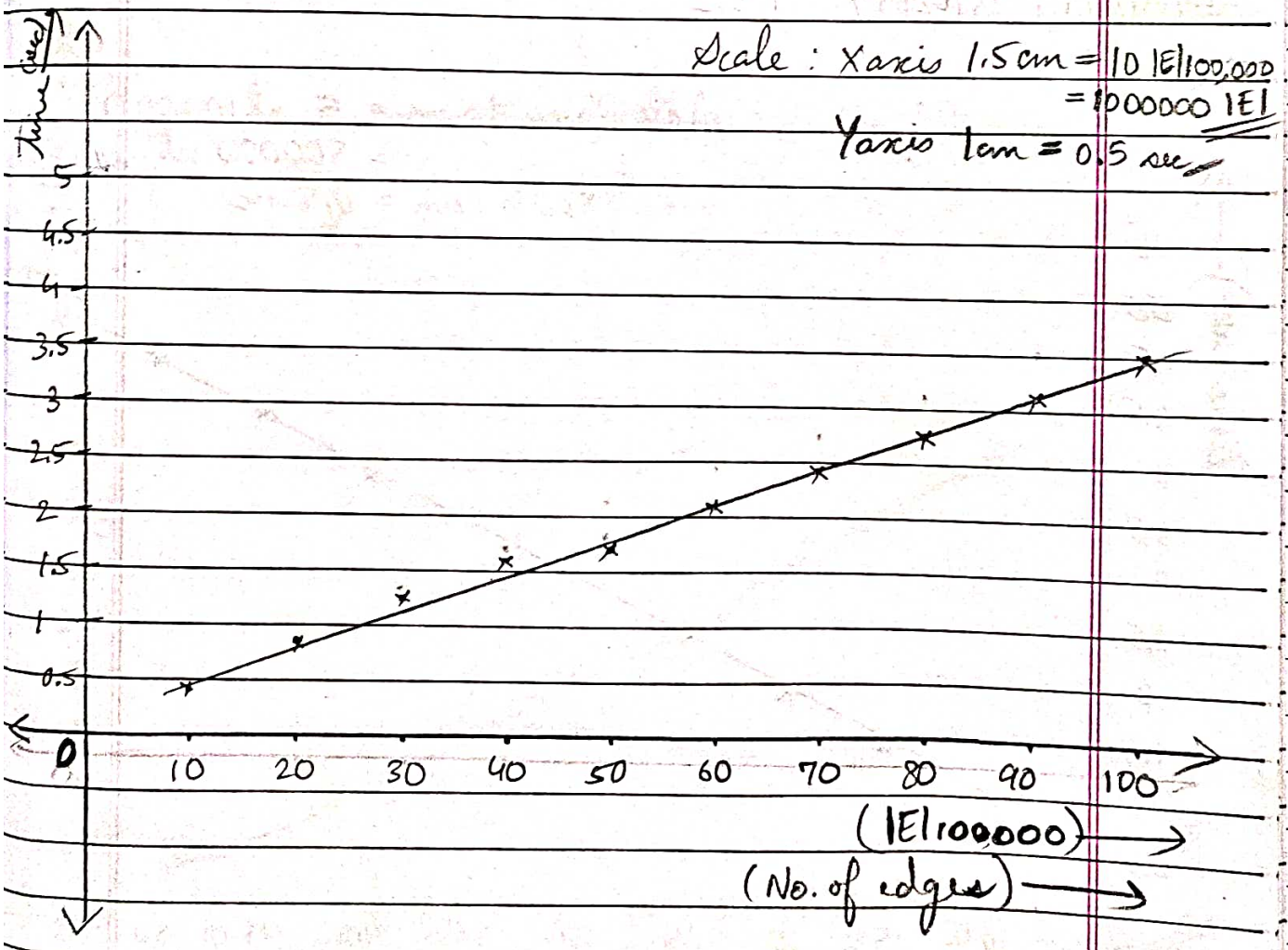
$\|E\|$	t	≈t
1000000	0.432915	0.4
2000000	0.825106	0.8
3000000	1.268006	1.3
4000000	1.663508	1.6
5000000	1.721398	1.7
6000000	2.059086	2.1
7000000	2.377281	2.4
8000000	2.733710	2.7
9000000	3.114764	3.1
10000000	3.505273	3.5

☆☆☆

adjacency list → O(V+E)

adjacency matrix → $O(V^2)$

→ O(V+E) we are doing here.



Scale : Xaxis 1.5cm = 10 |E|100,000
= 1000000 |E|

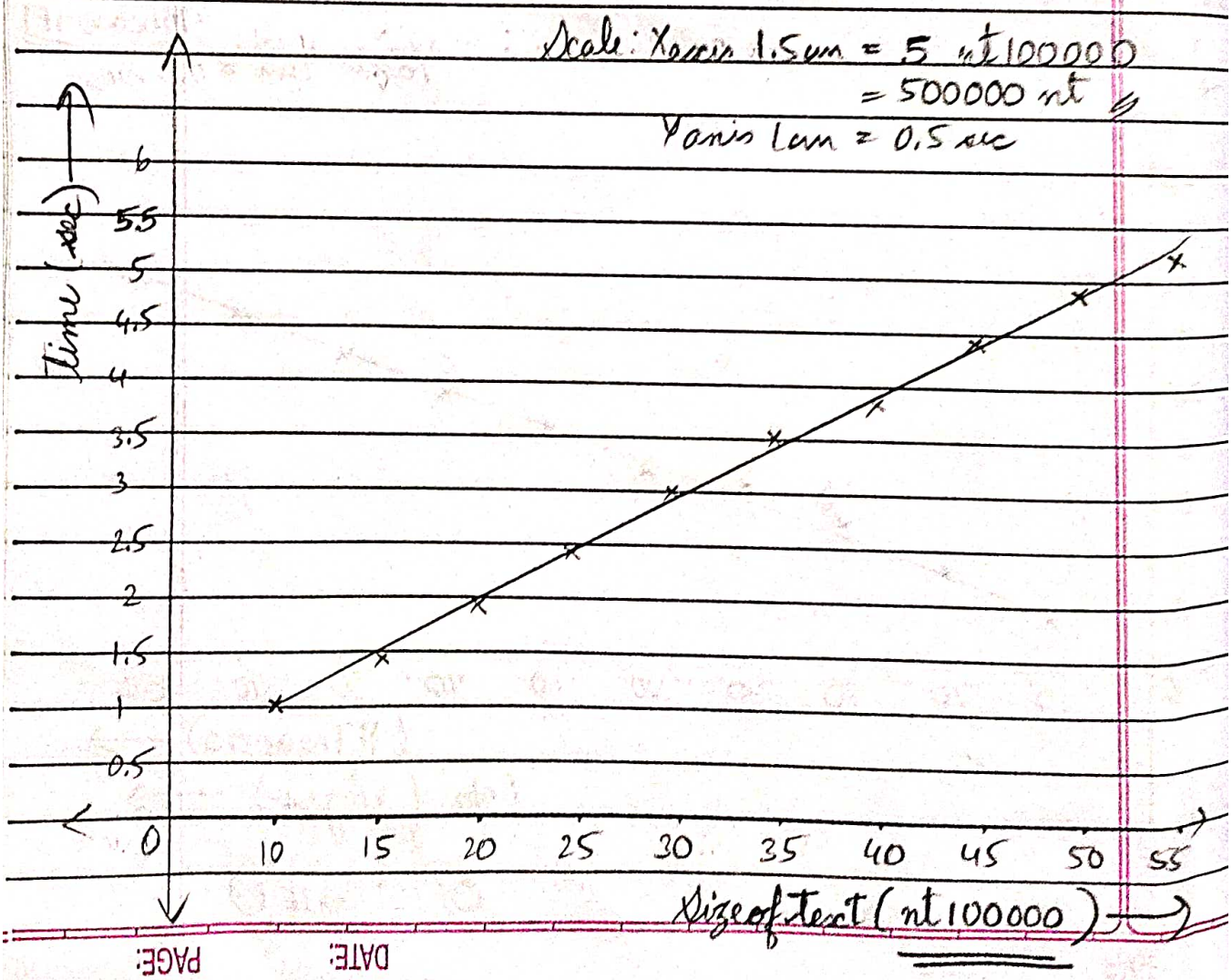
Yaxis 1cm = 0.5 sec

DATE: PAGE:

7] Hasopaal (sleep 0.1)

inp = 20 (fixed)

nt	t	≃t	
1000000	0.954643	1.0	Best case → sub-linear.
1500000	1.417405	1.4	Average case → O(n)
2000000	1.901902	1.9	Worst case → O(nlogn)
2500000	2.413174	2.4	
3000000	2.933515	3.0	
3500000	3.502000	3.5	
4000000	3.820820	3.8	
4500000	4.407284	4.4	
5000000	4.857511	4.8	
5500000	5.162197	5.2	



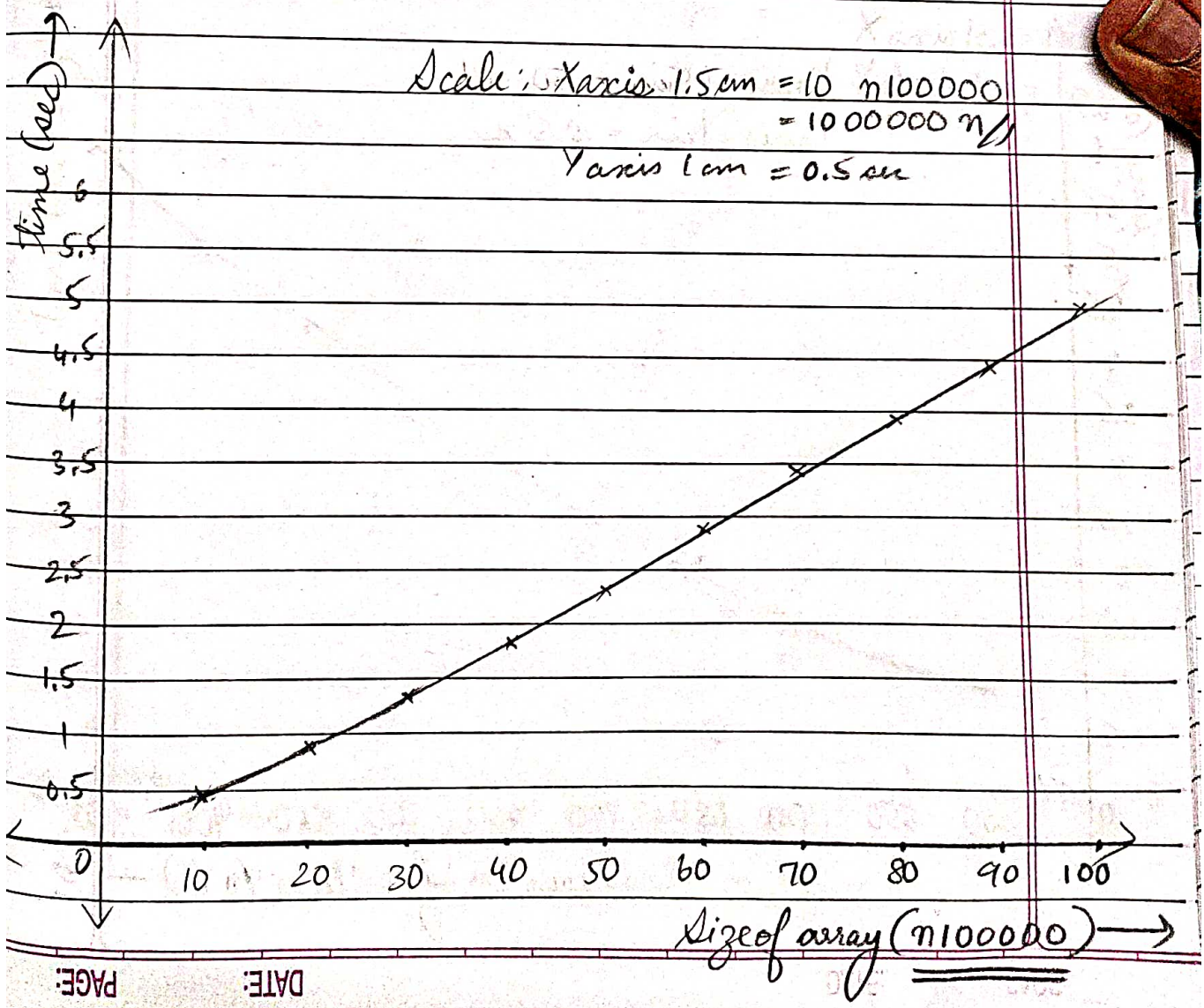
PAGE: DATE:

8) HeapSort (No sleep)

n	t	≈t
10 00 000	0.397414	0.4
20 00 000	0.818897	0.8
30 00 000	1.290093	1.3
40 00 000	1.760143	1.8
50 00 000	2.268754	2.3
60 00 000	2.770901	2.8
70 00 000	3.380562	3.4
80 00 000	3.884239	3.9
90 00 000	4.409547	4.4
10000000	5.007718	5.0

Avg, Worst O(n log n)

Best → distinct keys → O(n log n)
↳ equal keys → O(n).



PAGE: DATE:

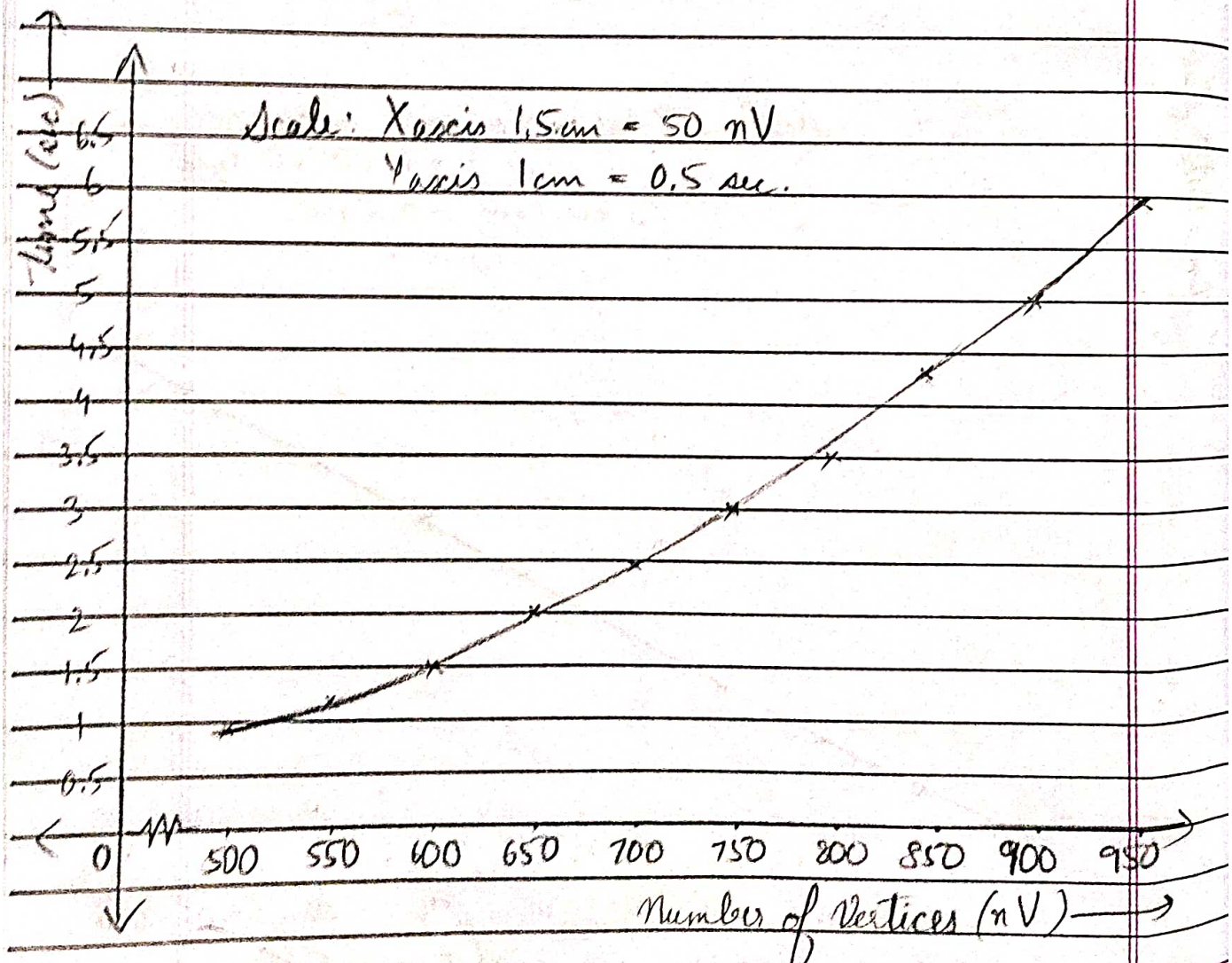
graph algo. → Better with adjacency matrix than adjacency list for this algo.

9) FloydWarshall (No sleep)

nV	t	≈t
500	0.884352	0.9
550	1.170444	1.2
600	1.540441	1.5
650	1.921768	2.0
700	2.434800	2.4
750	2.907238	3.0
800	3.563287	3.5
850	4.267111	4.3
900	4.978280	5.0
950	5.889677	5.9

$A, B, W \rightarrow O(n^3)$

space comp. $\rightarrow O(n^2)$

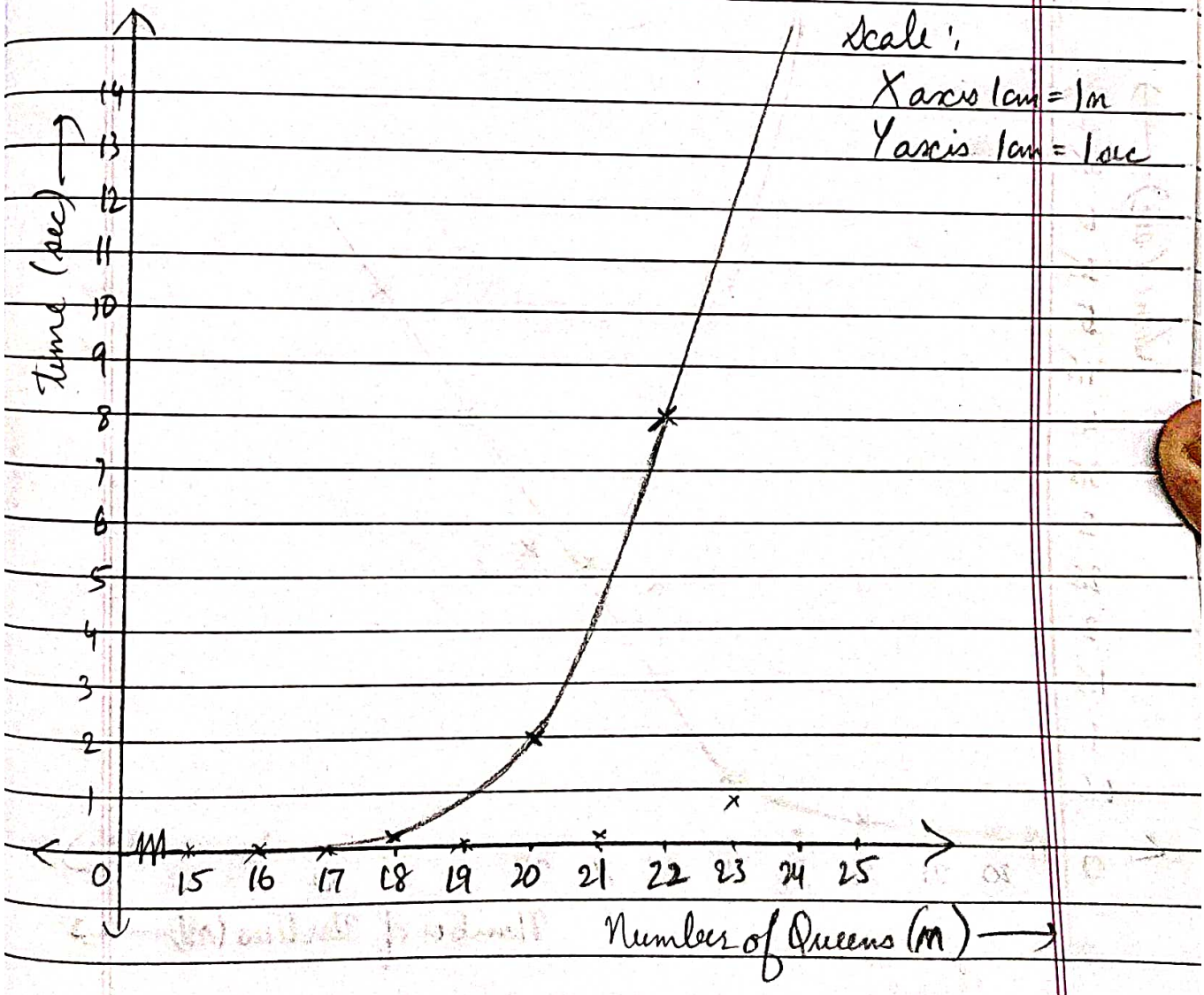


DATE: PAGE:

10] nQueen (no sleep)

n	t	≃ t
15	0.001754	0.0
16	0.033800	0.0
17	0.005706	0.0
18	0.206935	0.2
19	0.019543	0.0
20	2.044740	2.0
21	0.123243	0.1
22	7.984163	8.0
23	0.842578	0.8
24	63.239379	63.2

$O(n!)$



PAGE:

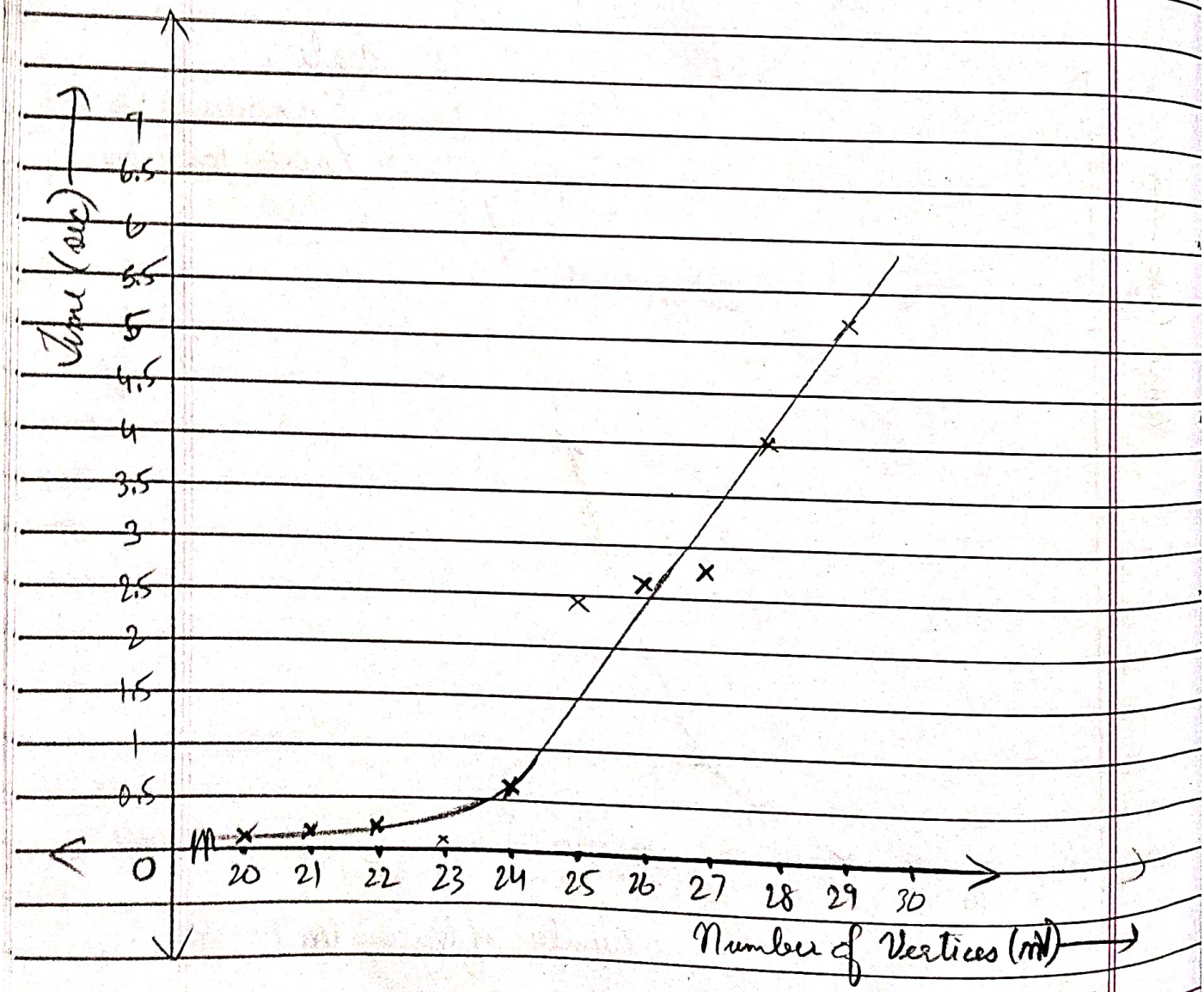
DATE:

$O((N-1)!) = O(2^{n-1})$

11] Travelling Salesman (No sleep)

nV	t	≈t
20	0.108445	0.1
21	0.149911	0.2
22	0.254713	0.3
23	0.12551	0.1
24	0.5714786	0.6
25	2.43544	2.4
26	2.555434	2.6
27	2.74792	2.7
28	4.05655	4.0
29	5.20842	5.2

$O(n^2 * 2^n)$



PAGE:

DATE: