1 P1 Solutions

a) $\Theta(n^3)$

b) $\Theta(n^2)$

c) $\Theta(\sqrt{n}\log n)$

d) $\Theta(n^2)$

2 P2 Solutions

(a) Asymptotic Worst-Case Running Times

1. Recursive Fibonacci

- Recurrence: $T(n) = T(n-1) + T(n-2) + O(1)$.
- Solution grows **exponentially**: $T(n) = O(2^n)$.

2. Iterative Fibonacci

- Runs a simple loop from 2 to $n-1$, each iteration in $O(1)$ time.
- Overall running time is **linear**: $T(n) = O(n)$.

(b) Empirical Tests

(i) Table

n	8	11	14	17	20
Iterative (s)	0.000002	0.000001	0.000001	0.000001	0.000001
Recursive (s)	0.000005	0.000009	0.000031	0.000131	0.000592
n	23	26	29	32	35
Iterative (s)	0.000002	0.000004	0.000005	0.000006	0.000005
Recursive (s)	0.002493	0.010482	0.044156	0.193094	0.776464
n	38	41	44	47	50
Iterative (s)	0.000006	0.000006	0.000006	0.000007	0.000007
Recursive (s)	3.241548	14.735663	60.933380	257.445385	1069.465645

Table 1: Experimental running times for Fibonacci algorithms in seconds

Machine Specifications:

- CPU: Ryzen 5 5600H

- RAM: 16 GB
- OS: Windows 11

(ii) Graph

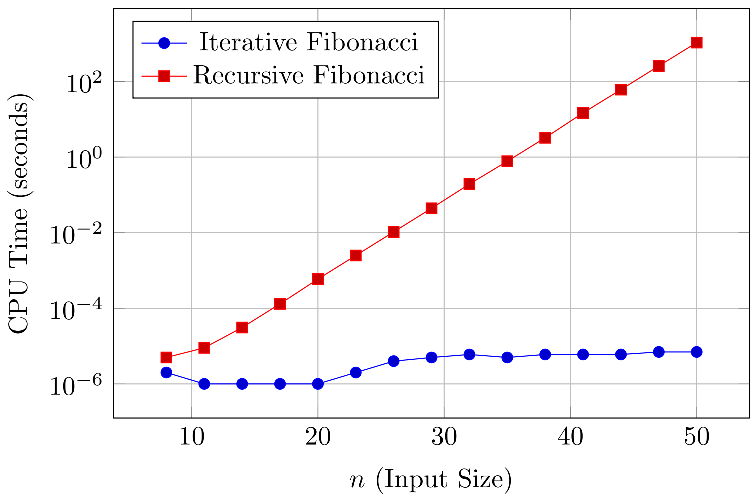

Figure 1: Empirical Running Times of Fibonacci Algorithms

(iii) Discussion of Scalability

- **Recursive:** The exponential $O(2^n)$ behavior makes recursion extremely slow for large n.
- **Iterative:** The linear $O(n)$ behavior is fast and scales well.

Conclusion: The empirical results confirm the theoretical analysis. The recursive method is infeasible for large n, while the iterative method is efficient.