Part I: Key Concepts Revision

Before diving into the specific problems, here is a quick summary of the core concepts required to solve them.

1. The Master Theorem

Used to solve recurrences of the form $T(n) = aT(n/b) + f(n)$, where $a \geq 1$, $b > 1$. a is the number of subproblems, n/b is the size of each subproblem, and $f(n)$ is the cost of the work done outside the recursive calls. Compare $f(n)$ with the critical exponent $n^{\log_b a}$:

- **Case 1 (Leaf Heavy):** If $f(n) = O(n^{\log_b a - \epsilon})$ for $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
- **Case 2 (Balanced):** If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
- **Case 3 (Root Heavy):** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for $\epsilon > 0$ and satisfies the regularity condition, then $T(n) = \Theta(f(n))$.

2. QuickSort Analysis

- **Pivot Quality:** The performance depends heavily on the split balance.
- **Best Case:** Perfectly balanced splits ($n/2$ vs $n/2$). Depth is $\Theta(\log n)$.
- **Worst Case:** Highly unbalanced splits (0 vs $n-1$). Depth is $\Theta(n)$.
- **Probability:** A random pivot is "good" (yielding a split better than α to $1-\alpha$) with probability $1 - 2\alpha$.

3. Recursion Tree Depth

For a problem of size n that shrinks by a factor of X at each step (e.g., $n \to n/X$):

- The depth of the recursion tree is $\log_X n$.
- Using change of base formula: $\log_X n = \frac{\ln n}{\ln X}$.

Part II: Problem Solutions

Question 1: Master Method Application

Problem: Solve $T(n) = 5T(n/3) + 4n$.
Solution:

1. Identify parameters: $a = 5, b = 3, f(n) = 4n$.
2. Calculate the critical exponent: $\log_b a = \log_3 5$.
3. Estimate the value: Since $3^1 = 3$ and $3^2 = 9$, we know $1 < \log_3 5 < 2$. Specifically, $\log_3 5 \approx 1.46$.
4. Compare $f(n)$ with $n^{\log_3 5}$:
$$f(n) = \Theta(n^1) \quad \text{vs} \quad n^{1.46}$$
Since $1 < 1.46$, $f(n)$ grows polynomially slower than $n^{\log_3 5}$. This fits **Case 1**.
5. **Result:** $T(n) = \Theta(n^{\log_3 5})$.

Question 2: FastPower Algorithm

Problem: Analyze the running time of the recursive exponentiation algorithm.
Solution:

- The algorithm computes a^b.
- **Recursive Step:** It computes $a^{\lfloor b/2 \rfloor}$ recursively.
- **Work per step:** Constant time arithmetic operations ($\Theta(1)$).
- **Recurrence:** $T(b) = T(b/2) + \Theta(1)$.
- This recurrence describes a process where the input is halved at every step. The depth is logarithmic.
- **Result:** $\Theta(\log b)$.

Question 3: QuickSort Pivot Probability

Problem: Probability that a random pivot produces a split where the smaller side is $\geq \alpha n$.
Solution:

1. Consider the sorted order of elements 1 to n.
2. To have a "bad" split (smaller side $< \alpha n$), the pivot must be in the first αn elements (too small) or the last αn elements (too big).
3. Total "bad" pivots $= \alpha n + \alpha n = 2\alpha n$.
4. Total "good" pivots = Total elements - Bad pivots $= n - 2\alpha n$.
5. Probability $= \frac{\text{Good Pivots}}{\text{Total Pivots}} = \frac{n(1-2\alpha)}{n}$.
6. **Result:** $1 - 2\alpha$.

Question 4: Recursion Depth Range

Problem: Find the range of depth d if split sizes are between αk and $(1-\alpha)k$.
Solution: The depth d is determined by how fast n reduces to 1.

- **Minimum Depth (Fastest Reduction):** Occurs when we always get the smallest fraction α.

$$n \cdot \alpha^d = 1 \implies d = \log_{1/\alpha} n = -\frac{\log n}{\log \alpha}$$

- **Maximum Depth (Slowest Reduction):** Occurs when we always get the largest fraction $(1-\alpha)$.

$$n \cdot (1-\alpha)^d = 1 \implies d = \log_{1/(1-\alpha)} n = -\frac{\log n}{\log(1-\alpha)}$$

Result: $-\frac{\log n}{\log \alpha} \leq d \leq -\frac{\log n}{\log(1-\alpha)}$

Question 5: QuickSort Recursion Depth (Min/Max)

Problem: Minimum and maximum recursion depth of QuickSort.
Solution:

- **Minimum Depth (Best Case):** The pivot is always the median, splitting the array into $n/2$ and $n/2$. The tree is perfectly balanced.

$$\text{Depth} = \Theta(\log n)$$

- **Maximum Depth (Worst Case):** The pivot is always the min or max, splitting the array into 0 and $n-1$. The tree becomes a linked list.

$$\text{Depth} = \Theta(n)$$

Result: Min: $\Theta(\log n)$, Max: $\Theta(n)$.

Part III: Optional Theory Problems

1. Square Root Recurrence

Recurrence: $T(n) \leq T(\lfloor \sqrt{n} \rfloor) + 1$. **Solution:** Let $n = 2^k$. Then $\sqrt{n} = 2^{k/2}$. The recurrence on the exponent k is $S(k) = S(k/2) + 1$. This means the exponent is halved at every step. It takes $\log k$ steps to reach base case. Since $k = \log n$, the result is $\Theta(\log k) = \Theta(\log \log n)$.

2. Local Minimum in Grid

Problem: Find local minimum in $n \times n$ grid in $O(n)$. **Solution:** Use Divide and Conquer.

1. Find the minimum element on the central row and central column. Let this be m.
2. If m is a local minimum, return it.
3. If not, move to the quadrant containing the neighbor smaller than m.
4. Recurrence: $T(n) = T(n/2) + O(n)$ (where $O(n)$ is scanning the cross).
5. This sums to $O(n)$.