Question Bank: Analysis of Algorithms
Dr. Mudassir Shabbir, March 13, 2025

---

1 Solve the following recurrence relations:

a. $T(n) = 2T(n/3) + 1$
b. $T(n) = 5T(n/4) + n$
c. $T(n) = 9T(n/3) + n^2$
d. $T(n) = 2T(n-1) + 1$
e. $T(n) = T(\sqrt{n}) + 1$

2 Solve recurrences.

- $T(n) = 8T(n/3) + n$
- $T(n) = 2T(n/2) + n^{0.5}$
- $T(n) = 3T(n/4) + n \log n$
- $T(n) = T(n-2) + n^2$

3 Solve recurrences.

- $T(n) = 8T(n/4) + n^{1.5}$
- $T(n) = 3^n T(n/2) + n^{0.5}$

4 Use the recursion tree method to solve $T(n) = 3T(n/4) + 5n^2$. Draw or describe the cost at each level of recursion and sum it up to find the total asymptotic cost. What is the Big-Θ solution for $T(n)$?

5 The Tower of Hanoi puzzle satisfies the recurrence $T(n) = 2T(n-1)+1$ with base $T(1) = 1$. Use the substitution method (induction) to solve this recurrence and find a closed-form solution for $T(n)$.

6 Recurrence with Log Factor: Solve the recurrence $T(n) = 2T(n/2) + n \log^2 n$. Which method or case of the Master Theorem is appropriate here, and why?

7 Unbalanced Divide-and-Conquer (Quicksort Average-case): Consider the recurrence $T(n) = T(n/3) + T(2n/3) + O(n)$, which arises from the average-case of QuickSort. Explain how you would solve this recurrence – for instance, by using a recursion tree or the more general Akra-Bazzi theorem (subproblem sizes are uneven) – and give the Big-O solution.

8 Strassen's Algorithm: The Strassen matrix multiplication algorithm follows the recurrence $T(n) = 7T(n/2) + O(n^2)$. Use the Master Theorem to determine the time complexity of this algorithm.

9 Solve the recurrence relation: $T(n) = 3T(n/2) + 4T(n/4) + O(n \log n)$ using any standard method to solve recursions.

10 Order according to increasing asymptotic growth - provide one line reason for each consecutive pair.

$$2^{3n}, 3^{2n}, n!, 2^{\log^* n}, \log^* 2^{2n}$$

11 Order according to increasing asymptotic growth - provide one line reason for each consecutive pair.

$$2^{3n}, 3^{2n}, n!, 2^{\log^* n}, \log^* 2^{2n}$$

12 Give runtime complexity of the following algorithm.

```
procedure ALGO1(x)                                  ▷ input x is a number
    if |x| > 2 then
        return ALGO1(√x)
    else
        return x
    end if
end procedure
```

13 Let us consider the random experiment of rolling two dice.

1. Define a random variable X as the number of 6's that you get. Find the expected value of X.
2. Define a random variable Y as the sum of the two dice. Find the expected value of Y.

14 Let X be a random variable that is equal to the number of heads in two flips of a fair coin. What is $E(X^2)$? What is $(E(X))^2$?

15 A company releases a series of n collectible cards. Every time a customer buys a pack, they receive a single random card (each card is equally likely). Find the expected number of purchases required to collect all n cards.

16 A gambler starts with "dollar A" and repeatedly bets "dollar 1" on a fair coin flip. If they win, they gain "dollar 1"; if they lose, they lose "dollar 1". The game stops when they reach "dollar B" or lose everything (0).
Find the expected number of bets until the game ends.

17 Willy Wonka has hidden **5 Golden Tickets** inside **1,000,000** randomly distributed chocolate bars. Each bar has an **equal chance** of containing a Golden Ticket.
You are a chocolate fanatic who can buy and open **one chocolate bar per day** until you find a Golden Ticket.
Tasks:

1. What is the **expected number** of chocolate bars you need to buy before you find a **Golden Ticket**?
2. What if **you and your friend both start buying bars separately**—what is the expected number of bars before at least one of you finds a ticket?
3. If **100,000 people** start buying chocolate bars at the same time, how many are expected to find a ticket before all five are discovered?

18 An array $A[1 \ldots n]$ is said to have a **majority element** if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to determine whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, so comparisons of the form "is $A[i] > A[j]$?" are not allowed. (Think of the array elements as GIF files, for instance.) However, you can answer questions of the form: "is $A[i] = A[j]$?" in constant time.

Part 1: Show how to solve this problem in $O(n \log n)$ time. **Hint:** Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a *divide-and-conquer* approach.

Part 2: Can you give a linear-time algorithm? **Hint:** Here's another *divide-and-conquer* approach:

- Pair up the elements of A arbitrarily to get $n/2$ pairs.
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them.

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if and only if A does.

19 Write an algorithm to merge three sorted arrays into a single sorted array.

20 Write an algorithm to merge k sorted arrays into a single sorted array.

21 Given two arrays A and B, write an algorithm to find the median of $A \cup B$ in the following cases:

- When A and B are sorted.
- When A and B are unsorted.

22 Give an $O(n \log k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (Hint: Use a min-heap for k-way merging.)

23 You are given an array of strings, where different strings may have different numbers of characters, but the total number of characters over all the strings is n. Show how to sort the strings in $O(n)$ time. (Note that the desired order here is the standard alphabetical order; for example, $a < ab < b$.)

24 What is the running time of the Breadth-First Search (BFS) algorithm if we represent its input graph using an adjacency matrix and modify the algorithm to handle this form of input?

25 The **diameter** of a tree $T = (V, F)$ is defined as $\max\{\delta(u, v) : u, v \in V\}$, that is, the largest of all shortest-path distances in the tree. Give an efficient algorithm to compute the diameter of a tree, and analyze the running time of your algorithm.

26 Given an n bit integer, write an efficient algorithm to return the square of n. In particular, prove that time complexity of product of two numbers is same as the time complexity of squaring a number, i.e., if you can square a n bit number is $O(f(n))$ then you can also take the product of two n bit numbers $O(f(n))$.

27 Suppose you are given a "black box" linear-time median finding algorithm. Design a simple linear-time algorithm to find k^{th} smallest element using this algorithm. Note that median finding algorithm that you are given is black-box and you can't change it in anyway; you can just call it to find median of a list in linear-time.

28 Solve recurrences.

- $T(n) = 8T(n/3) + n$
- $T(n) = 2T(n/2) + n^{0.5}$
- $T(n) = 3T(n/4) + n \log n$
- $T(n) = T(n-2) + n^2$

29 Order according to increasing asymptotic growth - provide one line reason for each consecutive pair.

$$2^{3n}, 3^{2n}, n!, 2^{\log^* n}, \log^* 2^{2n}$$

30 Given a list of n integers, design an efficient algorithm to find a *majority* element. A majority element is an element that appears more than $\lfloor n/2 \rfloor$ times.

31 Given an unweighted directed graph $G = (V, E)$ and a source vertex s and a destination vertex t, design an algorithm to find a shortest path from s to t.

32 Given a simple undirected graph $G = (V, E)$, design an algorithm to compute a new graph $H = (V, F)$ where $(u, v) \in F$ *if and only if,* $d(u, v) \leq 2$ in G, i.e., u, v are adjacent in H whenever their distance is 1 or 2 in the original graph G.

33 You are given m arrays $A_1, A_2, \ldots, A_m$ where each A_i contains n integers in sorted order. Given an $O(nm \log m)$ algorithm to merge these m sorted arrays into a single sorted array. You may use Priority Queues.

34 Given a set P of n two dimensional points, i.e., each point is represented as (x_i, y_i), give an efficient algorithm to compute closest pair of points in P, i.e., points in P for which Euclidean distance is minimized.

35 Solve recurrences.

- $T(n) = 8T(n/4) + n^{1.5}$
- $T(n) = 3^n T(n/2) + n^{0.5}$

36 Order according to increasing asymptotic growth - provide one line reason for each consecutive pair.

$$2^{3n}, 3^{2n}, n!, 2^{\log^* n}, \log^* 2^{2n}$$

```
procedure ALGO1(x)                                          ▷ input x is a number
    if |x| > 2 then
        return ALGO1(√x)
    else
        return x
    end if
end procedure
```

37 Give runtime complexity of the following algorithm.

38 How to sort n integers in the range $[0, n^3 - 1]$ in linear time?

39 Design and analyze an algorithm to count number of triangles in a simple undirected graph on n vertices.

40 Use the recursion tree method to solve $T(n) = 3T(n/4) + 5n^2$. Draw or describe the cost at each level of recursion and sum it up to find the total asymptotic cost. What is the Big-Θ solution for $T(n)$?

41 The Tower of Hanoi puzzle satisfies the recurrence $T(n) = 2T(n-1) + 1$ with base $T(1) = 1$. Use the substitution method (induction) to solve this recurrence and find a closed-form solution for $T(n)$.

42 Recurrence with Log Factor: Solve the recurrence $T(n) = 2T(n/2) + n \log^2 n$. Which method or case of the Master Theorem is appropriate here, and why?

43 Unbalanced Divide-and-Conquer (Quicksort Average-case): Consider the recurrence $T(n) = T(n/3) + T(2n/3) + O(n)$, which arises from the average-case of QuickSort. Explain how you would solve this recurrence – for instance, by using a recursion tree or the more general Akra-Bazzi theorem (subproblem sizes are uneven) – and give the Big-O solution.

44 Strassen's Algorithm: The Strassen matrix multiplication algorithm follows the recurrence $T(n) = 7T(n/2) + O(n^2)$. Use the Master Theorem to determine the time complexity of this algorithm.

45 Let G be a graph of order n and size m that does not have any cycle of length 3.

- Prove that if u and v are adjacent vertices of G, then $d(u) + d(v) \leq n$.
- Prove that if $n = 2k$, then $m \leq k^2$. Hint: Induction on $k \geq 1$.

46 Seven people attending a conference want to have lunch together at a round table during the three days that the conference lasts. To get to know each other better, they decide to sit in such a way that two people are next to each other at most once. Can they achieve their goal? What happens if the conference lasts 5 days?

47 You are given n points in the unit disk, $p_i = (x_i, y_i)$, such that $0 \leq x_i^2 + y_i^2 \leq 1$ for $i = 1, 2, \ldots, n$. Suppose that the points are uniformly distributed, meaning the probability of finding a point in any region of the disk is proportional to the area of that region.
Design an algorithm with an average-case running time of $\Theta(n)$ to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin.
Hint: Design the bucket sizes in BUCKET-SORT to reflect the uniform distribution of the points in the unit disk.

48 We would like to sort playing cards from a deck. Each card has a denomination (1 to 13) and a suit (CLUBS < DIAMONDS < HEARTS < SPADES).
A card c_1 is considered less than a card c_2 if either of the following is true:

- The suit of c_1 is less than the suit of c_2, or
- c_1 and c_2 are of the same suit, but the denomination of c_1 is less than the denomination of c_2.

(a) Consider sorting cards of the same suit based purely on their denominations using 2-way quicksort. After a random shuffle, we have the sequence:

$$5, 6, 8, 3, 9, 4, 7.$$

Show the result of the first call to `partition()` by listing the exchanged elements.

49 Helping Professor to Form Groups Efficiently for a Discrete Structures Class Activity
The professor is organizing a group activity for a Discrete Structures class and needs to divide students into 5 to 6 groups based on their collaboration history.
Professor has the list of students V.

a) Given a lookup table T, where $T[u]$ for $u \in V$ is a list of students that u has previously worked with, ensure that each group consists of students who are all connected either directly or indirectly. Describe an efficient algorithm that returns the minimum number of groups needed. Analyze its running time.

b) Now suppose there are only two groups available, and you are given a different lookup table S, where $S[u]$ lists students who have conflicts with u (e.g., past disagreements or ineffective teamwork). The goal is to form groups such that no two students in the same group have conflicts. Describe an efficient algorithm that determines whether this is possible, returning TRUE if it is and FALSE otherwise. Analyze its running time.

50 What are the minimum and maximum numbers of elements in a heap of height h?

51 Where in a max-heap might the smallest element reside, assuming that all elements are distinct?

52 At which levels in a max-heap might the kth largest element reside, for $2 \leq k \leq \lfloor n/2 \rfloor$, assuming that all elements are distinct?

53 Is an array that is in sorted order a min-heap?

54 Is the array with values $\{33, 19, 20, 15, 13, 10, 2, 13, 16, 12\}$ a max-heap?

55 Suppose that the objects in a max-priority queue are just keys. Illustrate the operation of MAX-HEAP-INSERT$(A, 10)$ on the heap $A = (15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1)$.

56 Give a brief argument that the running time of PARTITION on a subarray of size n is $\Theta(n)$.

57 What is the running time of QUICKSORT when all elements of array A have the same value?

58 Show that the running time of QUICKSORT is $\Theta(n^2)$ when the array A contains distinct elements, is sorted in decreasing order and first element is selected as pivot everytime..

59 Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a : b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

60 Counting sort can also work efficiently if the input values have fractional parts, but the number of digits in the fractional part is small. Suppose that you are given n numbers in the range 0 to k, each with at most d decimal (base 10) digits to the right of the decimal point. Modify counting sort to run in $\Theta(n + 10^d k)$ time.

61 Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

62 The *transpose* of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$. That is, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

63 There are two types of professional wrestlers: "faces" (short for "babyfaces," i.e., "good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. You are given the names of n professional wrestlers and a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as faces and the remainder as heels such that each rivalry is between a face and a heel. If it is possible to perform such a designation, your algorithm should produce it.

64 Another way to topologically sort a directed acyclic graph $G = (V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $O(V + E)$. What happens to this algorithm if G has cycles?

65 How can the number of strongly connected components of a graph change if a new edge is added?

66 Find a cycle in a graph.

67 Prove that the height of a union-find data structure with path compression is at most $\log n$, i.e., $H(UF) \leq \log n$.

68 KaratSuba when the numbers are divided into 4 parts.

69 Suppose you are given an infinite chessboard with some blocked cells. A knight (which moves in an "L" shape) starts at position $(0, 0)$. Given a target position (x, y), determine the minimum number of moves required to reach the target (or return -1 if impossible).

70 Sorting a Database of 10 BILLION People (Name, Age, Salary, Country)
You have 10 billion records where each person has:

1. **Name** (String, up to 50 characters)
2. **Age** (Integer, 0 to 120)
3. **Salary** (Integer, 4 to 8 digits)
4. **Country** (3-letter country code, e.g., "USA", "IND")

Sort them in the following order:

① **Country** (alphabetically)

② **Age** (ascending)

③ **Salary** (descending)

④ **Name** (alphabetically)

71 You are given N activities, each with a start time S_i and an end time E_i. However, each activity can have an overlapping penalty P_i (if scheduled alongside another overlapping activity). Modify the classic activity selection problem to minimize total penalty while maximizing the number of activities selected.

- **Input:** List of activities with S_i, E_i, P_i
- **Output:** Maximum number of activities with the minimum penalty.

Constraints:

$$1 \leq N \leq 10^5, \quad 1 \leq S_i, E_i, P_i \leq 10^9$$

72 A company releases a series of n collectible cards. Every time a customer buys a pack, they receive a single random card (each card is equally likely). Find the expected number of purchases required to collect all n cards.

73 A gambler starts with "dollar A" and repeatedly bets "dollar 1" on a fair coin flip. If they win, they gain "dollar 1"; if they lose, they lose "dollar 1". The game stops when they reach "dollar B" or lose everything (0).
Find the expected number of bets until the game ends.

74 In an attempt to improve the efficiency of standard **matrix multiplication**, a new approach is proposed:
Instead of performing **direct multiplication** using three nested loops (which takes $O(n^3)$ time), the given $n \times n$ matrices A and B are divided into four equal submatrices as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

The result matrix C is also structured similarly:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

The multiplication is then performed recursively using the formula:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Since each recursive step **operates on smaller submatrices** of size $n/2 \times n/2$, this seems like an improved approach that **leverages recursion** to break down matrix multiplication, potentially making it faster than the traditional $O(n^3)$ schoolbook algorithm.
Tasks:

1. Compare this approach to the standard **schoolbook matrix multiplication** algorithm. Does this method actually improve performance?
2. Provide **complexity analysis** for this approach.
3. Can you suggest some modifications or an approach to **make it do better** than this?
4. Would **parallelizing** this method across **multiple GPUs** lead to better performance?

75 You are given a **dataset containing 500GB** of unsorted **transaction records**, but your system has only **8GB of RAM** available for processing. Since the dataset is too large to fit into memory at once, a straightforward in-memory sorting approach is not feasible. Devise a plan to **efficiently sort** this dataset while minimizing I/O operations.

76 Willy Wonka has hidden **5 Golden Tickets** inside **1,000,000** randomly distributed chocolate bars. Each bar has an **equal chance** of containing a Golden Ticket.
You are a chocolate fanatic who can buy and open **one chocolate bar per day** until you find a Golden Ticket.
Tasks:

1. What is the **expected number** of chocolate bars you need to buy before you find a **Golden Ticket**?
2. What if **you and your friend both start buying bars separately**—what is the expected number of bars before at least one of you finds a ticket?
3. If **100,000 people** start buying chocolate bars at the same time, how many are expected to find a ticket before all five are discovered?

77 You decide to use **Bucket Sort** to sort an array of n floating-point numbers, which are supposed to be **uniformly distributed** in the range $[0, 1]$.
However, something **unexpected happens**—instead of spreading across multiple buckets, **every single element ends up in just one bucket**!
Tasks:

1. What happens to the sorting process when all elements fall into a **single bucket**?
2. What is the **worst-case time complexity** of Bucket Sort in this scenario?
3. How can you **modify** the algorithm to prevent this worst-case scenario?

78 You have n elements to sort and p processors available. Ideally, you want each processor to handle $O(n/p)$ elements, achieving a perfect parallel speedup.
Tasks:

1. Which of **linear-time sorting algorithms** is easiest to parallelize and why?
2. What challenges arise when distributing elements across **multiple processors**?
3. Can you truly achieve $O(n/p)$ speedup, or will overheads and dependencies slow you down?

79 You are given an **unsorted array of n integers** and a number k. Your task is to find the **kth largest element** in the array. However, you **cannot sort** the entire array because it would be too slow for large datasets. Instead, you must find a way to solve the problem efficiently using a **Min-Heap**.

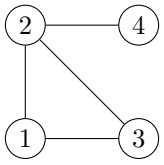
80 Solve the recurrence relation: $T(n) = 3T(n/2) + 4T(n/4) + O(n \log n)$ using any standard method to solve recursions.

81 Sorting and Space Complexity - How can you sort 100MB of data using only 1MB of space?

82 Sorting Algorithms - Why is Quicksort preferred over Merge Sort even when both have $O(n \log n)$ average complexity?

83 Radix Sort Analysis - How does Radix Sort achieve linear time complexity? When does it fail? Why can't its complexity be lower than $O(n \log n)$?

84 Graph Theory - How many paths and walks are there in the given graph? Identify them.



85 Graph Connectivity Check - Given an adjacency list, write a BFS algorithm to determine whether a graph is strongly connected or weakly connected.

86 How can number of strongly connected components change if an edge is removed?

87 Design an algorithm to find all biconnected components in an undirected graph.

88 Design an efficient algorithm to find the second-best minimum spanning tree of a given graph.

89 Find a spanning tree T of a graph G such that the maximum edge weight in T is minimized.

90 Bipartite Graph Verification: Given an undirected graph, determine if it can be divided into two sets such that no vertices within the same set are adjacent. Implement using either BFS or DFS.

91 You have n courses and a list of prerequisite pairs (a, b), meaning "course a must be taken before course b." Model this as a directed graph and use topological sort to determine whether you can finish all courses and, if yes, in which order. If multiple valid schedules are possible, how could you modify your approach to find *all* possible topological sorts?

92 Similar to articulation points, a *bridge* (or cut edge) is an edge whose removal increases the number of connected components. Provide an $O(n+m)$ algorithm to identify all bridges in an undirected graph. Show how the DFS tree can be used to detect these edges.

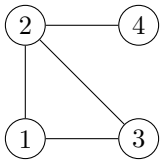
93 How do you check if the Minimum Spanning Tree of a given weighted, connected graph is *unique*? State and justify a procedure that uses MST properties to determine whether multiple MSTs of the same total weight could exist.

94 Given a DAG $G = (V, E)$ with integer edge lengths, describe an $O(n + m)$ algorithm to find the length of the *longest path* in G.

95 Run QuickSort/Median of Medians Algorithm on the following input. Show the array after each partition step.

3 8 2 5 1 4

96 Run SCC Algorithm on the following graph. Show the DFS tree and finish times for each vertex.



97 Run Kruskal/Prim's Algorithm on the following graph. Show the MST and the order in which edges are added.