Engineering Analyst Campus Hiring Program 2025

Congratulations on Qualifying for the Aptitude Test! This document outlines the key topics that will help you prepare for the next test on Computer Science and Quantitative Aptitude Skills. **Please note, the list of topics is indicative and not exhaustive.** We also provided a sample test to give you an idea of the potential pattern of questions.

Test Sections

1. **Programming:** Three programming questions on data structures and algorithms
2. **Quantitative Aptitude:** Multiple choice questions related to Mathematics (sample topics below)
3. **Computer Science:** Multiple choice questions related to Computer Science (sample topics below)
4. **Tell Us About Yourself:** One subjective question to understand your thought process

All questions and sections are required. You increase your chances of qualification by scoring the maximum in all sections. However, scoring in every section is not a prerequisite for qualification. More details are available in the sample Test Paper.

You may refer to standard books, websites, and platforms that cover the below concepts.

Suggested Syllabus for Computer Science

#	Topic	Additional Detail
1.	**Asymptotic Notations: Measuring efficiency of an Algorithm**	**Understanding of:** • Mathematical tools used to describe and analyze the runtime behavior of Algorithms in terms of their time and space complexity in relation to the input set size. • Commonly used asymptotic notations like: Big-O Notation, Omega Notation, Theta Notation etc. Deduction of the above notations for the common algorithmic techniques (e.g., Linear Search, Binary Search, Bubble sort).
2.	**Data Structures**	**Understanding of:** • Basic concepts of organizing and storing data for facilitating efficient access and modification by usage pattern. • Complexity analysis of operations (e.g., Insertion, Deletion, Traversal, Searching, and Sorting) of well-known Data structures. **Examples:** • Introductory Data Structures (e.g., Static Arrays, Sorted Arrays, Dynamic Arrays, Linked Lists). • Linear Data Structures (e.g., Stacks, Queues, Priority Queues) • Non-Linear Data Structures (e.g., Graph, Tries, Min Heap, Max Heap, Binary Search Tree, Self-balancing Trees: B, B+, AVL, Red Black, Skip List). • Dictionary Data Structures (e.g., Hash-Tables, Hash-Maps, Ordered and Unordered Maps and Sets).

3.	**Algorithmic Paradigms**	**Understanding of:** • Various approaches or strategies for designing algorithms to solve computational problems. **Examples:** • Divide and Conquer, Backtracking, Branch and bound, Brute-force search, Dynamic programming, Greedy algorithm, Recursion etc.
4	**Algorithms: Searching**	**Understanding of:** • Searching from an algorithmic standpoint and a grasp on key concepts is expected. **Examples:** • Types: Linear Search, Binary Search • Complexity Analysis: In terms of Time and Space Complexity • Data Structures: Common Data Structures (e.g., Arrays, BST, AVL, RB, Hash-tables) and their applicability in Linear or Binary or both. • Practical Considerations: Implications based on sorted vs un-sorted data. Size of datasets etc. And the finding right algorithms to employ based on these pre-conditions.
5.	**Algorithms: Sorting**	**Understanding of:** • Comparison based and Linear time sorting algorithms and ability to use Decision Tree Model for analysis of sorting algorithm. • Performance characteristics in terms of space and time complexity of well-known sorting methods (e.g., Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Radix Sort, Bucket Sort, Counting Sort). • Implementation of sorting algorithms. • Identifying application of a particular sorting technique based on the input data set characteristics.
6.	**Algorithm: Traversal**	**Understanding of:** • Traversal of connected data structures like Trees, and Graphs and in-depth knowledge of different traversal techniques and complexities - Depth first (Pre-Order, In-Order, and Post-Order), Breadth first - Level Order. • Directed Acyclic Graph. • Implementing Minimum Spanning Tree using algorithms like Prim's Algorithm, Kruskal's algorithm. • Implementing common Algorithms used to find "shortest" / "optimal" path between two nodes in graph (e.g., Dijkstra's Algorithm, in depth understanding of purpose, approach, and complexity of these algorithm is recommended). • Choosing appropriate algorithm and implement based on what "optimal" means (e.g., Shortest Distance, least cost, or minimum time). • Topological Sorting.

8.	**Object Oriented Programming: Concepts**	**Understanding of:** • OOP concepts, and its application to achieve modular, re-usable and maintainable code **Examples:** • Abstraction, Encapsulation, Inheritance - Single, Multi-level, Hierarchical, and Multiple, Polymorphism - Static and Dynamic, Abstraction and Composition).

9.	**Aptitude Test Topics**	**For the Quantitative aptitude question, awareness and understanding of the below topics is expected**
		• Probability and Statistics
		  ○ Permutations & Combinations
		  ○ Bayes theorem
		  ○ Random variables
		  ○ Expected values
		  ○ Distributions
		  ○ Means & Variances
		• Quantitative Aptitude
		  ○ Time and work
		  ○ Time speed and distance
		  ○ Percentage, Averages
		  ○ Geometry
		  ○ Number theory

Programming Guidelines

There would be programming questions that would test problem solving skills and ability to program in either Python, C++, C or Java. The problems would derive from concepts covered in the Computer Science section above. Test cases would evaluate code for correctness and performance of the solution.

The below links could be a Good starting point for your learning journey:

Algorithm:

https://www.coursera.org/learn/algorithms-part1

https://www.coursera.org/learn/algorithms-part2

Algo and DS:

https://techdevguide.withgoogle.com/resources/courses/dsa/?no-filter=true

https://www.w3schools.com/dsa/

https://www.freecodecamp.org/news/learn-data-structures-and-algorithms/

Contact Us

In case you have any questions, please post them via email to: Enganalysthiringprog@ny.email.gs.com