UnCodemy

Data Structures Important Questions PDF

The most important Data Structures questions for B.Tech exams are infix to postfix conversion, linked list operations, circular queue, binary tree traversals, BST and AVL insertion, graph BFS and DFS, Dijkstra and Kruskal, quick sort, heap sort and hashing with collision handling. Practise each with a worked example, because most long answers ask you to trace an algorithm step by step.

Key takeaways

- Algorithm tracing (sorting, traversals, conversions) is the most repeated question type in Data Structures papers.
- Write every algorithm in clear pseudocode with its time complexity.
- Trees and graphs usually need the most time, so practise them on paper.
- These are practice questions, not past papers; check your own syllabus and official previous papers.

About these important questions

This page lists the most important Data Structures questions for B.Tech CSE and IT exams, arranged unit by unit. The units follow the standard syllabus that most Indian technical universities share, so the order may differ slightly from yours.

Please note: these are practice questions based on common syllabus topics. They are not verified past papers, and we do not claim that any question will appear in your exam. Check your own university syllabus first, then solve the official previous year papers. AKTU students can download them from the AKTU old question paper archive.

Unit 1: Introduction, Arrays and Recursion

Covers abstract data types, complexity, asymptotic notation, arrays and recursion.

Short questions (2 marks)

- Define a data structure and an abstract data type (ADT).
- Differentiate between linear and non-linear data structures.
- What is the time-space trade-off?
- Give the formula for the address of A[i][j] in row-major and column-major order.

Long questions

1. Explain Big O, Omega and Theta notation with graphs and one example each.
2. A 2D array A[-5..5, 3..10] has base address 1000 and 4 bytes per element. Find the address of A[2][6] in row-major and column-major order. *Hint:* find the number of columns (8) first, then apply the formula carefully with the lower bounds.
3. Write a recursive algorithm for Tower of Hanoi and derive its time complexity.
4. Explain sparse matrices and their triplet representation with an example.

Unit 2: Stacks and Queues

Covers stack operations, expression conversion, queue types and their applications.

Short questions (2 marks)

- What is stack overflow and underflow?
- Differentiate between a stack and a queue.
- What is a priority queue?

Long questions

1. Convert the infix expression `A + (B * C - (D / E ^ F) * G) * H` to postfix using a stack. Show the stack at every step in a table.
2. Evaluate a postfix expression such as `5 6 2 + * 12 4 / -` using a stack.
3. Write algorithms for insertion and deletion in a circular queue and explain why it is better than a linear queue.
4. Explain a deque and the input-restricted and output-restricted variants.
5. How do you implement a queue using two stacks?

Write the conversion table with three columns: symbol scanned, stack, postfix output. Examiners look for this table.

Unit 3: Linked Lists

Covers singly, doubly and circular linked lists and their uses.

Short questions (2 marks)

- List two advantages of a linked list over an array.
- What is a header node?
- What is a circular linked list?

Long questions

1. Write algorithms to insert a node at the beginning, end and a given position of a singly linked list.
2. Write an algorithm to delete a node from a doubly linked list and explain the pointer changes with a diagram.
3. Write an algorithm to reverse a singly linked list. *Hint:* use three pointers: previous, current and next.
4. Represent the polynomial 5x^3 + 4x^2 + 2 using a linked list and write an algorithm to add two polynomials.

5. Explain a stack and a queue implemented with linked lists.

Unit 4: Trees

Covers binary trees, BST, AVL trees, B-trees, heaps and Huffman coding. This unit often carries the heaviest long questions.

Short questions (2 marks)

- Define a complete and a full binary tree.
- What is the balance factor in an AVL tree?
- What is a threaded binary tree?

Long questions

1. Construct a binary tree from given inorder and preorder (or postorder) traversals.
2. Insert the keys 50, 20, 60, 10, 8, 15, 32, 46, 11, 48 into a BST. Then delete a node with two children.
3. Insert a sequence of keys into an AVL tree, naming each rotation (LL, RR, LR, RL).
4. Build a B-tree of order 3 or 5 from a list of keys and explain splitting.
5. Construct a Huffman tree for given character frequencies and find the codes.

Unit 5: Graphs, Sorting, Searching and Hashing

Covers graph representation and traversal, shortest paths, spanning trees, sorting, searching and hashing.

Short questions (2 marks)

- Differentiate between an adjacency matrix and an adjacency list.
- What is a stable sort?
- What is a hash collision? Name two ways to resolve it.

Long questions

1. Perform BFS and DFS on a given graph and show the queue or stack contents.
2. Find the minimum spanning tree using Prim's and Kruskal's algorithms.
3. Apply Dijkstra's algorithm to find shortest paths from a source vertex.
4. Sort a list using quick sort, merge sort and heap sort, showing each pass, and compare their complexities.
5. Insert keys into a hash table of size 10 using linear probing, quadratic probing and chaining.
6. Write binary search and state its time complexity.

Sample answer outline: Infix to postfix conversion

A typical 7 to 10 mark question asks you to convert an infix expression to postfix using a stack. Structure the answer like this:

1. **State the rule (2 to 3 lines).** Operands go straight to the output. An opening bracket is pushed. On a closing bracket, pop to the output until the opening bracket. For an operator, pop

operators of higher or equal precedence (except for right-associative ^), then push the new one. At the end, pop everything left.
2. **Give the precedence table.** ^ is highest and right-associative, then * and /, then + and -.
3. **Trace in a table.** Use three columns (symbol, stack, postfix) and one row for every symbol scanned. Do not skip rows.
4. **Write the final answer clearly.** For example, `A + B * C` becomes `A B C * +`.
5. **Add complexity.** The algorithm makes one pass, so it runs in O(n) time with O(n) stack space.

The same structure works for prefix conversion: reverse the expression, swap the brackets, convert, then reverse the result.

Marks weightage and how to write answers

A common pattern, used by AKTU and many other universities, has a compulsory short-answer section that covers every unit, followed by long-answer questions with a choice inside each unit. Marks per question change between universities and years, so confirm the scheme on a recent official paper.

- **Short answers (about 2 marks):** a crisp definition, one example or diagram, and nothing more. Three to five lines is enough.
- **Long answers (7 to 10 marks):** start with a definition, add a labelled diagram or table, explain the steps, and finish with an example, advantages or a comparison.
- For every algorithm, write the best, average and worst case time complexity at the end. It is an easy extra mark.
- Draw trees and graphs neatly with labelled nodes. A messy tree trace is hard to mark.

A 7-day revision plan for Data Structures

1. **Days 1 to 5:** one unit a day. Read your notes, then write the long answers for that unit from memory and check them against your notes.
2. **Day 6:** redo the tracing questions: infix to postfix, AVL rotations, BFS/DFS, Dijkstra and one sort of each kind.
3. **Day 7:** solve one full official previous year paper in three hours, then check which units cost you time.

Keep a single sheet of formulas, definitions and diagrams for the last hour before the exam.

Frequently asked questions

Which unit of Data Structures is most important?

Trees and graphs usually carry the longest questions, because they test tracing skill. Stacks and sorting are also common. Do not skip any unit, since the short-answer section covers all of them.

Should I write code in C or pseudocode in a Data Structures exam?

Either is accepted at most universities unless the question says otherwise. Clear pseudocode with comments is faster and easier to mark. Use C when the question asks for a program.

How do I prepare Data Structures for placements as well?

Learn the same topics with coding practice: implement each structure, then solve easy and medium problems on arrays, linked lists, trees and graphs. The theory you learn for exams helps in interviews.

Are these Data Structures questions from previous year papers?

No. They are practice questions built from topics that appear in most B.Tech Data Structures syllabi. Use them with your own university's official previous papers, for example the AKTU archive at https://aktu.ac.in/old-question-paper.html.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- AKTU: old question papers (official archive)
- NPTEL: online courses from IITs (search for Data Structures)
- MIT OpenCourseWare: Introduction to Algorithms (6.006)

About this PDF

These are practice questions for revision, prepared by the Uncodemy trainer team from common B.Tech syllabus topics. They are not past papers, a certificate or a promise of marks. Spot an error? Write to info@uncodemy.com and the team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550