DAA Important Questions PDF (Unit-wise)

The most important DAA questions are solving recurrences with the Master theorem, merge sort and quick sort analysis, heap sort, red-black trees, Kruskal, Prim and Dijkstra, fractional knapsack, Huffman coding, 0/1 knapsack, LCS, matrix chain multiplication, Floyd-Warshall, N-Queens, sum of subsets, string matching and P, NP and NP-complete classes.

Key takeaways

- Recurrences and the Master theorem appear in nearly every DAA paper.
- Dynamic programming numericals (LCS, matrix chain, 0/1 knapsack) need a filled table to score full marks.
- Always state the algorithm design technique and its time complexity.
- 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 Design and Analysis of Algorithms 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, Recurrences and Sorting

Covers algorithm analysis, asymptotic notation, recurrences and comparison sorts.

Short questions (2 marks)

- Define an algorithm and list its properties.
- Arrange functions such as n log n, n^2, 2^n and log n in increasing order of growth.
- What is the lower bound for comparison-based sorting?

Long questions

1. Solve T(n) = 2T(n/2) + n, T(n) = 7T(n/2) + n^2 and T(n) = T(n/3) + T(2n/3) + n. *Hint:* use the Master theorem for the first two and a recursion tree for the third.

2. Explain quick sort, and derive its best and worst case. When does the worst case occur?
3. Sort a list using heap sort and show the heap after each step.
4. Explain counting sort, radix sort and bucket sort, and say why they beat n log n.
5. Explain Shell sort with an example.

Unit 2: Advanced Data Structures

Covers red-black trees, B-trees, binomial and Fibonacci heaps, tries and skip lists. The exact list varies by university.

Short questions (2 marks)

- List the properties of a red-black tree.
- What is a binomial tree of order k?

Long questions

1. Insert a sequence of keys into a red-black tree, showing recolouring and rotations.
2. Insert and delete keys in a B-tree of minimum degree t = 3.
3. Explain the union of two binomial heaps with an example.
4. Compare the amortised costs of operations on binomial and Fibonacci heaps.
5. Explain a trie and its use in string search.

Unit 3: Divide and Conquer and Greedy Methods

Covers matrix multiplication, convex hull, and greedy problems on graphs and scheduling.

Short questions (2 marks)

- Differentiate between divide and conquer and the greedy method.
- What is an optimal substructure?

Long questions

1. Explain Strassen's matrix multiplication and derive its complexity, O(n^2.81).
2. Solve a fractional knapsack instance with given weights, profits and capacity.
3. Solve job sequencing with deadlines for a given set of jobs.
4. Find the minimum spanning tree with Kruskal's and Prim's algorithms and compare them.
5. Apply Dijkstra's algorithm and explain why it fails with negative edges. Then explain Bellman-Ford.
6. Construct Huffman codes for given frequencies.

Unit 4: Dynamic Programming, Backtracking and Branch and Bound

Covers table-based optimisation and state-space search. This is the highest-scoring numerical unit.

Short questions (2 marks)

- State the principle of optimality.

- Differentiate between backtracking and branch and bound.

Long questions

1. Find the longest common subsequence of two strings and draw the table with arrows.
2. Find the optimal parenthesisation for a matrix chain with given dimensions.
3. Solve a 0/1 knapsack instance with dynamic programming.
4. Apply Floyd-Warshall to a 4-vertex graph and show each matrix.
5. Solve the 4-Queens or 8-Queens problem with a state-space tree.
6. Solve sum of subsets and graph colouring using backtracking.
7. Solve a small travelling salesperson problem using branch and bound.

Unit 5: Selected Topics: String Matching and NP-Completeness

Covers string matching, approximation algorithms and complexity classes.

Short questions (2 marks)

- Define P, NP, NP-hard and NP-complete.
- What is a spurious hit in the Rabin-Karp algorithm?

Long questions

1. Compute the prefix function for a pattern and run the KMP algorithm on a text.
2. Explain the Rabin-Karp algorithm with a numerical example and its worst case.
3. Explain the naive string matching algorithm and its complexity.
4. Prove that the vertex cover or CLIQUE problem is NP-complete, using a reduction.
5. Explain an approximation algorithm for vertex cover and its ratio bound.
6. Explain the fast Fourier transform (FFT) and its use in polynomial multiplication.

Sample answer outline: Longest common subsequence

LCS is one of the most repeated DP numericals. A full-marks answer has five parts:

1. **Define the problem.** Given strings X of length m and Y of length n, find the longest sequence that appears in both in the same order, not necessarily next to each other.
2. **Write the recurrence.** c[i, j] = 0 if i = 0 or j = 0; c[i, j] = c[i-1, j-1] + 1 if x(i) = y(j); otherwise c[i, j] = max(c[i-1, j], c[i, j-1]).
3. **Fill the table.** Draw an (m+1) by (n+1) grid. Fill row by row, and mark each cell with an arrow: diagonal for a match, up or left for the larger neighbour.
4. **Trace back.** Start at c[m, n] and follow the arrows. Every diagonal arrow gives one character of the LCS. Write the LCS and its length.
5. **State the complexity.** O(mn) time and O(mn) space. Mention that the space can drop to O(min(m, n)) if only the length is needed.

Use the same five parts for matrix chain multiplication and 0/1 knapsack: problem, recurrence, table, trace back, complexity.

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 DP questions, draw the full table and mark how you trace back the answer. A final answer without the table loses most marks.
- For NP-completeness, state the two steps: show the problem is in NP, then reduce a known NP-complete problem to it.

A 7-day revision plan for DAA

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:** solve one numerical of each type again: recurrence, LCS, matrix chain, knapsack, Floyd-Warshall and KMP.
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 topics are most important in DAA?

Recurrences, sorting analysis, greedy algorithms (knapsack, MST, Dijkstra), dynamic programming (LCS, matrix chain, 0/1 knapsack), backtracking (N-Queens) and NP-completeness come up most often.

How do I solve recurrences quickly in the exam?

Check first whether the recurrence fits the Master theorem form $T(n) = aT(n/b) + f(n)$. If it does not, use a recursion tree or substitution. Practise five or six standard recurrences until the steps are automatic.

Is DAA useful for coding interviews?

Yes. Greedy, dynamic programming and graph algorithms from DAA are the core of most coding interview problems. Implementing them in code strengthens both exam and interview preparation.

Are these DAA questions from previous year papers?

No. They are practice questions built from topics that appear in most B.Tech Design and Analysis of Algorithms 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)
- MIT OpenCourseWare: Introduction to Algorithms (6.006)
- NPTEL: online courses from IITs (search for Design and Analysis of Algorithms)

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