Finals Question Bank: Analysis of Algorithms
May 29, 2025

---

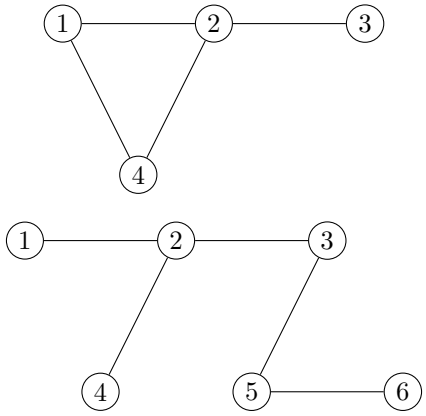
1 Let us consider the random experiment of rolling two dice.

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.

2 Perform the analysis of Median of Median's Algorithm where we make chunks of size 9 instead of 5 and solve the recurrence using substitution method.

3 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 $(E(X))^2$?

4 Apply the algorithm step by step to find the articulation points in the following graphs:



5 Given an undirected graph with distinct non-negative edge weights. Suppose that you have computed shortest paths to all nodes from a particular node s. Now suppose each edge weight is increased by 1; the new weights are $w_e := w_e + 1$. Do the shortest paths change? Give an example where they change or prove they cannot change.

6 Give a general example of a weighted, directed graph G on n nodes where at least half of the edge weights are negative, and Dijkstra's algorithm correctly finds the shortest paths from a source vertex. Note that n is not fixed, so you cannot just give an example on 10 or 20 vertices. Your job is to maximize the number of edges that contain negative weights.

7 How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle?

8 "In GoldExtractor game, you collect coins using two extractors: a red vertical one that moves right and a blue horizontal one that moves up. Each time an extractor moves to a new line containing coins, it collects one coin, and all other coins on that line are lost. The game ends when either extractor reaches its final position, so plan your moves carefully to maximize your total score!"
Design a complete Dynamic Programming algorithm for the problem. Your input is (x,y) coordinates of n coins, and your output is a number which is maximum number of coins collected. You can assume that $0 < x, y < n$.

9 Find the decision version of the minimum spanning tree problem. State the original problem and the decision problem formally.

10 Rod Cutting Problem
Given a rod of length n and an array of prices `price` such that `price[i]` represents the price of a rod of length $i + 1$, determine the maximum value obtainable by cutting up the rod and selling the pieces.

- **Input:** $n = 8$, `price` = [1, 5, 8, 9, 10, 17, 17, 20]
- **Output:** 22 (Cut into rods of lengths 2 and 6 for prices 5 and 17)

11 Prove that the following problem is in NP:
Given an integer x, is x NOT a prime?

12 Reduce 4-SAT to 3-SAT.

13 Reduce Subset Sum to Partition problem.

14 Reduce Partition to Subset Sum problem.

15 Reduce Partition to Bin Packing probem.

16 Reduce Vertex Cover to Set Cover problem.

17 Reduce Maximum Clique to Subgraph Isomorphism problem.

18 Reduce Subset Sum to Knapsack problem.

19 Reduce Vertex Cover to Maximum Dominating Set problem.

20 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. $T(n) = T(\sqrt{n}) + 1$

21 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.

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

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

24 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.

25 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.)

26 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, $a < ab < b$.)

27 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?

28 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.

29 For each of the following strings:

a. Build a frequency table for the characters.

b. Construct the Huffman merge tree.

c. Generate Huffman codes for each character.

d. Encode the full string using the Huffman codes.

e. Compare the number of bits in the Huffman-encoded version with standard ASCII encoding (8 bits per character).

`AGCTTAGGCTTAA`

30 Coin Change Problem

Given coin denominations $c_1, c_2, \ldots, c_n$ and a target amount A, find the minimum number of coins needed to make change for the amount A. Assume an infinite supply of coins of each denomination.

31 Longest Increasing Subsequence

Given a sequence of integers $A = a_1, a_2, \ldots, a_n$, find the length of the longest subsequence such that all elements of the subsequence are sorted in increasing order.

32 Reduce Maximum Independent Set to Maximum Clique problem.

33 Reduce Hamiltonian Cycle to Hamiltonian Path problem.

34 Box Stacking Problem
Definition: Given a set of n boxes, each with height h_i, width w_i, and depth d_i, determine the maximum possible height of a stack that can be formed under the following constraints:

- A box can be rotated so that any side functions as its base.
- A box can only be stacked on top of another if both its width and depth are strictly less than those of the box below it.
- Only one instance of each box can be used in the stack.

Input: An array of n boxes, each defined by three integers: height h_i, width w_i, and depth d_i.
Output: Maximum stack height achievable by stacking the boxes.
Recurrence Relation:

$$\mathrm{DP}[i] = \max(h_i, h_i + \mathrm{DP}[j] \mid j < i \text{ and } w_i < w_j \text{ and } d_i < d_j)$$

Steps:

1. Generate all rotations of each box (3 per box) so width $\geq$ depth.
2. Sort all $3n$ boxes by decreasing base area ($w \times d$).
3. Apply the recurrence using bottom-up DP.

Example: Input: Boxes = [(4,6,7), (1,2,3), (4,5,6)]
Output: 15

35 3SAT to Mario Level Completion
Definition:

- **3SAT:** Given a Boolean formula in CNF with exactly three literals per clause, determine whether there exists a truth assignment that satisfies the formula.
- **Mario Level Completion:** Given a Mario-style platform level with switches, obstacles, and doors, determine whether Mario can reach the goal under the game's rules.

Example (3SAT): Formula: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$
Satisfying assignment: $x_1 = \text{true}$, $x_2 = \text{false}$, $x_3 = \text{true}$
Example (Mario Level): Each variable corresponds to a choice of path: assigning true or false blocks the other path. Each clause is a locked door opened if at least one of its literals' paths has been taken.
Task: Reduce from 3SAT to Mario Level Completion. Construct a Mario level such that Mario can reach the goal if and only if the 3SAT formula is satisfiable.

36 Huffman Coding
Definition: Given a string of characters, generate prefix-free binary codes using Huffman's algorithm to minimize the total encoded message length.
Input String: "data structures and algorithms"
Tasks:

1. Count the frequency of each character (including spaces).
2. Build the Huffman tree from frequencies.
3. Assign binary codes to each character.
4. Encode the original string using those binary codes.

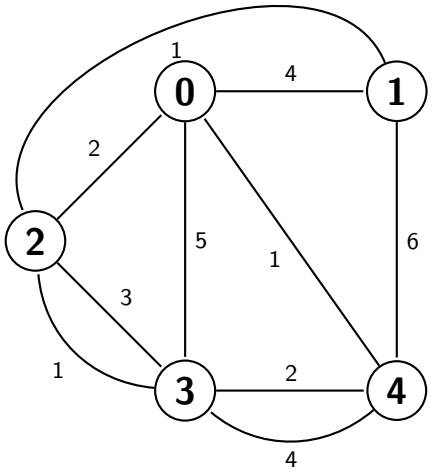
Output: A binary string representing the Huffman encoding of the input, and the binary codes used.

37 Knapsack to Bin Partitioning
Definition:

- **Knapsack Problem:** Given n items with weights w_i and values v_i, and capacity W, determine the maximum total value of a subset of items whose total weight does not exceed W.
- **Bin Partitioning Problem (Decision Version):** Given n items with weights w_i, a bin capacity C, and an integer k, determine whether the items can be partitioned into k or fewer bins, such that the total weight in each bin does not exceed C.

Example (Knapsack):
Items = $\{(4,6),(2,4),(3,5),(5,8)\}$, $W = 7$
Max value = 11 (select items with weights 2 and 5)
Example (Bin Partitioning):
Item weights = $\{4,2,3,5\}$, $C = 7$, $k = 2$
Valid bins: $\{2,5\}$, $\{3,4\}$
Task: Reduce from Knapsack to Bin Partitioning. Construct an instance of the bin problem such that solving it corresponds to solving the original Knapsack instance.

38 All-Pairs Shortest Paths Using Floyd-Warshall Algorithm
Problem Definition:
Given a directed graph $G = (V, E)$ with edge weights (positive or negative, but no negative cycles), compute the shortest path distances between every pair of vertices using the Floyd-Warshall algorithm.
Input:
A weighted directed graph with $n = 5$ vertices and the following edges shown in the diagram:



Output:
A matrix D such that $D[i][j]$ contains the shortest path distance from vertex i to vertex j.
Recurrence Relation:
Let $D^{(k)}[i][j]$ be the shortest distance from vertex i to j using only intermediate vertices from the set $\{0, 1, ..., k\}$. Then:

$$D^{(0)}[i][j] = \begin{cases} 0 & \text{if } i = j \\ w(i,j) & \text{if } (i,j) \in E \\ \infty & \text{otherwise} \end{cases}$$

$$D^{(k)}[i][j] = \min\{D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j]\}$$

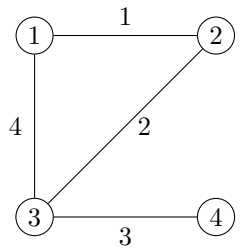
Example: Using the provided graph, the initial distance matrix $D^{(0)}$ is: $\begin{pmatrix} 0 & 4 & 2 & 5 & 1 \\ \infty & 0 & \infty & \infty & 6 \\ \infty & 1 & 0 & 3 & \infty \\ \infty & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & 4 & 0 \end{pmatrix}$
After applying Floyd-Warshall, the final distance matrix D gives the shortest paths between all pairs of vertices.

39 Smallest Number with Given Digit Count and Sum
Given two integers s and d, find the smallest possible number that has exactly d digits and a sum of digits equal to s. Return the number as a string. If no such number exists, return "-1".
Examples:

- **Input:** $s = 9$, $d = 2$
  Output: "18"
  Explanation: 18 is the smallest number possible with the sum of digits = 9 and total digits = 2.
- **Input:** $s = 20$, $d = 3$
  Output: "299"
  Explanation: 299 is the smallest number possible with the sum of digits = 20 and total digits = 3.

- **Input:** $s = 1$, $d = 1$
  Output: "1"
  Explanation: 1 is the smallest number possible with the sum of digits = 1 and total digits = 1.

40 Minimum Spanning Tree (MST)
Given an undirected, weighted and connected graph G with n vertices and m edges, find the weight of the Minimum Spanning Tree.
Example:



MST weight = 6 (edges with weights 1, 2, and 3)

41 Given an array of n integers, find the length of the longest increasing subsequence.
Example: Given an array $A = [10, 9, 2, 5, 3, 7, 101, 18]$
LIS = $[2, 3, 7, 101]$ and the length is 4.

42 Show how the Maximum Independent Set problem can be reduced to the Maximum Clique problem.

43 Huffman Coding
Construct a Huffman Tree using the given frequency table, then determine the Huffman Encoding for each character.

Character	Frequency
A	5
B	9
C	12
D	13
E	16
F	45

44 Minimum Cost with K Stops (Flight Problem)
Given flights between cities with costs, a source, destination, and max K stops allowed, find the minimum cost path using a modified Dijkstra or BFS+DP approach.

45 SSSP with Edge Penalty
Each edge has a weight w and a penalty p. If you take two consecutive edges with the same penalty, you pay the penalty only once. Find the shortest path from the source to destination minimizing the total cost (weight + penalty rules).

46 Transitive Closure using Floyd-Warshall
Given a directed graph, compute the transitive closure (i.e., for every pair (u, v), determine if v is reachable from u). You must modify Floyd-Warshall to work with boolean matrix.

47 Minimum Number of Platforms (Train Problem)
Given arrival and departure times of trains at a station, find the minimum number of platforms required so that no train waits.

48 Longest Bitonic Subsequence
Find the length of the longest subsequence that first increases and then decreases. Combine LIS from left and LDS from right for each position.

49 Subset Product Problem
Given a set of positive integers and a target product P, is there a subset whose product is exactly P?

50 Knapsack Problem
You're going on a treasure hunt with a backpack that can carry only W kg. There are n items, each with a weight and value. You can take each item only once or not at all. Which items should you pick to get maximum value without overloading your backpack?

51 Fractional Knapsack
You're still collecting treasure! But this time, your magical bag allows you to take fractions of items. You can take half a gold bar if needed. Your goal? Maximize the loot you can carry in the same W kg bag.
And will greedy approach fail? Explain with example.

52 Travelling Salesman Problem (TSP)
You are a delivery person. You must visit all cities once, then return home. Your task is to find the shortest path that does this.
You're given a weighted graph of cities and a bound of total cost you incur during your trip. For this problem, prove it is an NP-hard problem.

53 Transitive Closure using Floyd–Warshall
You want to know who is related to whom in a network. Even if there's no direct relation, can person A be related to person B through others?
Instead of a list of all relations, you just want Yes/No for every pair.

54 Bellman–Ford Algorithm – Single Source Shortest Path
You're navigating a graph that might have negative tolls (yep, they pay you!). Find the shortest path from one place to all others.
Also, detect if there's a nasty loop giving infinite money.

55 Given an instance of the CLIQUE problem, construct a polynomial-time transformation that converts it into an instance of the INTERVAL SCHEDULING WITH CONSTRAINTS problem such that:
The original graph $G = (V, E)$ contains a clique of size k if and only if the corresponding interval scheduling instance allows selecting k non-overlapping intervals that satisfy the specified constraints.
Definitions:

- **CLIQUE Problem:** Given a graph $G = (V, E)$ and an integer k, does G contain a clique (a complete subgraph) of size k?
- **Interval Scheduling with Constraints:** Given a set of intervals with associated constraints (e.g., mutual exclusivity, resource limits, or time gaps), is it possible to select a subset of k mutually compatible intervals?

56 Given an instance of the Travelling Salesman Problem (TSP), construct a polynomial-time transformation that produces an instance of the Hamiltonian Cycle problem such that:
The original TSP instance has a tour of total weight $\leq K$ if and only if the corresponding graph in the HAM-CYCLE instance contains a Hamiltonian cycle.
Definitions:

- **Travelling Salesman Problem (Decision version):** Given a complete weighted graph $G = (V, E)$ and an integer K, does there exist a tour (a cycle visiting every vertex exactly once and returning to the start) of total weight less than or equal to K?
- **Hamiltonian Cycle Problem:** Given an unweighted graph $G = (V, E)$, does there exist a cycle that visits every vertex exactly once?

57 Why does not Dijkstra work for the negative weights? Explain with the reasoning.

58 Given a 2D binary matrix of 0s and 1s, find the area of the largest square containing only 1s.
Example Input:

```
Matrix:
1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0
```

Expected Output: Area = 9 (3x3 square)

59 You are continuously inserting integers into a list. Return the kth largest element after each insertion.

60 House Robber Problem
You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, and you cannot rob two adjacent houses because it will alert the police. Given an array of non-negative integers representing the amount of money in each house, determine the maximum amount of money you can rob without alerting the police.

Example:
Input: nums [2, 7, 9, 3, 1]
Output: 12
Explanation: Rob house 1 ($2), skip house 2 ($7), rob house 3 ($9), and rob house 5 ($1) for a total of 2+9+1 = 12.

61 All-Pairs Shortest Paths (Floyd-Warshall Algorithm)
Given a directed graph with n vertices and weighted edges (some weights may be negative, but no negative cycles), find the shortest paths between all pairs of vertices.
Input:
Graph represented as an adjacency matrix:

```
[ [0 3 ∞ ∞]
  [∞ 0 1 ∞]
  [∞ ∞ 0 2]
  [2 ∞ ∞ 0] ]
```

Output:
Shortest-path matrix.

62 Dijkstra Algorithm on Graph with Negative Edge

- Run Dijkstra from A to C on the graph with edges A→B(2), B→C(–1), A→C(4). What shortest path and distance does it find?
- Does Dijkstra produce the correct shortest path here despite the negative edge?

63 Compress a File Header
Given a list of characters and their frequencies from a text file header, build the Huffman tree and output the code for each character.
Input: {a: 5, b: 9, c: 12, d: 13}
Task: Build Huffman codes.

64 Reduction: Partition to Job Schedule
Reduce Partition problem to Job schedule problem.

65 3-SAT to Vertex Cover (NP-Complete Reduction) What: A way to convert a 3-SAT logical formula into a graph so that finding a vertex cover corresponds exactly to finding a satisfying assignment for the formula.
How:

- For each variable x_i, create two nodes x_i and $\neg x_i$ connected by an edge (representing the choice of true/false).
- For each clause (3 literals), create a triangle of three nodes connected in a cycle.
- Connect each literal node in the clause triangle to its corresponding variable node.
- The vertex cover size $k = n + 2m$ where n = variables and m = clauses.

Why: Picking one vertex from each variable pair corresponds to a truth assignment. The clause triangles force the selection of vertices such that the clause is "covered" (satisfied). Thus, vertex cover solutions correspond to formula satisfiability.

66 SPFA (Shortest Path Faster Algorithm) What: An optimized version of Bellman-Ford algorithm to find shortest paths from a single source, especially for graphs with negative edges but no negative cycles.
How:

- Uses a queue to process nodes whose distance estimate can still be improved.
- Nodes are added to the queue only when their distance is updated, reducing unnecessary relaxations.
- Faster on average than Bellman-Ford because it avoids processing all edges every iteration.

When to use: Graphs with negative weights but no negative cycles, where Dijkstra's algorithm cannot be applied.

67 Binary Search + Patience Sorting for LIS (Longest Increasing Subsequence) What: Efficient method to find the length of the Longest Increasing Subsequence in $O(n \log n)$ time.
How:

- Maintain a list $tails$, where $tails[i]$ is the smallest possible tail value of an increasing subsequence of length $i + 1$.

- For each element x in the sequence, use binary search on $tails$ to find the position to place x (replace or append).
- The length of $tails$ at the end is the length of LIS.

Why: Patience sorting mimics the card game and the binary search efficiently maintains the subsequence tails.

68 Johnson's Algorithm (All Pairs Shortest Path) What: Computes shortest paths between all pairs of vertices in a weighted, directed graph which may contain negative edges but no negative cycles.
How:

- Add a new vertex connected to every other vertex with zero-weight edges.
- Run Bellman-Ford from this new vertex to find a potential function $h(v)$ to reweight edges and remove negative weights.
- Reweight edges using $h(u), h(v)$: $w'(u, v) = w(u, v) + h(u) - h(v)$, all non-negative now.
- Run Dijkstra's algorithm from each vertex on the reweighted graph.
- Correct final distances by reversing reweighting.

Why: Combines Bellman-Ford's ability to handle negative weights and Dijkstra's efficiency on non-negative weights.

69 Activity Selection Problem
What: Select the maximum number of activities that don't overlap given their start and finish times.
How:

- Sort activities by finish times.
- Iteratively pick the earliest finishing activity that starts after the last selected one.

Why: This greedy choice ensures optimality because picking the activity that frees the schedule earliest maximizes remaining time for others.