Time Complexity :

1. Topological Sort (Kahn’s Algorithm)

Time Complexity: O(V + E)
Steps:

- Initialize indegree[] of all vertices → O(V)
- Add all vertices with indegree 0 to queue → O(V)
- While queue not empty:
    - Pop front node, add to result → O(V)
    - Reduce indegree of neighbors, if any hits 0 → add to queue → O(E)
- Total: O(V + E)

2. Dijkstra’s Algorithm (Min Heap / Priority Queue)

Time Complexity: O((V + E) log V)
Steps:

- Initialize distance[] with ∞, set source = 0 → O(V)
- Use PriorityQueue (min-heap) for selecting min dist → O(log V) per op
- For each node, relax its neighbors → O(E log V)
- Total: O((V + E) log V)

3. Prim’s Algorithm (Min Heap)

Time Complexity: O((V + E) log V)
Steps:
⚪ Initialize visited[] and minHeap → O(V)
⚪ Start from any node, add all edges to heap
⚪ Pick min edge, if dest not visited, mark it and push its neighbors → O(E log V)
⚪ Repeat until all nodes visited
⚪ Total: O((V + E) log V)

4. QuickSort

Time Complexity:

- Best/Average: O(n log n)
- Worst (unlucky pivot): O(n^2)

Steps:
⚪ Choose pivot
⚪ Partition array (place pivot in correct position) → O(n)
⚪ Recursively quicksort left + right of pivot
⚪ Total: O(n log n) average

5. Gale-Shapley (Stable Marriage Problem)

Time Complexity: O(n^2)
Steps:
⚪ Each unengaged guy proposes to the next girl on his list → O(n^2) max
⚪ Each girl picks preferred among current + new proposal
⚪ Repeat until all guys are engaged
⚪ Total: O(n^2)

6. Inversion Count (Using Merge Sort)

Time Complexity: O(n log n)
Steps:

- Modify merge sort
- In merge step: if arr[i] > arr[j], count += mid - i
- Recursive divide + merge
- Total: O(n log n)

7. Kruskal’s Algorithm (MST using DSU)

Time Complexity: O(E log E)
Steps:

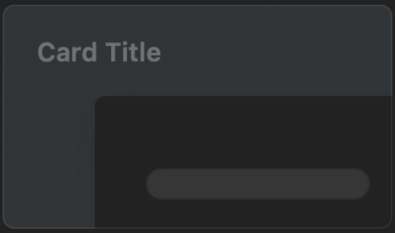
- Sort all edges by weight → O(E log E)
- Initialize DSU (Disjoint Set Union) → O(E)
- For each edge, if endpoints in diff sets → add to MST, union sets
- Stop when MST has (V - 1) edges
- Total: O(E log E)

8. Merge Sort

Time Complexity: O(n log n)

Steps:

Preview of the page will be displayed here. Click to add content.



Card Title

- Divide array in halves → O(log n)
- Recursively sort left + right
- Merge the two sorted halves → O(n)
- Total: O(n log n)