Sorting:

- Sorting by Ranking: $W(n) = B(n) = A(n) = O(n(n-1))$ (Oblivious Algorithm)
- Sorting by Swapping
  - Bubble Sort: $W(n) = O(\frac{n^2}{2})$, $B(n) = O(n)$, $S(n) = O(1)$ (Stable, Adaptive Algorithm)
  - Odd-Even Exchange Sort: $W(n) = O(n^2)$, $B(n) = O(n)$, $S(n) = O(1)$ (Stable, Adaptive Algorithm)
- Sorting by Insertion (Online Algorithms)
  - Linear Insertion Sort: $W(n) = O(\frac{n^2}{2})$, $B(n) = O(n)$, $S(n) = O(1)$ (Stable, Adaptive Algorithm)
  - Binary Insertion Sort: $W(n) = O(n \log n) + O(n^2)$ (Stable, Flexible Algorithm)
  - Tree Sort: $W(n) = O(n^2)$, $B(n) = A(n) = O(n \log n) + O(n)$, $S(n) = O(n)$
- Sorting by Selection (Offline Algorithms)
  - Selection Sort: $W(n) = B(n) = O(n^2)$, $S(n) = O(1)$ (Unstable, Oblivious Algorithm)
  - Tournament Sort: $W(n) = B(n) = A(n) = O(n \log n)$, $S(n) = O(n)$ (Unstable)
  - Heap Sort: $W(n) = A(n) = O(n \log n)$, $S(n) = O(1)$ **[W(n) derivation]**
  - Priority Queue **[Textbook]**
- Sorting by Merging
  - Mergesort: $W(n) = B(n) = O(n \log n)$, $S(n) = O(n)$ **[W(n) derivation]** (Stable, Oblivious)
- Sorting by Splitting
  - Quicksort: $W(n) = O(n^2)$, $B(n) = A(n) = O(n \log n)$ **[A(n) derivation]**
- Shell Sort: W(n) varies based on h value selection **[W(n) derivation]**

Linear Sorting:

- Counting Sort: $W(n) = O(n)$, if range is $O(n)$, $S(n) = O(n+m)$, where m is range or max value
- Bucket Sort: $W(n) = O(n \log n)$, $A(n) = O(n)$, $S(n) = O(nk)$ or $O(n+k)$ for linked list **[A(n) derivation]**
- Bucket Sort with Insertion sort: $W(n) = O(n^2)$, $A(n) = O(n)$ for $k = O(n)$ **[A(n) derivation]**
- Radix Sort: $W(n) = A(n) = O(n)$

Minimum Spanning Tree (MST):

- Prim-Dijkstra's MST algorithm basic idea: Working, $W(n) = O(n^3)$
- Prim-Dijkstra's MST algorithm using NEAR array: Algorithm, Proof, $W(n) = O(n^2)$ (All cases)
- Prim-Dijkstra's MST algorithm using min-heap: $W(n) = O(n \log n + m \log n)$ (All cases)
- Prim-Dijkstra's MST algorithm using Fibonacci heap: $W(n) = O(n \log n + m)$ (All cases)
- Kruskal's MST algorithm basic idea: Algorithm, Proof
- Kruskal's MST algorithm using Find, Union: Working, $W(n) = m \log n$, $m = n$ (sparse), $\frac{n^2}{\log n}$ (avg), n^2 (dense)
- Kruskal's MST: W-Union and C-Find functions (3 lemmas)

One-to-All Shortest Path:

- Djikstra's shortest path Algorithm: Working, $W(n) = n$*Extract-Min + m*Decrease-Key, Proof of correctness
- Bellman Ford Algorithm: Working, Proof of correctness, $W(n) = O(nm)$
- Topological Sorting (DAG and DFS): Working, $W(n) = O(n+m)$

All-to-All Shortest Path:

- Bellman Ford 'n' times: $W(n) = O(n^2m)$, For dense graph, $W(n) = O(n^4)$
- Djikstra's 'n' times: $W(n) = n^2$*Extract-Min + nm*Decrease-Key
- Dynamic Programming: Basic Idea, Working
- Floyd Warshall Algorithm: $W(n) = O(n^3)$
- Transitive Closure: Existence of paths between all pairs of vertices

Definitions:

- P: Class of decision problems which have polynomially bounded algorithms, $W(n) \leq p(n)$
- Non-deterministic Algorithms: Guess a solution and verify it | Demo
  - Problem: π, Instance: I, Domain: D_π, Yes-Instances: Y_π, No-Instances: $D_\pi - Y_\pi$
- NP: Class of decision problems which have polynomially bounded non-deterministic algorithms, i.e., guesses are polynomial time verifiable (Non-deterministic Polynomial)
- Polynomial reduction: It takes polynomial time to convert every problem into Satisfiability problem | Demo

- NP-Complete: NP Problem to which other NP problems can be transformed to (Ex: Satisfiability), i.e., Hardest problems in NP
- NP-Hard: Problems that are at least as hard as NP-Complete problems
- Cook's Theorem: Satisfiability is a NP-C problem. If Satisfiability becomes P, NP will become P | Demo
- Approximation Algorithms: Polynomial time algorithm for NP-C problems which don't guarantee optimal solution but will be close to optimal

Problems:

- Coloring: Minimum number of k-colors required to color the vertices of graph such that adjacent colors differ
  - Optimization problem, Decision problem, Both (Chromatic Number)
  - Vizing's Theorem
- Clique: Maximum possible size of complete subgraph in a given graph
  - Optimization problem, Decision problem, Both (Clique Search)
  - Peterson's Graph
  - Kuratowski's Theorem
- Satisfiability: Possibility of some integer values satisfying the given equations/statements
  - SAT, 2-SAT, 3-SAT
- Bin-Packing: n objects to be placed in bin of capacity L each. Minimum number of bins needed | Demo
  - Decision: NP-C, Optimization: NP-H
  - Approximation: First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD), Best Fit Decreasing (BFD)
- Cook's Theorem: Garey and Johnson Proof