Tables

Q1) Difference between Prim's Algorithm and Kruskal's Algorithm.

Factor	**Prim's Algorithm**	**Kruskal's Algorithm**
Approach	Grows MST from **one starting node**	Selects **smallest edges globally**
Data Structure	Uses **Min-Heap / Priority Queue**	Uses **Union–Find**
Best For	**Dense graphs**	**Sparse graphs**
Starting Point	Needs a **start vertex**	**No start** needed
Cycle Handling	Cycles **don't occur naturally**	Uses Union–Find to **avoid cycles**
Edge Choice	Chooses **cheapest edge** from current tree	Chooses **cheapest edge** overall
Working Style	**Node-based**	**Edge-based**
Complexity	O(E log V)	O(E log V)
MST Formation	Tree grows **continuously**	Components **merge gradually**

Q2) Difference between greedy approach and dynamic programming.

Factor	**Greedy Approach**	**Dynamic Programming**
Strategy	Makes the **locally optimal** choice at each step	Solves **subproblems** and combines results for global optimum
Optimality	May **not always** give the global optimum	**Always** gives global optimum if problem has overlapping subproblems & optimal substructure
Overlapping Subproblems	**Not required**	**Required**

Factor	Greedy Approach	Dynamic Programming
Backtracking	**No backtracking**	May **require backtracking** for reconstruction
Example	Kruskal's and Prim's algorithm	Matrix Chain Multiplication, Floyd-Warshall
Complexity	Usually **faster and simpler**	**More complex**, but accurate

---

Q3) Difference between naïve string matching and Rabin Karp.

Factor	Naïve String Matching	Rabin-Karp Algorithm
Approach	Compares pattern at every position	Uses **hashing** to compare pattern with text substrings
Time Complexity (Average)	**O(nm)**	**O(n + m)** average
Time Complexity (Worst)	**O(nm)**	**O(nm)** (due to hash collisions)
Efficiency	**Less efficient** for large texts	**More efficient** for multiple pattern matching
Use of Hash	**Not used**	**Used** for quick comparison
Example Use	Simple search in small data	Searching multiple patterns or plagiarism detection

---

Q4) Differentiate between NP-Hard and NP-complete problem.

Factor	NP-Hard	NP-Complete
Definition	Problems **at least as hard as NP problems**	Problems that are **both in NP and NP-Hard**

Factor	**NP-Hard**	**NP-Complete**
Verification	**May not** be verifiable in polynomial time	**Can** be verified in polynomial time
Solution	Not required to have polynomial-time verifier	Has polynomial-time verifier
Relation	**Superset** of NP-Complete	**Subset** of NP-Hard
Example	Halting Problem	Traveling Salesman, 3-SAT
Solvability	**May not** be solvable at all	**Solvable**, but not in polynomial time (so far)