Sequential search

=> Searching for a name or contact in a phone book or contact list

=> Searching for a book in a library catalog or a specific aisle in a bookstore

=> Checking whether a product is in stock at a grocery store or online retailer

=> Scanning through an email inbox to find a specific message or sender

=> Looking for a specific word or phrase in a document or webpage using a search function

Binary search

=> Searching for a word in a dictionary or a phonebook

=> Finding a specific time slot or appointment in a sorted calendar system

=> Locating a specific record or entry in a sorted database or spreadsheet

=> Finding a phone number in a sorted phone list

=> Searching for a name or contact in a sorted contact list

Quick sort

=> Sorting a large list of phone numbers or addresses in a phonebook or address book

=> Sorting a large collection of names or other text data in alphabetical order

=> Sorting a list of transactions or financial data by date or amount

=> Sorting a list of items in an e-commerce website by price or popularity

=> Sorting a list of search results on a search engine by relevance or popularity

Merge sort

=> Sorting a large amount of data, such as in a database or spreadsheet

=> Sorting and merging multiple files or data streams, such as in a data backup or synchronization process

=> Sorting and merging multiple arrays of data, such as in a parallel computing or distributed systems application

=> Sorting and merging multiple lists of search results or recommendations, such as in a web search engine or e-commerce website.

=> Merging and sorting multiple sets of data, such as in the process of deduplication of data

Insertion sort

=> Sorting a deck of playing cards by hand

=> Sorting a small list of names or addresses in alphabetical order

=> Sorting a list of numbers or dates by their value or time

=> Organizing a playlist of songs in a music player based on title or artist

=> Sorting a stack of papers or files into a specific order

Selection sort

=> Sorting a deck of cards by repeatedly selecting the smallest card and swapping it with the leftmost unsorted card

=> Sorting a list of prices by repeatedly selecting the cheapest item and moving it to the beginning of the list

=> Sorting a list of names alphabetically by repeatedly selecting the name that comes first alphabetically and swapping it with the leftmost unsorted name

=> Sorting a list of grades by repeatedly selecting the lowest grade and moving it to the beginning of the list

=> Sorting a list of files by file size by repeatedly selecting the smallest file and moving it to the beginning of the lists

Prims

=> Finding the shortest or cheapest route for a telecommunication network

=> Road network planning

=> Designing efficient airline routes

=> Resource allocation

=> electrical power distribution

Kruskal

1.Designing efficient transportation networks: Kruskal's algorithm can be used to find the minimum cost of building roads, railways, and other transportation infrastructure to connect cities or towns.

2.Power grid design: Kruskal's algorithm can be used to optimize the design of a power grid by minimizing the total length of power lines needed to connect all the substations.

3.Cable TV network design: Kruskal's algorithm can be used to determine the most cost-effective way to connect all the households in a region to a cable TV network.

4.Computer network design: Kruskal's algorithm can be used to find the most efficient way to connect multiple computers in a network by minimizing the total cost of cables or wireless connections needed.

5.DNA sequencing: Kruskal's algorithm can be used to reconstruct the DNA sequence of a genome by assembling the short DNA fragments obtained from sequencing machines.

Djikstra

=> Finding the shortest path between two locations on a map or GPS navigation system.

=> Finding the shortest route for a vehicle or delivery truck to travel between multiple destinations.

=> Calculating the shortest path for a drone or robot to navigate through an environment, such as a warehouse or factory.

=> Planning network routing in telecommunications and computer networking, such as finding the shortest path between two nodes on a network.

=> Optimizing the layout of printed circuit boards (PCBs) in electronics design, where the distance between components needs to be minimized.

Knapsack using DP

=> Resource allocation: In a business setting, a company may need to allocate resources such as budget, personnel, and equipment to various projects or tasks

=> Portfolio optimization: The knapsack problem can be used to optimize investment portfolios by selecting the most valuable investments that meet certain constraints, such as a limited budget or a maximum risk level.

=> Cutting stock problem: The cutting stock problem involves cutting large sheets of material into smaller pieces to minimize waste.

=> DNA sequencing: In bioinformatics, the knapsack problem can be used to determine the optimal sequence alignment of DNA or protein sequences, which can aid in understanding genetic and biological processes.

=> Scheduling: The knapsack problem can be used to optimize scheduling in various industries, such as transportation, manufacturing, and logistics.

String edit using DP

=> Spell checking and correction in word processors and text editors

=> Voice recognition and speech-to-text conversion in digital assistants and communication devices

=> Image recognition and object detection in computer vision

=> Text mining and natural language processing in search engines and online advertising

=> Comparison of audio and video signals for music and video recognition

Backtracking

=> Solving Sudoku puzzles: Backtracking is commonly used to solve Sudoku puzzles by exploring all possible solutions and backtracking when an invalid move is made.

=> Route planning: Backtracking can be used to find the shortest or fastest route between two points on a map by exploring all possible paths and backtracking when a dead end is reached.

=> Generating mazes: Backtracking can be used to generate mazes by exploring all possible paths and backtracking when a dead end is reached.

=> Solving the traveling salesman problem: Backtracking can be used to solve the traveling salesman problem, which involves finding the shortest route that visits all cities in a given list exactly once and returns to the starting city.

=> Planning chess moves: Backtracking can be used to plan chess moves by exploring all possible moves and backtracking when a move is found to be invalid or leads to a disadvantage.