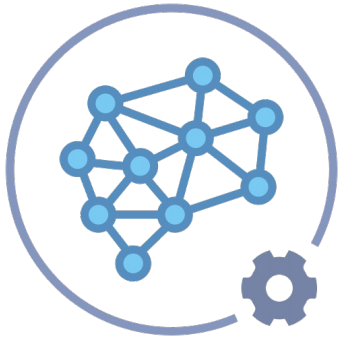
Algorithms

Tushar B. Kute,
http://tusharkute.com



What is algorithm?

- An algorithm is a step-by-step procedure for solving a problem.
- It is a sequence of instructions that can be carried out by a computer or by a human.
- Algorithms are used in all aspects of computer science, from sorting and searching to machine learning and artificial intelligence.

What is algorithm?

- They are precise: The steps in an algorithm must be defined in a way that is unambiguous and repeatable.
- They are complete: An algorithm must include all of the steps necessary to solve the problem.
- They are effective: An algorithm must be efficient and effective in solving the problem.

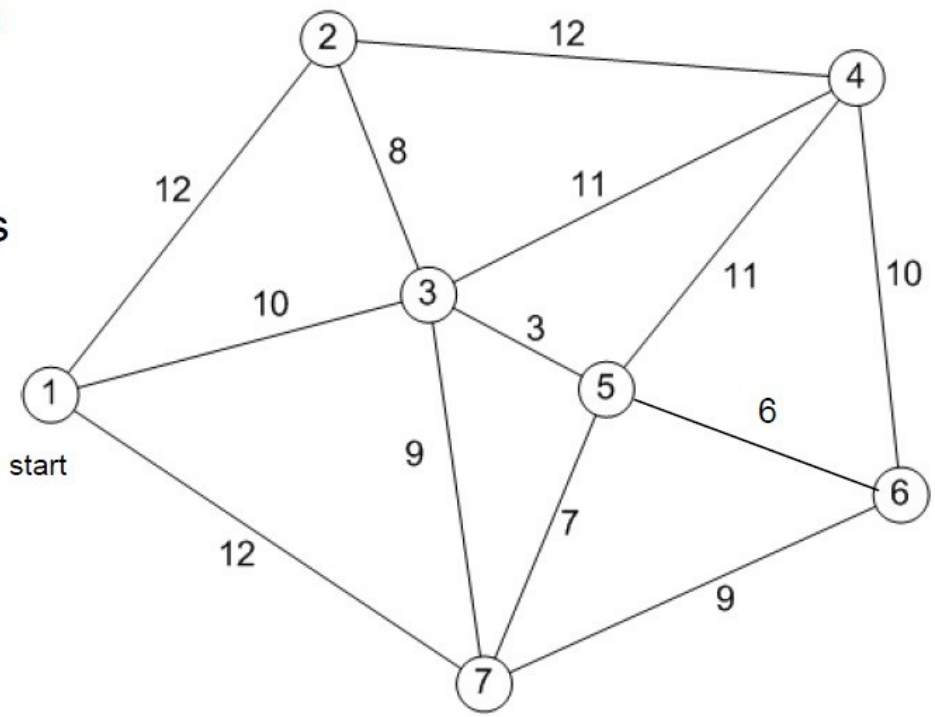
miTu skillologies
tusharkute.com

Algorithm?: Classes

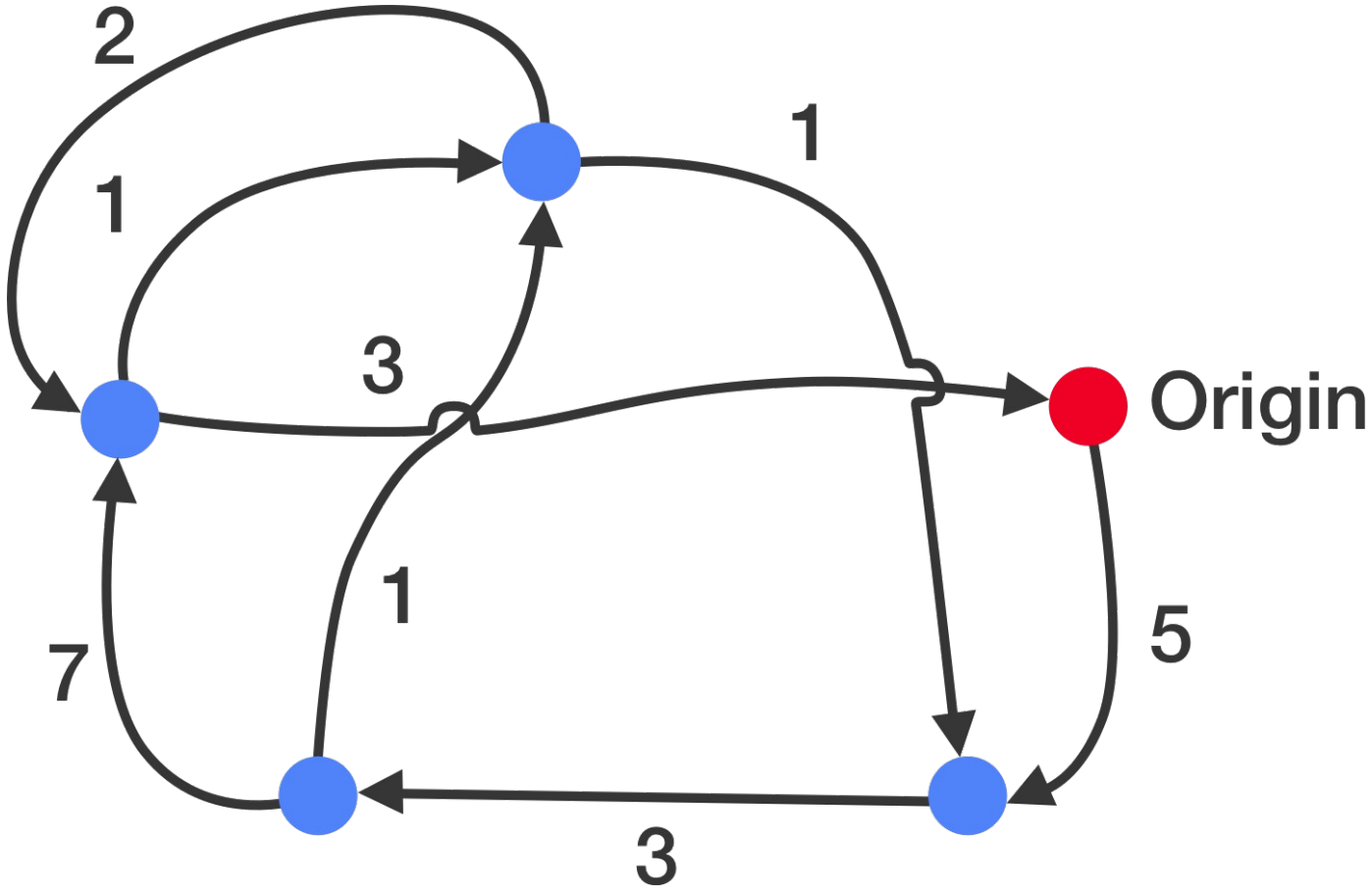
- Divide and Conquer Algorithm
- Greedy algorithm
- Dynamic Programming algorithm
- Brute force algorithm
- Backtracking algorithms
- Branch-and-bound algorithms
- Stochastic algorithms

Traveling Salesman Problem

- The salesman must travel to all cities once before returning home
- The distance between each city is given, and is assumed to be the same in both directions
- Objective - Minimize the total distance to be travelled

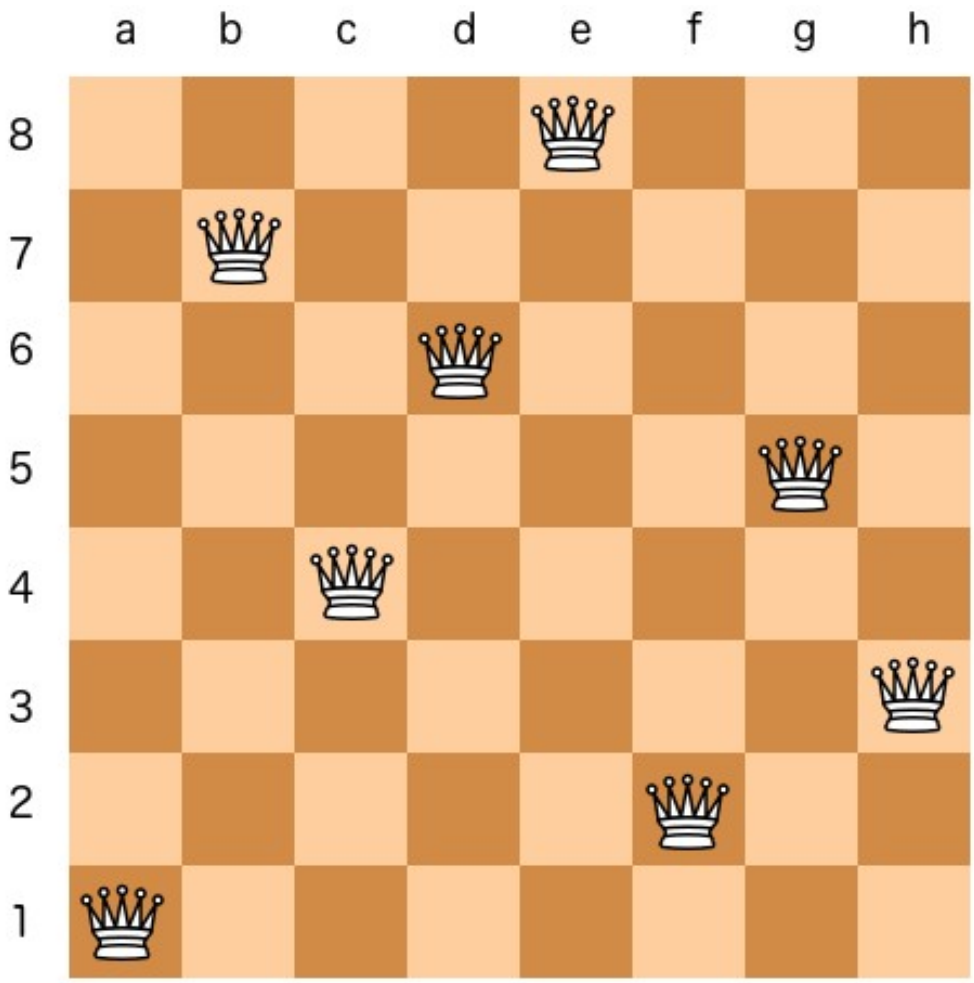


Traveling Salesman Problem



Knapsack Problem

8-Queens Problem



How to write efficient Algorithm?

- Time complexity: The time complexity of an algorithm is the amount of time it takes to run as a function of the size of the input. It is important to choose an algorithm with a low time complexity, especially if the input is large.
- Space complexity: The space complexity of an algorithm is the amount of memory it requires to run as a function of the size of the input. It is important to choose an algorithm with a low space complexity, especially if the memory is limited.

How to write efficient Algorithm?

- Correctness: The algorithm must be correct, meaning that it must always produce the correct output for any given input.
- Readability: The algorithm should be readable and easy to understand. This makes it easier to debug and maintain the algorithm.

Efficient Algorithm: Examples

- Quicksort: Quicksort is a divide and conquer algorithm for sorting a list of numbers. It has an average time complexity of O(n log n) and a space complexity of O(log n).
- Merge sort: Merge sort is another divide and conquer algorithm for sorting a list of numbers. It has a time complexity of O(n log n) in all cases.

Efficient Algorithm: Examples

- Dijkstra's algorithm: Dijkstra's algorithm is a greedy algorithm for finding the shortest path in a graph. It has a time complexity of O(|E| + |V| log |V|), where |E| is the number of edges in the graph and |V| is the number of vertices in the graph.
- Prim's algorithm: Prim's algorithm is a greedy algorithm for finding the minimum spanning tree of a graph. It has a time complexity of O(|E| + |V| log |V|).

Algorithm Design Techniques

- Divide and Conquer Algorithm
- Greedy algorithm
- Dynamic Programming algorithm
- Brute force algorithm
- Backtracking algorithms
- Stochastic algorithms

tusharkute.com

Algorithm Analysis

- Asymptotic analysis: Asymptotic analysis considers the behavior of the algorithm as the input size approaches infinity.
  - This is the most common type of algorithm analysis, and it is used to compare the efficiency of different algorithms.
- Average-case analysis: Average-case analysis considers the behavior of the algorithm over all possible inputs.
  - This type of analysis is more difficult to perform than asymptotic analysis, but it can provide more accurate estimates of the performance of the algorithm in practice.

miTu skillologies
tusharkute.com

Asymptotic Analysis

- Asymptotic analysis is a technique for evaluating the performance of algorithms as the input size approaches infinity.
- It is the most common type of algorithm analysis, and it is used to compare the efficiency of different algorithms.

miTu skillologies
tusharkute.com

Asymptotic Analysis

- Big O notation: Big O notation describes the upper bound on the growth rate of an algorithm as the input size approaches infinity.
- Omega notation: Omega notation describes the lower bound on the growth rate of an algorithm as the input size approaches infinity.
- Theta notation: Theta notation describes the tight bound on the growth rate of an algorithm as the input size approaches infinity.

Divide and Conquer

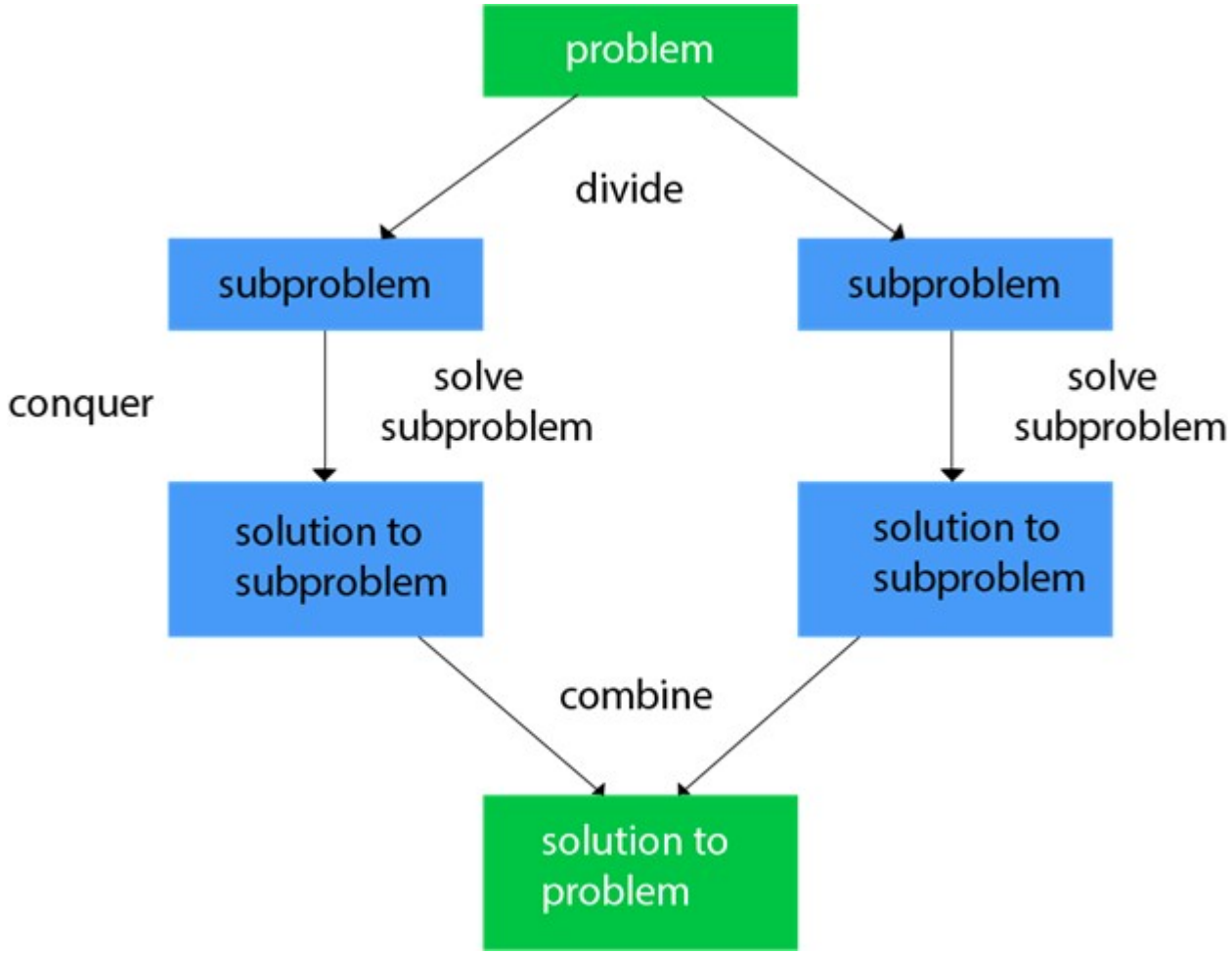
- A divide-and-conquer algorithm is an algorithm design paradigm that works by recursively breaking down a problem into smaller subproblems, solving the subproblems, and then combining the solutions to the subproblems to solve the original problem.
- This technique is often used for sorting and searching algorithms.

Divide and Conquer

- Here are the steps involved in a divide-and-conquer algorithm:
  - Divide: Divide the problem into two or more smaller subproblems.
  - Conquer: Solve the subproblems recursively.
  - Combine: Combine the solutions to the subproblems to solve the original problem.

tusharkute.com

Divide and Conquer

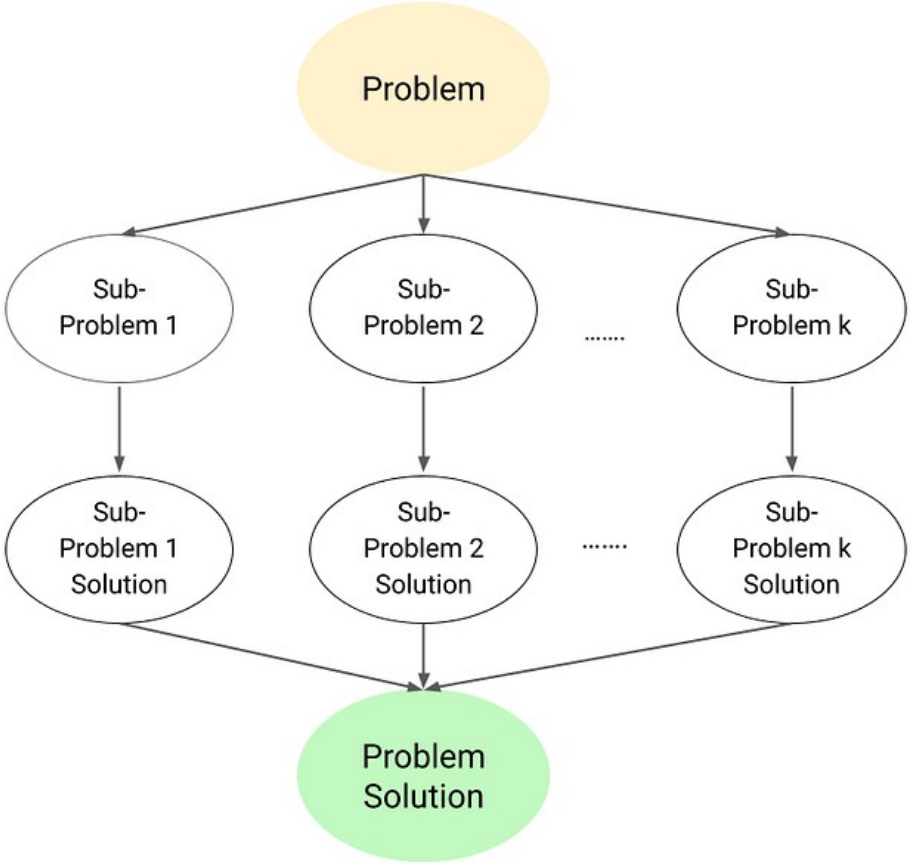


Divide and Conquer

Divide
Dividing the problem into smaller sub-problems

Conquer
Solving each sub-problems recursively

Combine
Combining sub-problem solutions to build the original problem solution



Divide and Conquer: Benefits

- Here are some of the benefits of divide-and-conquer algorithms:
  - Efficiency: Divide-and-conquer algorithms are often very efficient, especially for problems with large inputs.
  - Simplicity: Divide-and-conquer algorithms are typically easy to design and implement.
  - Parallelization: Divide-and-conquer algorithms can be easily parallelized, which can lead to significant performance improvements for large inputs.

Divide and Conquer: Drawbacks

- Here are some of the drawbacks of divide-and-conquer algorithms:
  - Overhead: Divide-and-conquer algorithms can have some overhead associated with dividing the problem into subproblems and combining the solutions to the subproblems.
  - Stack space: Divide-and-conquer algorithms can use a lot of stack space, especially for problems with deep recursion.

Divide and Conquer: Examples

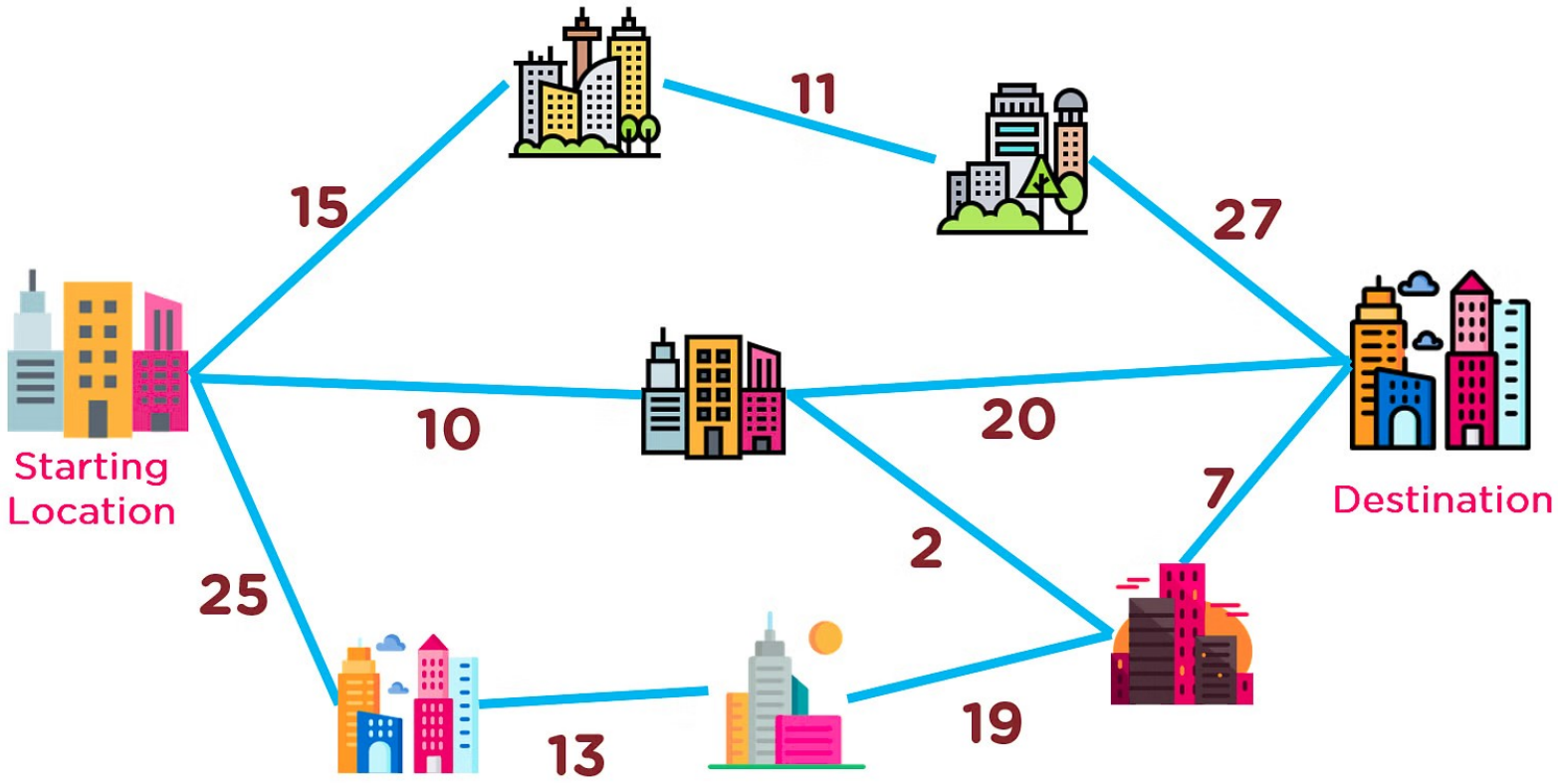
- Quicksort
- Mergesort
- Binary Search
- Strassen's algorithm

Greedy Algorithm

- A greedy algorithm is a type of algorithm that makes the locally optimal choice at each step in the hope of finding a globally optimal solution.
- Greedy algorithms are often very efficient, but they may not always find the optimal solution.
- Greedy algorithms are typically used for optimization problems, such as finding the shortest path in a graph or the maximum spanning tree of a graph.
- They can also be used for other types of problems, such as scheduling and knapsack problems.

miTu skillologies
tusharkute.com

Greedy Algorithm



Greedy Algorithm

- Advantages:
  - Often very efficient
  - Can be used to find good solutions to problems even when the optimal solution is difficult to find
- Disadvantages:
  - May not always find the optimal solution
  - Can be sensitive to the order in which the choices are made

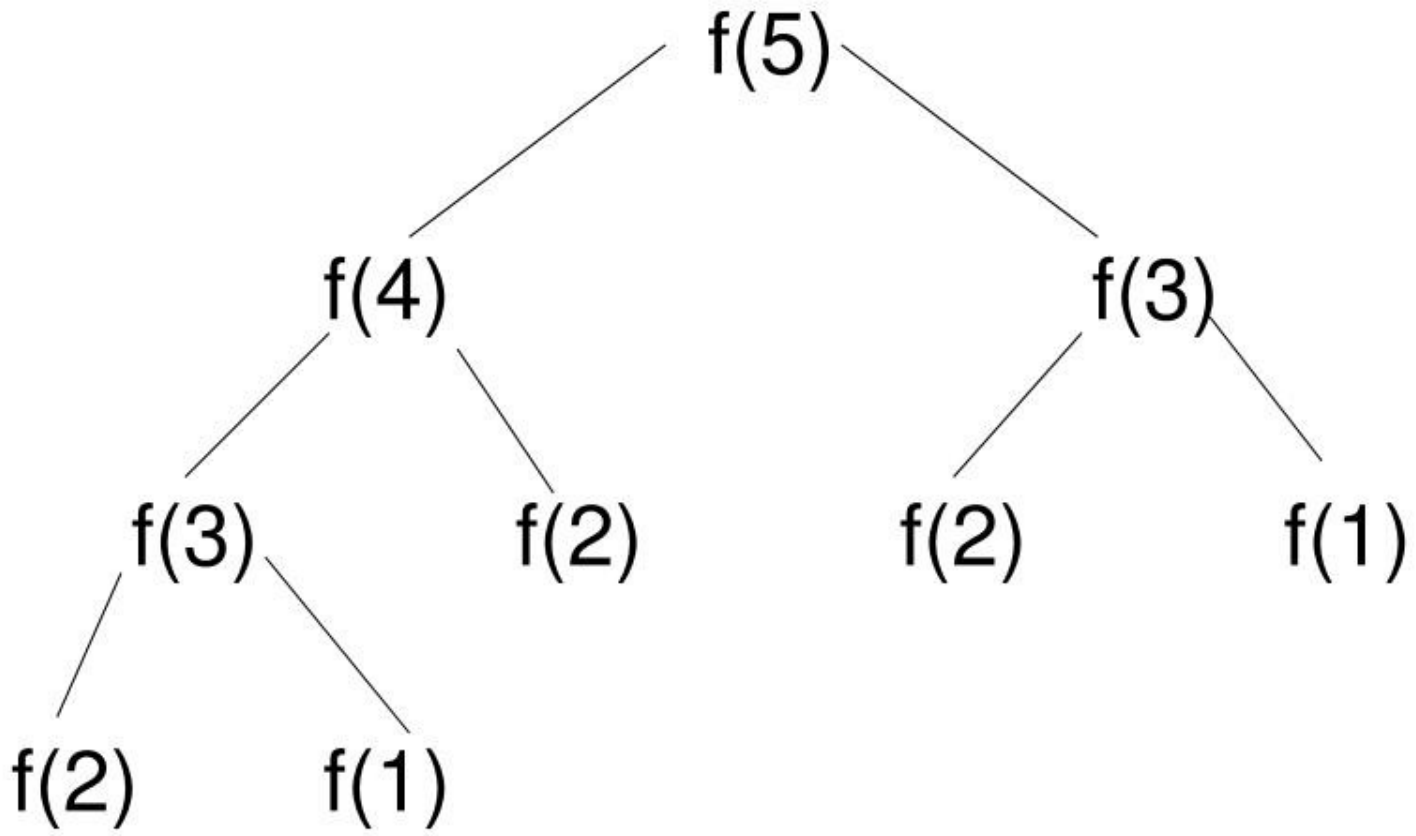
Greedy Algorithm: Examples

- Dijkstra's algorithm: An algorithm that finds the shortest path between a single source vertex and all other vertices in a weighted graph.
- Prim's algorithm: An algorithm that finds the minimum spanning tree of a connected undirected graph.
- Kruskal's algorithm: An algorithm that finds the minimum spanning tree of a connected undirected graph.

Dynamic Programming

- Dynamic programming algorithms are a type of algorithm that store the results of intermediate computations to avoid recomputing them.
- This can lead to significant speedups for problems with overlapping subproblems.

Dynamic Programming



Dynamic Programming

- Longest common subsequence (LCS): An algorithm that finds the longest common subsequence of two strings.
- Knapsack problem: An algorithm that finds the subset of items with the greatest total value that can be fitted into a knapsack of a given capacity.
- Floyd-Warshall algorithm: An algorithm that finds the shortest paths between all pairs of vertices in a weighted graph.
- Bellman-Ford algorithm: An algorithm that finds the shortest paths between all pairs of vertices in a weighted graph, even if the graph contains negative edge weights.

miTu skillologies
tusharkute.com

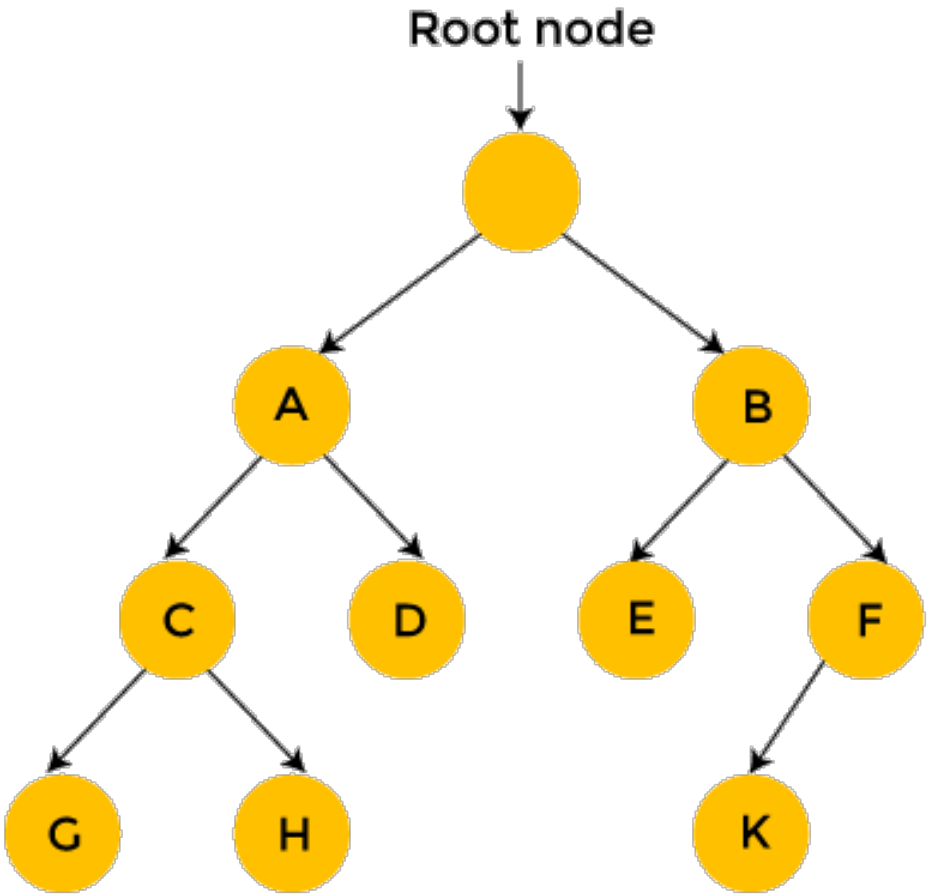
Brute Force Algorithm

- A brute force algorithm is an algorithm that solves a problem by trying all possible solutions.
- It is the simplest type of algorithm, but it is also the least efficient.
- Brute force algorithms are often used to solve small problems or problems where there are no other known algorithms.

Brute Force Algorithm

- Searching for a specific element in an array.
- Finding the shortest path between two points in a graph
- Solving the knapsack problem

Brute Force Algorithm



Brute Force Algorithm

- Advantages:
  - Simple to design and implement
  - Guaranteed to find a solution if one exists
- Disadvantages:
  - Very inefficient for large problems
  - May require a lot of memory

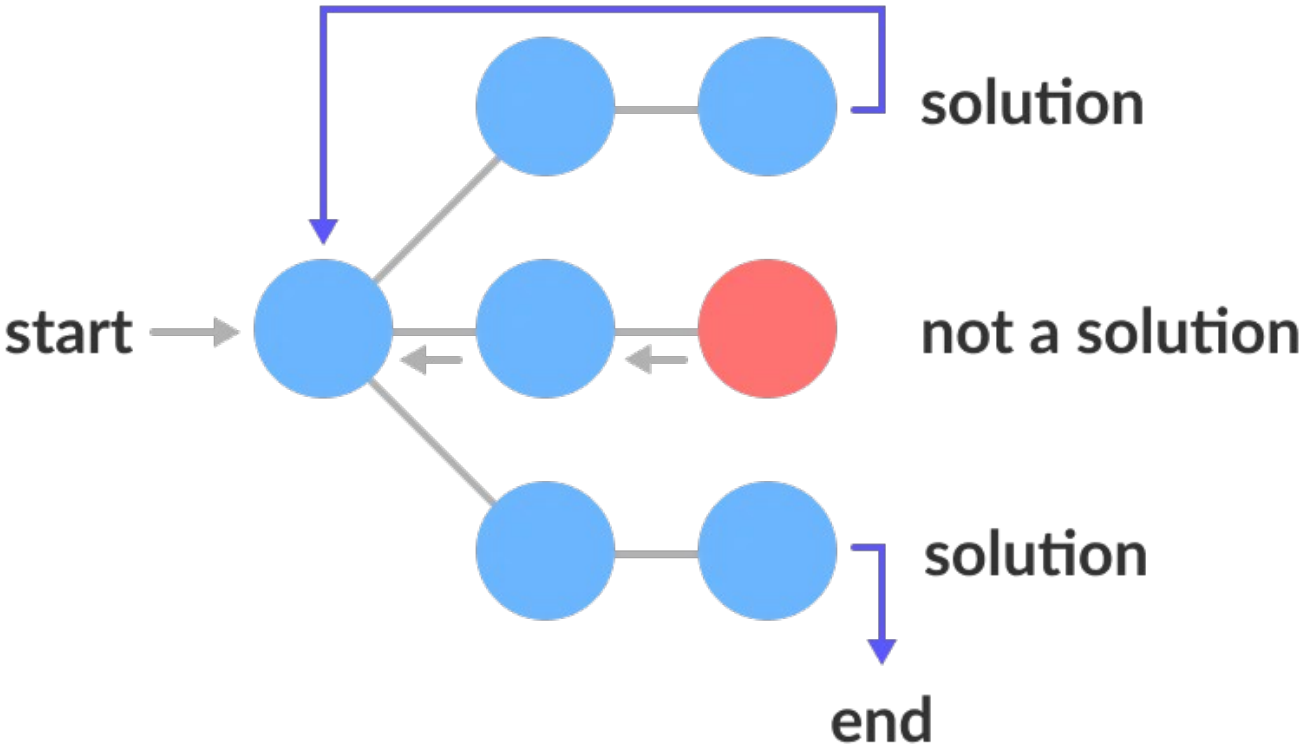
miTu skillologies
tusharkute.com

Backtracking Algorithms

- Backtracking algorithms are a type of algorithm that solves a problem by recursively exploring all possible solutions.
- When a dead end is reached, the algorithm backtracks and tries a different solution. Backtracking algorithms are often used to solve combinatorial problems, such as the n-queens problem and the traveling salesman problem.

tusharkute.com

Backtracking Algorithms



Backtracking Algorithms

- The n-queens problem
- The traveling salesman problem
- The 8-puzzle
- The Sudoku puzzle
- The cryptarithmetic puzzle
- The Hamiltonian circuit problem
- The clique problem
- The vertex cover problem
- The graph coloring problem
- The scheduling problem
- The resource allocation problem

Backtracking Algorithms

- Maze solving: An algorithm that finds a path from the start to the end of a maze.
- Subsets: An algorithm that finds all subsets of a given set.
- Permutations: An algorithm that finds all permutations of a given set.
- Anagrams: An algorithm that finds all anagrams of a given word.

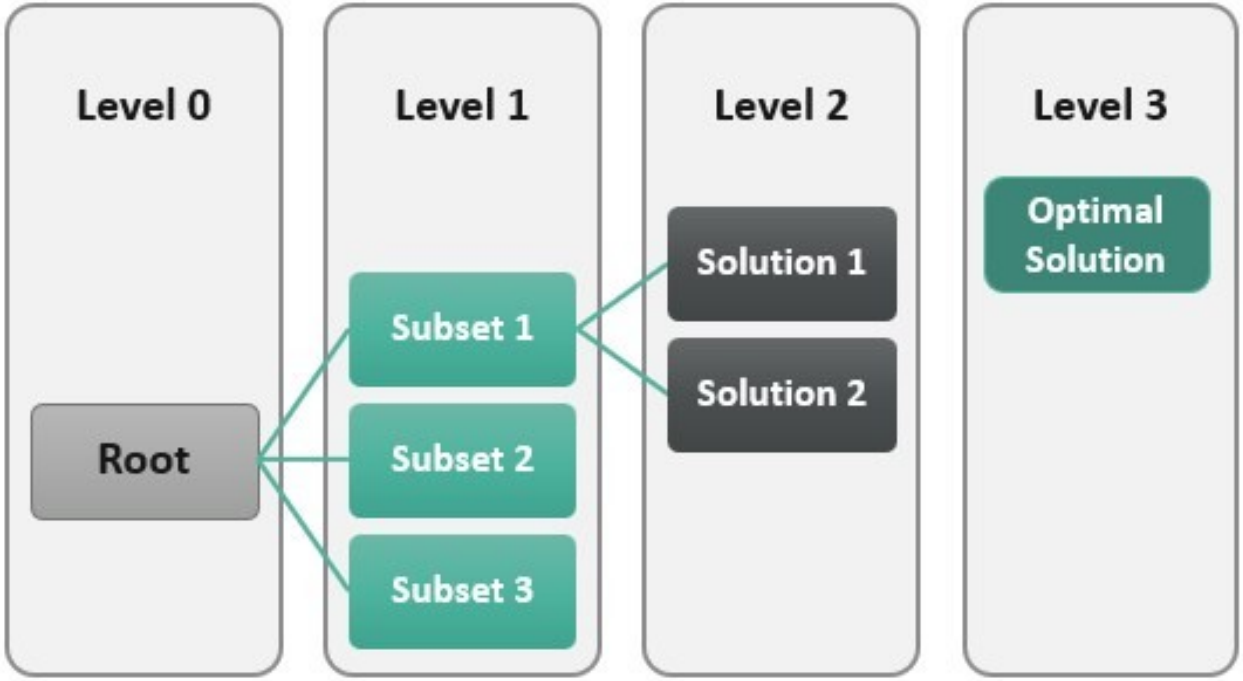
Backtracking Algorithms

- Advantages:
  - Guaranteed to find a solution if one exists
  - Can be used to solve a wide variety of problems
- Disadvantages:
  - Can be inefficient for large problems
  - May require a lot of memory

Branch-and-bound algorithms

- Branch-and-bound algorithms are a type of optimization algorithm that solves problems by recursively partitioning the search space into smaller and smaller subproblems, and then bounding the optimal solution of each subproblem.
- The algorithm then explores the subproblems in a systematic way, pruning subproblems whose bounds cannot contain the optimal solution.

Branch-and-bound algorithms



Branch-and-bound algorithms

- Branch-and-bound algorithms are often used to solve combinatorial problems, such as the traveling salesman problem, the knapsack problem, and the maximum cut problem.
- They can also be used to solve continuous optimization problems, such as linear programming and nonlinear programming problems.

Branch-and-bound algorithms

- Traveling salesman problem (TSP)
- Knapsack problem

tusharkute.com

Branch-and-bound algorithms

- Advantages:
  - Can find optimal solutions to problems
  - Can be used to solve a wide variety of problems, including combinatorial and continuous optimization problems
- Disadvantages:
  - Can be computationally expensive for problems with a large number of possible solutions
  - Can be difficult to implement

Stochastic Algorithms

- Stochastic algorithms are algorithms that use randomness to make decisions.
- These algorithms are often used to solve problems that are too difficult or time-consuming to solve using deterministic algorithms.

Stochastic Algorithms

- Here are some examples of stochastic algorithms:
  - Simulated annealing: An algorithm that finds the global minimum of a function by repeatedly heating and cooling the function.
  - Genetic algorithms: An algorithm that finds the optimal solution to a problem by repeatedly evolving a population of solutions.

miTu skillologies
tusharkute.com

Stochastic Algorithms

- Particle swarm optimization: An algorithm that finds the optimal solution to a problem by repeatedly moving a swarm of particles through the search space.
- Tabu search: An algorithm that finds the optimal solution to a problem by exploring the search space while avoiding getting stuck in local optima.
- Monte Carlo methods: A class of algorithms that use random sampling to solve problems.

Stochastic Algorithms: Benefits

- Here are some of the benefits of using stochastic algorithms:
  - Can find good solutions to problems that are too difficult or time-consuming to solve using deterministic algorithms
  - Can be parallelized to improve performance
  - Can be used to solve a wide variety of problems

miTu skillologies
tusharkute.com

Stochastic Algorithms: Benefits

- Here are some of the drawbacks of using stochastic algorithms:
  - Not guaranteed to find the optimal solution
  - Sensitive to their parameters
  - Can be difficult to implement

Stochastic Algorithms: Examples

- Financial trading: Stochastic algorithms can be used to develop trading strategies that can generate profits in financial markets.
- Machine learning: Stochastic algorithms are often used in machine learning to train models that can make predictions or classify data.
- Robotics: Stochastic algorithms can be used to develop control algorithms for robots that can navigate and interact with the world.
- Optimization: Stochastic algorithms can be used to solve a wide variety of optimization problems, such as routing problems, scheduling problems, and knapsack problems.

tusharkute.com

Algorithm : Analysis

- Priori Analysis:
  - Here, priori analysis is the theoretical analysis of an algorithm which is done before implementing the algorithm. Various factors can be considered before implementing the algorithm like processor speed, which has no effect on the implementation part.
- Posterior Analysis:
  - Here, posterior analysis is a practical analysis of an algorithm. The practical analysis is achieved by implementing the algorithm using any programming language. This analysis basically evaluate that how much running time and space taken by the algorithm.

miTu skillologies
tusharkute.com

Time Complexity

- The time complexity of an algorithm is the amount of time required to complete the execution.
- The time complexity of an algorithm is denoted by the big O notation. Here, big O notation is the asymptotic notation to represent the time complexity.
- The time complexity is mainly calculated by counting the number of steps to finish the execution.

Space Complexity

- Space complexity: An algorithm's space complexity is the amount of space required to solve a problem and produce an output.
- Similar to the time complexity, space complexity is also expressed in big O notation.

Applications of Data Structures

- Operating systems
  - Linked List, Stack, Queue
- Databases
  - Trees
- Compilers
  - Symbol Tables
- Graphics
  - Linked List

Applications of Data Structures

- Networking
  - Trees and Graphs
- Web Browsers
  - Stack
- Search Engines
  - Trees and many
- Social Network
  - Trees and Graphs

Thank you

This presentation is created using LibreOffice Impress 7.4.1.2, can be used freely as per GNU General Public License

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com