Link

LMNs- Algorithms

Read Discuss Courses

LMNs– Algorithms

Analyze an algorithm

1) **Worst Case Analysis (Usually Done)**:In the worst case analysis, we calculate upper bound on running time of an algorithm by considering worst case (a situation where algorithm takes maximum time)
2) **Average Case Analysis (Sometimes done)** :In average case analysis, we take all possible inputs and calculate computing time for all of the inputs.
3) **Best Case Analysis (Bogus)** :In the best case analysis, we calculate lower bound on running time of an algorithm.

Asymptotic Notations

- **Θ Notation:**The theta notation bounds a functions from above and below, so it defines exact asymptotic behavior.

```
Θ((g(n)) = {f(n): there exist positive constants c1, c2 and n0 such that
                 0 <= c1*g(n) <= f(n) <= c2*g(n) for all n >= n0}
```

- **Big O Notation:** The Big O notation defines an upper bound of an algorithm, it bounds a function only from above.

```
O(g(n)) = { f(n): there exist positive constants c and n0 such that
                   0 <= f(n) <= cg(n) for all n >= n0}
```

Skip to content

- **Ω Notation:** Just as Big O notation provides an asymptotic upper bound on a function, Ω notation provides an asymptotic lower bound.

```
Ω (g(n)) = {f(n): there exist positive constants c and n0 such that
                0 <= cg(n) <= f(n) for all n >= n0}.
```

Solving recurrences

- **Substitution Method**: We make a guess for the solution and then we use mathematical induction to prove the guess is correct or incorrect.
- **Recurrence Tree Method:** We draw a recurrence tree and calculate the time taken by every level of tree. Finally, we sum the work done at all levels.
- **Master theorem Method:** Only for following type of recurrences or for recurrences that can be transformed to following type.

```
T(n) = aT(n/b) + f(n) where a >= 1 and b > 1
```

Sorting

Algorithm	Worst Case	Average Case	Best Case	Min. no. of swaps	Max. no. of swaps
Bubble	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$	**0**	$\Theta(n^2)$
Selection	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	**0**	$\Theta(n)$
Insertion	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$	**0**	$\Theta(n^2)$
Quick	$\Theta(n^2)$	$\Theta(nlgn)$	$\Theta(nlgn)$	**0**	$\Theta(n^2)$
Merge	$\Theta(nlgn)$	$\Theta(nlgn)$	$\Theta(nlgn)$	**Is not in-place sorting**	**Is not in-place sorting**
Heap	$\Theta(nlgn)$	$\Theta(nlgn)$	$\Theta(nlgn)$	$O(nlgn)$	$\Theta(nlgn)$

Searching

Algorithm	Worst Case	Average Case	Best Case
Linear Search	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$
Binary Search	$O(logn)$	$O(logn)$	$O(1)$

Trees

Trees: Unlike Arrays, Linked Lists, Stack and queues, which are linear data structures, trees are hierarchical data structures. Depth First Traversals: (a) Inorder (b) Preorder (c) Postorder Important Tree Properties and Formulas

Binary Search Tree

Skip to content

Binary Search Tree, is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree. There must be no duplicate nodes.

1. Insertion
2. Deletion

AVL Tree

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.

1. Insertion
2. Deletion

B-Tree

B-Tree is a self-balancing search tree. In most of the other self-balancing search trees (like AVL and Red Black Trees), it is assumed that everything is in main memory. To understand use of B-Trees, we must think of huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses.

Properties of B-Tree

1. B-Tree Insertion
2. B-Tree Deletion

Graph

Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost. Following two are the most commonly used representations of graph.

1. **Adjacency Matrix**: Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph.
2. **Adjacency List :** An array of linked lists is used. Size of the array is equal to number of vertices.

Graph Algorithms

Algorithm	**Time Complexity**
Breadth First Traversal for a Graph	O(V+E) for adjacency list representation and O(V * V) for adjacency matrix representation.
Depth First Traversal for a Graph	O(V+E) for adjacency list representation and O(V * V) for adjacency matrix representation.
Dijkstra's shortest path algorithm	Adjacency matrix- O(V^2). Adjacency list- O(E log V)
Topological Sorting: Shortest Path in Directed Acyclic Graph	O(V+E)

Some Interesting Graph Questions

Minimum Spanning Tree

Minimum Spanning Tree (MST) problem: Given connected graph G with positive edge weights, find a min weight set of edges that connects all of the vertices. MST is fundamental problem with diverse applic Skip to content

1. Network design– telephone, electrical, hydraulic, TV cable, computer, road

2. Approximation algorithms for NP-hard problems – traveling salesperson problem, Steiner tree

3. Cluster analysis- k clustering problem can be viewed as finding an MST and deleting the k-1 most expensive edges.

Example: Prim's Minimum Spanning Tree Algorithm, Kruskal's Minimum Spanning Tree Algorithm

Divide and Conquer

1. ***Divide:*** Break the given problem into subproblems of same type.
2. ***Conquer:*** Recursively solve these subproblems
3. ***Combine:*** Appropriately combine the answers

Following are some standard algorithms that are Divide and Conquer algorithms.
1) Binary Search is a searching algorithm. In each step, the algorithm compares the input element x with the value of the middle element in array. If the values match, return the index of middle. Otherwise, if x is less than the middle element, then the algorithm recurs for left side of middle element, else recurs for right side of middle element.
2) Quicksort is a sorting algorithm. The algorithm picks a pivot element, rearranges the array elements in such a way that all elements smaller than the picked pivot element move to left side of pivot, and all greater elements move to right side. Finally, the algorithm recursively sorts the subarrays on left and right of pivot element.
3) Merge Sort is also a sorting algorithm. The algorithm divides the array in two halves, recursively sorts them and finally merges the two sorted halves.
4) Closest Pair of Points The problem is to find the closest pair of points in a set of points in x-y plane. The problem can be solved in O(n^2) time by calculating distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(nLogn) time.

Greedy Approach

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Greedy algorithms are used for optimization problems. An optimization problem can be solved using Greedy if the problem has the following property: *At every step, we can make a choice that looks best at the moment, and we get the optimal solution of the complete problem*. **Following are some standard algorithms that are Greedy algorithms.**

1) Kruskal's Minimum Spanning Tree (MST): In Kruskal's algorithm, we create a MST by picking edges one by one. The Greedy Choice is to pick the smallest weight edge that doesn't cause a cycle in the MST constructed so far.

2) Prim's Minimum Spanning Tree: In Prim's algorithm also, we create a MST by picking edges one by one. We maintain two sets: set of the vertices already included in MST and the set of the vertices not yet included. The Greedy Choice is to pick the smallest weight edge that connects the two sets.

3) Dijkstra's Shortest Path: The Dijkstra's algorithm is very similar to Prim's algorithm. The shortest path tree is built up, edge by edge. We maintain two sets: set of the vertices already included in the tree and the set of the vertices not yet included. The Greedy Choice is to pick the edge that connects the two sets and is on the smallest weight path from source to the set that contains not yet included vertices.

4) Huffman Coding: Huffman Coding is a loss-less compression technique. It assigns variable length bit codes to different characters. The Greedy Choice is to assign least bit length code to the most frequent character.

Dynamic Programming

Dynamic Programming is an algorithmic paradigm that solves a given complex problem by breaking it into subproblems and stores the results of subproblems to avoid computing the same results again. **Properties:**

1. **Overlapping Subproblems:** Dynamic Programming is mainly used when solutions of same subproblems are needed again and again. In dynamic programming, computed solutions to subproblems are stored in a table so that these don't have to be recomputed.

Uses: Fibonacci Numbers

2. **Optimal Substructure**: A given problems has Optimal Substructure Property if optimal solution of the given problem can be obtained by using optimal solutions of its subproblems.

Uses: Longest Increasing Subsequence, Shortest Path Two App

1. Memoization (Top Down)
2. Tabulation (Bottom Up)

Examples: Floyd Warshall Algorithm, Bellman–Ford Algorithm for Shortest Paths

BackTracking

Backtracking is an algorithmic paradigm that tries different solutions until finds a solution that "works". Backtracking works in an incremental way to attack problems. Typically, we start from an empty solution vector and one by one add items .Meaning of item varies from problem to problem. Example: Hamiltonian Cycle

Last Updated : 27 Sep, 2022

Similar Reads

LMNs-C/C++

LMNs-Data Structure

Bitwise Algorithms | Intermediate

Bitwise Algorithms | Basic

Mathematical Algorithms | Number Digits

Mathematical Algorithms | Divisibility and Large Numbers

Mathematical Algorithms | Prime Factorization and Divisors

Quizes on Algorithms

Skip to content

Mathematical Algorithms | Prime numbers and Primality Tests

Mathematical Algorithms | GCD & LCM

Placement Training Program
Apply for Mentor

Master System Design
Master CP
GeeksforGeeks Videos

DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Visualisation Tutorial
Pandas Tutorial
NumPy Tutorial
NLP Tutorial
Deep Learning Tutorial

Python

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Python Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Competitive Programming

Top DS or Algo for CP
Top 50 Tree
Top 50 Graph
Top 50 Array
Top 50 String
Top 50 DP
Top 15 Websites for CP

System Design

What is System Design
Monolithic and Distributed SD
High Level Design or HLD
Low Level Design or LLD
Crack System Design Round
System Design Interview Questions

JavaScript

TypeScript
ReactJS
NextJS
AngularJS
NodeJS
Express.js
Lodash
Web Browser

Skip to content

Engineering Maths

Grokking Modern System Design

NCERT Solutions

Class 12

Class 11

Class 10

Class 9

Class 8

Complete Study Material

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

Accountancy

Business Studies

Indian Economics

Macroeconomics

Microeconimics

Statistics for Economics

Management & Finance

Management

HR Managament

Income Tax

Finance

Economics

UPSC Study Material

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economy Notes

Ethics Notes

Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

SSC CGL Practice Papers

Colleges

Indian Colleges Admission & Campus Experiences

Companies

IT Companies

Software Development Companies

Preparation Corner

Company Wise Preparation

Preparation for SDE

Exams

JEE Mains

JEE Advanced

GATE CS

NEET

UGC NET

More Tutorials

Software Development

Software Testing

Write & Earn

Write an Article

Improve an Article

Skip to content

Top Engineering Colleges

Top BCA Colleges

Top MBA Colleges

Top Architecture College

Choose College For Graduation

Artificial Intelligence(AI) Companies

CyberSecurity Companies

Service Based Companies

Product Based Companies

PSUs for CS Engineers

Experienced Interviews

Internship Interviews

Competitive Programming

Aptitude Preparation

Puzzles

Product Management

SAP

SEO

Linux

Excel

Pick Topics to Write

Share your Experiences

Internships