Data Structures and Algorithms - 14

Anantha Sharma

Contents

- Time & Space complexity
- Data Structures (Arrays, Lists, Stack, Queue, Heap, Hash table)
- Algorithms - Divide & Conquer, Recursion, Backtracking approaches
- Algorithms - Dynamic Programming (memoization, tabulation)
- Algorithms - DFS
- Algorithms - BFS
- Algorithms - Flood Fill, sliding window
- Algorithms - Greedy
- Algorithms - Finding Short Distance (Dijkstra, Floyd-Warshall)

Introduction

- A **data structure** is a data organization, and storage format that is usually chosen for efficient access to data

- An **algorithm** is a procedure used for solving a problem

- **Problem Solving skill**
  - Algorithmic problem solving skills are the **ability to apply logical, analytical, and creative thinking to formulate and execute algorithms**
  - Enables to tackle problems that are **not trivial or straightforward**
  - Can be improved with training & practice
  - Abstract problems created for training & practice

Time complexity

O(n)

```
for(i = 0; i < n; i++)
{
   // do something
}
```

O(n^2)

```
for(i = 0; i < n; i++)
   for(j = 0; j < n; j++)
   {
      // do something
   }
```

O(n log n)

```
for(i = 0; i < n; i=i*2)       // logn
   for(j = 0; j < n; j++)
   {
      // do something
   }
```

Time complexity

- CPU frequency ~2GHz --> ~10^9 instructions per second
- Complexity O(~10^7) expected to complete within 1 sec
- O(n) for n = 10^7
- O(n log n) for n = 10^5
- O(n^2) for n = 1000
- O(2^n) for n < 27
- O(n!) for n < 10

Memory complexity

- Stack memory limit 1MB
  - Recursion call depth >3000 may give segmentation fault

- Heap memory
  - Should not use more than 10MB ==> time limit exceed
  - Less than 1MB is good. Need not optimize

- Time optimization is higher priority than Memory optimization

Data structures

- Arrays
  - Used almost everywhere. Easy index access.
  - https://www.geeksforgeeks.org/array-data-structure/
- Linked list
  - Used in many of algorithms
  - https://www.geeksforgeeks.org/introduction-to-linked-list-data-structure-and-algorithm-tutorial/
- Stack (LIFO)
  - https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/
- Queue (FIFO)
  - used in tree/graph algorithms
  - https://www.geeksforgeeks.org/introduction-to-queue-data-structure-and-algorithm-tutorials/

Exercise

- Trapping rain water
  - https://www.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1
  - https://github.com/teja963/Advanced-DSA-and-CS-Theory/blob/master/Array/22.%20Trapping%20rainwater.cpp

- Maximum continuous sub array sum
  - https://github.com/teja963/Advanced-DSA-and-CS-Theory/blob/master/Array/Easy%20Ques/kadanes_algo.cpp

- Find Maximum element in a dynamic data set
  - Add / delete / modify elements

Exercise

- Count pairs with given sum
  - Given an array of N integers, and an integer K, find the number of pairs of elements in the array whose sum is equal to K
  - https://www.geeksforgeeks.org/problems/count-pairs-with-given-sum5022/1

- Subarray with 0 sum
  - Find if there is a subarray (of size at-least one) with 0 sum
  - https://www.geeksforgeeks.org/problems/subarray-with-0-sum-1587115621/1

Data structures

- Tree: Various concepts
  - https://www.geeksforgeeks.org/generic-treesn-array-trees/
  - https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/

- Binary Tree
  - https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/
  - Balanced Binary Tree: Log N complexity

- Heap: log n complexity
  - https://www.geeksforgeeks.org/introduction-to-heap-data-structure-and-algorithm-tutorials/

- Hash map
  - https://www.geeksforgeeks.org/introduction-to-hashing-data-structure-and-algorithm-tutorials/
  - https://www.geeksforgeeks.org/introduction-to-map-data-structure-and-algorithm-tutorials/
  - Versatile use case - highly important

Data structures

- Graph
  - Representation: Adjacency Matrix, Adjacency list, List of Edges
  - https://www.geeksforgeeks.org/introduction-to-graphs-data-structure-and-algorithm-tutorials/

- Advanced data structures
  - Trie, Segment Tree, Suffix array, Disjoint set, AVL tree, Red-Black Tree, Treap, ...
  - https://www.geeksforgeeks.org/advanced-data-structures/

Problem solving steps

- Understand problem
  - Read problem statement carefully
  - Constraints, restrictions, input size, number of function calls, value ranges

- Identify candidate solution
  - **Identify target complexity**
  - Divide into **known sub problems**
  - Check by **applying all known algorithms**
  - **Confirm complexity** of solution

- Write code **slowly & precisely**
  - Habituate **modular** & **optimal** coding

- Submit & hope to pass

Best practice

- Problem solving has muscle memory
  - Continuous practice makes identifying solution easier & faster
  - Given break for 3~4 months, needs restart from almost starting
- Habituate coding style
  - Write code slowly
  - Unit test each module
  - Compiler hints: register, restrict, auto
  - Target error free code
- 1 problem per week
- Buddies: group of 3~4 people

Exercise

- Binary Search
  - Given a sorted array of size N and an integer K, find the position(0-based indexing) at which K is present in the array
  - https://www.geeksforgeeks.org/problems/binary-search-1587115620/1
- Merge Sort
  - https://www.geeksforgeeks.org/merge-sort/
- Quick Sort
  - https://www.geeksforgeeks.org/quick-sort/

Divide and conquer

- Divide given problem into sub problems
- Solve both / all sub problems independently
- Merge results of sub problems to solve bigger problem
- Repeat steps

Recursion

- Method of implementing some algorithms
  - Divide & conquer
  - DFS graph traversal
  - Dynamic programming
  - Backtracking
- Recursion will use program stack
  - Stack limit can cause memory error
- Need precise termination condition
- Typically **code size is smaller**

Back tracking

- Backtracking is a problem-solving algorithmic technique that involves finding a solution incrementally by **trying different options** and **undoing** them if they lead to a **dead end**

- Typically implemented using recursion

- Time complexity could be exponential or factorial

Exercise

- Permutations of a given string
  - https://www.geeksforgeeks.org/problems/permutations-of-a-given-string2041/1
- N-Queen problem
  - https://www.geeksforgeeks.org/problems/n-queen-problem0315/1
- Rat in a Maze Problem
  - https://www.geeksforgeeks.org/problems/rat-in-a-maze-problem/1

Exercise

- **Coin change problem**
  - Given an integer array of **coins[ ]** of size N representing different types of denominations and an integer **sum**, the task is to count all combinations of coins to make a given value sum.
  - https://www.geeksforgeeks.org/coin-change-dp-7/

- **0/1 Knapsack Problem**
  - Given **N items w**here each item has some **weight** and **profit** associated with it and also given a **bag** with **capacity W** [i.e., the bag can hold at most W weight in it], The task is to put the items into the bag such that the sum of profits associated with them is the maximum possible.
  - https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

- **Subset Sum Problem**
  - Given a set of **non-negative integers** and a value **sum**, the task is to check if there is a subset of the given set whose sum is equal to the given sum.
  - https://www.geeksforgeeks.org/subset-sum-problem-dp-25/

Dynamic Programming

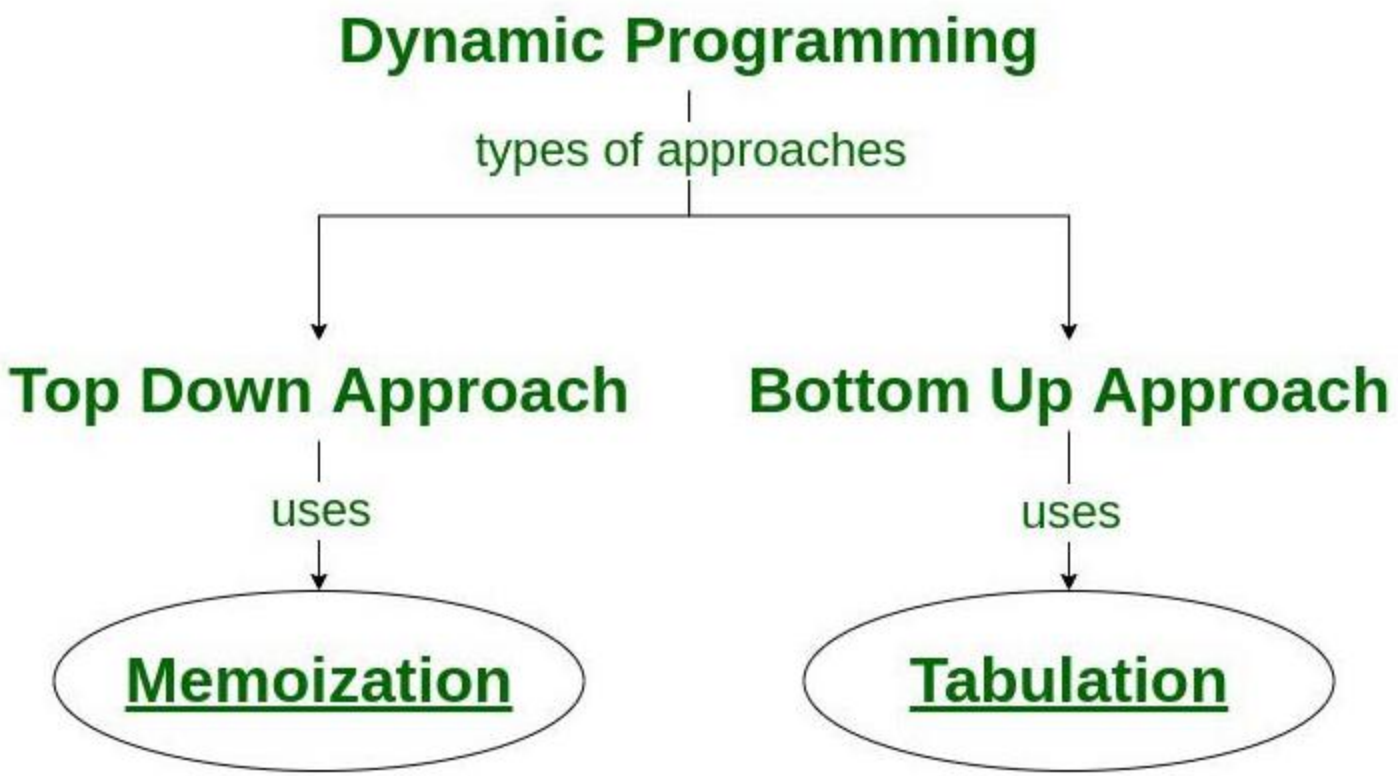
- Two necessary conditions should satisfy for Dynamic Programming

- **Overlapping Subproblems**: Same subproblems are needed repetitively for solving the actual problem.

- **Optimal Substructure Property**: The optimal solution of the given problem can be obtained by using optimal solutions of its subproblems.

Dynamic Programming

- Approaches



Dynamic Programming

- **Find recurrence relation**
- **Find existence of Overlapping sub problem** – if found, then it is a candidate for Dynamic Programming
- Typically easier to think and find out through top down approach
- After finding recurrence equation, it can be implemented either through memoization or tabulation

Exercise

- Longest Common Subsequence (LCS)
  - Given two strings, S1 and S2, the task is to find the length of the Longest Common Subsequence
  - https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/

- Nth Fibonacci Number
  - https://www.geeksforgeeks.org/problems/nth-fibonacci-number1335/1

- Longest Palindrome in a String
  - https://www.geeksforgeeks.org/problems/longest-palindrome-in-a-string3411/1

Exercise

- Many problems related to Dynamic Programming
  - https://www.geeksforgeeks.org/explore?page=1&category=Dynamic%20Programming&sortBy=submissions
  - Longest Increasing Subsequence
  - Edit Distance
  - Count ways to reach the Nth stair
  - Egg Dropping Puzzle
  - Rod Cutting
  - Matrix Chain Multiplication
  - Strictly Increasing Array

Exercise

- Shortest Source to Destination Path in a matrix
  - https://www.geeksforgeeks.org/problems/shortest-source-to-destination-path3544/1

- Steps by Knight
  - Given a square chessboard, the initial position of Knight and position of a target. Find out the minimum steps a Knight will take to reach the target position.
  - https://www.geeksforgeeks.org/problems/steps-by-knight5927/1

- Number of Distinct Islands
  - https://www.geeksforgeeks.org/problems/number-of-distinct-islands/1

Breadth First Search (BFS)

- BFS is fundamental **graph traversal algorithm**. It involves visiting all the connected nodes of a graph in a level-by-level manner

- Time complexity O(V+E)

- Useful for flood fill, path finding in maze

```
1  procedure BFS(G, root) is
2      let Q be a queue
3      label root as explored
4      Q.enqueue(root)
5      while Q is not empty do
6          v := Q.dequeue()
7          if v is the goal then
8              return v
9          for all edges from v to w in G.adjacentEdges(v) do
10             if w is not labeled as explored then
11                 label w as explored
12                 w.parent := v
13                 Q.enqueue(w)
```

Depth First Search (DFS)

- DFS is fundamental **graph traversal algorithm**. Explore one path till dead end and then back tracking.

- Pre-order, in-order, post-order traversal

- Useful method for Tree & Directed acyclic graph

- Usually implemented through recursion

```
procedure DFS(G, v) is
    label v as discovered
    for all directed edges from v to w that are in G.adjacentEdges(v) do
        if vertex w is not labeled as discovered then
            recursively call DFS(G, w)
```

Exercise

- Bipartite Graph
  - Check whether given graph is bipartite or not.
  - https://www.geeksforgeeks.org/problems/bipartite-graph/1

- Detect cycle in an undirected graph
  - https://www.geeksforgeeks.org/problems/detect-cycle-in-an-undirected-graph/1

- Word Search
  - Given a 2D board of letters and a word, check if the word exists in the board in 4 directions
  - https://www.geeksforgeeks.org/problems/find-the-string-in-grid0111/1

Exercise

- Topological sort
  - Given a Directed Acyclic Graph (DAG) with V vertices and E edges, Find any Topological Sorting.
  - https://www.geeksforgeeks.org/problems/topological-sort/1

- Replace O's with X's
  - Given a matrix where every element is either 'O' or 'X'. Replace all 'O' or a group of 'O' with 'X' that are surrounded by 'X'.
  - https://www.geeksforgeeks.org/problems/replace-os-with-xs0052/1

- Number of Provinces
  - Given an undirected graph, two vertices u and v belong to a single province if there is a path between u to v. Task is to find the number of provinces.
  - https://www.geeksforgeeks.org/problems/number-of-provinces/1

Exercise

- Flood fill Algorithm
  - Given a coordinate (sr, sc) representing the starting pixel of the flood fill, and a pixel value newColor, "flood fill" the image.
  - https://www.geeksforgeeks.org/problems/flood-fill-algorithm1856/1

- Covid Spread
  - An infected patient at ward [i,j] can infect other uninfected patient at adjacent ward in unit time. Determine the minimum units of time after which there won't remain any uninfected patient.
  - https://www.geeksforgeeks.org/problems/covid-spread--141631/1

Exercise

- Find number of anagrams
  - https://www.geeksforgeeks.org/problems/count-occurences-of-anagrams5839/1

- Find subarray with given sum
  - Given an unsorted array A of size N that contains only non-negative integers, find a continuous sub-array that adds to a given number Sum
  - https://www.geeksforgeeks.org/problems/subarray-with-given-sum-1587115621/1

- Max Sum Subarray of size K
  - https://www.geeksforgeeks.org/problems/max-sum-subarray-of-size-k5313/1

Sliding window

- Select a window (section) from starting.
  - Starting window size can be based on requirements

- Evaluate current result and determine either to **add next element** or **remove first element** or **both**

- [Add next element]

- [Remove next element]

- Repeat until end

Exercise

- Majority Element
  - Find the majority element in the array. A majority element appears strictly more than N/2 times.
  - https://www.geeksforgeeks.org/problems/majority-element-1587115620/1

- N meetings in one room
  - There are N meetings in the form of (start[i], end[i]). Find the maximum number of meetings that can be accommodated.
  - https://www.geeksforgeeks.org/problems/n-meetings-in-one-room-1587115620/1

- Maximize Toys
  - Given cost[ ] of N toys and an integer K amount, find maximum number of toys can buy.
  - https://www.geeksforgeeks.org/problems/maximize-toys0331/1

Greedy Algorithm

- A greedy algorithm is a problem-solving technique that makes the best local choice at each step in the hope of finding the global optimum solution.

- Applications
  - Dijkstra's Algorithm
  - Kruskal's Algorithm
  - Fractional Knapsack Problem
  - Huffman Coding

Shortest Path

- Dijkstra's algorithm
  - https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
  - Time complexity O(E log V)

```
1   function Dijkstra(Graph, source):
2       create vertex priority queue Q
3
4       dist[source] ← 0                          // Initialization
5       Q.add_with_priority(source, 0)            // associated priority equals dist[·]
6
7       for each vertex v in Graph.Vertices:
8           if v ≠ source
9               prev[v] ← UNDEFINED               // Predecessor of v
10              dist[v] ← INFINITY                // Unknown distance from source to v
11              Q.add_with_priority(v, INFINITY)
12
13
14      while Q is not empty:                     // The main loop
15          u ← Q.extract_min()                   // Remove and return best vertex
16          for each neighbor v of u:             // Go through all v neighbors of u
17              alt ← dist[u] + Graph.Edges(u, v)
18              if alt < dist[v]:
19                  prev[v] ← u
20                  dist[v] ← alt
21                  Q.decrease_priority(v, alt)
22
23      return dist, prev
```

Floyd-Warshall algorithm

- Floyd-Warshall all pair shortest path algorithm
  - https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/
  - Time complexity $O(V^3)$

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each edge (u, v) do
    dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
for each vertex v do
    dist[v][v] ← 0
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if
```

Exercise

- Minimum Platforms
  - Given arrival and departure times of all trains that reach a railway station, Find the minimum number of platforms required.
  - https://www.geeksforgeeks.org/problems/minimum-platforms-1587115620/1

Exercise

- Cheapest Flights Within K Stops
  - Given three integers src, dst, and k, return the cheapest price from src to dst with at most k stops.
  - https://www.geeksforgeeks.org/problems/cheapest-flights-within-k-stops/1

Exercise

- Find minimum range
  - Given H houses a row with co-ordinates (h1, h2... hn) and M signal towers all having same coverage distance (signal strength) can be installed anywhere, find minimum range required for the towers to coverage all houses.

Exercise

- Maximum Rectangular Area in a Histogram
  - Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars
  - https://www.geeksforgeeks.org/problems/maximum-rectangular-area-in-a-histogram-1587115620/1

Exercise

- Max rectangle
  - Find the maximum area of a rectangle formed only of 1s in the given matrix.
  - https://www.geeksforgeeks.org/problems/max-rectangle/1

Exercise

- Find median in a stream
  - https://www.geeksforgeeks.org/problems/find-median-in-a-stream-1587115620/1

Exercise

- Zero Sum Subarrays
  - Find the total count of sub-arrays having their sum equal to 0.
  - https://www.geeksforgeeks.org/problems/zero-sum-subarrays1825/1

Exercise

- Longest K unique characters substring
  - Given a string you need to print the size of the longest possible substring that has exactly K unique characters
  - https://www.geeksforgeeks.org/problems/longest-k-unique-characters-substring0853/1

Exercise

- Square root of a number
  - Given an integer x, find the square root of x.
  - https://www.geeksforgeeks.org/problems/square-root/1

Exercise

- Fill the Tank
  - N water tanks are connected by pipelines (as Tree). Water will be poured to a Source tank, when the tank is filled, the excess water evenly flows to the connected tanks. Find minimum amount of water poured into the selected tank so that all other tank is filled.
  - https://www.geeksforgeeks.org/problems/fill-the-tank3026/1

Exercise

- Job Sequencing Problem
  - Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. every job takes a single unit of time. Maximize the total profit if only one job can be scheduled at a time.
  - https://www.geeksforgeeks.org/job-sequencing-problem/

Exercise

- Max sum submatrix
  - Find the maximum sum among all (a x b) sub-matrices of a given matrix.
  - https://www.geeksforgeeks.org/problems/max-sum-submatrix2725/1

Exercise

- Range Minimum Query in array
  - Range Minimum Query (i, j): find minimum element from index i to j
  - Update (i, value): update ith element to value

Exercise

- Range Sum Query in array
  - Range Sum Query (i, j): find sum of element from index i to j
  - Update (i, value): update ith element to value

Practice !

Practice !!

Practice !!!