BST

A diagram of a diagram

Description automatically generated

1. Insertion: Insert these elements in order: 97, 24, 6, 8
2. Deletion: On this new tree, delete these elements in order: 9, 7, 54, 8(the top one)

Dynamic Programming

* Dynamic programming strategy and steps;
* 0-1 Knapsack problem (how to divide into small subproblems, recursive formula, bottom up solution, table)
  + Backpack with weight capacity of 10, here is your items:

|  |  |  |
| --- | --- | --- |
| Item | Weight | Benefit |
| 1 | 9 | 8 |
| 2 | 3 | 4 |
| 3 | 2 | 2 |
| 4 | 5 | 7 |

* + What is the optimal solution? (show chart and which items to add)
  + What is the subproblem in your chart for the cell (5, 2)
* LCS problem (how to divide into small subproblems, recursive formula, bottom-up solution, table)
* Change problem (how to divide into small subproblems, recursive formula, bottom-up solution, table)
* How to apply dynamic programming ideas to solve new problems.

Greedy Programming:

* Greedy Programming strategy, property, why the optimal solution is not guaranteed.
* Compare greedy programming and dynamic programming.
* Fractional knapsack problem
  + Backpack with weight capacity of 10, here is your items:

|  |  |  |
| --- | --- | --- |
| Item | Weight | Benefit |
| 1 | 9 | 8 |
| 2 | 3 | 4 |
| 3 | 2 | 2 |
| 4 | 5 | 7 |

* + What is the greedy solution? (assume 1 of each item, yes optimal would be different than before)
* Why can Huffman coding compress data?
* Huffman coding, Huffman tree, compression ratio, decoding
  + Make a Huffman code off of this: aaaaaaaaabcdbdbdbabba
  + Decode this: 000101001101

Graph:

* Terminologies (directed, undirected, weighted, unweighted, loop, simple graph, complete graph, parallel edges, subgraph, cycle)
* Graph Representation (adjacency list, adjacency matrix), time complexity, compare two representations (pros and cons)
* Graph Search (BFS, DFS)
  + A diagram of a network

    Description automatically generated
  + BFS, go!
  + DFS on same graph, go!
  + Don’t have to name the edges cuz im nice…but u could if u wanted
* Applications of BFS and DFS
* Minimum Spanning Tree Definition,
* Kruskal’s algorithm, Prim’s Algorithm, time complexity
  + A diagram of a network

    Description automatically generated with medium confidence
  + Kruskal’s, go!
  + Prim’s, go! (start at 0)
* Single-Source Shortest Path definition, negative edges;
* Bellman-Ford Algorithm, Dijkstra’s Algorithm, time complexity;
  + A black and white drawing of a triangle with circles and lines

    Description automatically generated
  + Dijkstra, go! (Start @ 0)
  + A diagram of a triangle with numbers and circles

    Description automatically generated
  + Belleman Ford on 3rd cycle alone, go! (l is start, order of cycle is: (k, l), (m, k), (l, m), (k, m), (n, l), (j, n), (l, j))
* All-pairs Shortest Path definition, Floyd-Warshall Algorithm (idea, recursive formula, subproblems)
  + A diagram of a triangle with lines and numbers

    Description automatically generated
  + Floyd-Warshall, go! (assume 1 = a, g = 2, h = 3, j = 4 for your D#)
* Make sure you understand all examples we learned.