15/3/23 UNIT-4 ANALYSIS OF GRAPH ALGORITHMS

Dynamic programming - warshalls and Floyd's algorithm - optimal binary search tree - Greedy technique - container loading problem - Huffman's trees - knapsack problems.

Dynamic programming

~~Fib(n) = Fib(n-1) + Fib(n-2)~~

Fib

A technique used to solve problems with overlapping sub problems.

Instead of solving the overlapping sub problems again and again they can be solved using dynamic programming once and record the results in a table from which a solution to original problem can be obtained.

Eg: $Fib(n) = Fib(n-1) + Fib(n-2)$

$Fib(5) = Fib(4) + Fib(3)$

Calculation of fib(n) requires repeated calculation for smaller values of n.

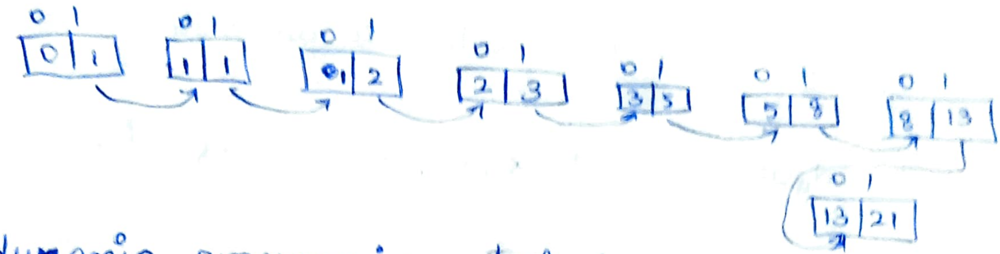
This can be avoided by storing or recording the previous value in an array.

Eg:

0	1	2	3	4	5	6	7	8
0	1	1	2	3	5	8	13	21

It uses n+1 locations causing storage issues.
Another optimatic solution is to stored only recent two values in an array whose locations can be overwritten.

eg



dynamic programming technique can be best used to solve optimization problem.

So this technique is called as principle of optimality.

The algorithms using this technique are

1. warshall's algorithm
2. Floyd's algorithm
3. optimal binary search tree

warshall's & Floyd's algorithm

Basic idea

Simpler versions of the problems are solved which can be used to solve higher versions of the problems till the original solutions is obtained.

warshall's algorithm

used to identify the transitive closure of a matrix. for a directed graph with every pair 2 matrices are possible 1.adjacency matrix, 2. transitive closure matrix

Floyd's algorithm

used to find the shortest path between every pair of vertices in a graph.

All pair shortest path algorithm.

16/3/23

Warshall's algorithm

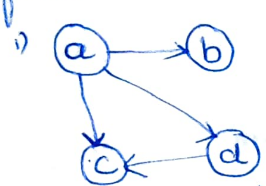
Adjacency matrix → It informs whether their is an edge from one vertex to an other.

Transitive closure → It informs whether their is a path from one vertex to another.

In adjacency matrix, A, $a_{ij} = 1$ iff edge from i to j.

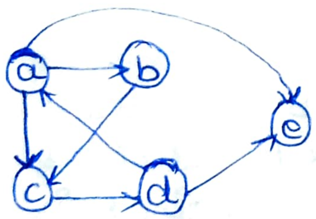
In transitive closure matrix, T, $t_{ij} = 1$ iff path from i to j.

Eg: 1)



$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \qquad T = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

2)



$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad T = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The algorithm uses dfs (or) bfs to identify the transitive closure.

This is implemented using n+1 matrices $R^{(0)}$, $R^{(1)}$, (adjacency matrix)

$R^{(2)}, \dots \dots R^{(n)}$ ↳ Transitive closure

Each is an $n \times n$ matrix.
Any matrix R^{k} is generated from R^{k-1}

In $R^{(k)}$, $r_{ij}^{(k)} = 1$ if their is path from i to j with ~~If from i~~ the intermediate vertices $1, 2, 3, \dots, k$

For eg., In $R^{(3)}$, $r_{14} = 1$ iff their is a path from 1 to 4 using 1 and 2 are intermediate vertices.

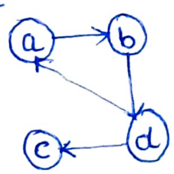
$$r_{ij}^{(k)} = 1 \text{ if } r_{ij}^{(k-1)} = 1$$

(or)

$$\text{if } r_{ik}^{(k-1)} = 1 \text{ and } r_{kj}^{(k-1)} = 1$$

$$R^{(k-1)} = \begin{array}{c|cc} & j & k \\ \hline k & 1 & \\ i & & 1 \end{array} \qquad R^{(k-1)} = \begin{array}{c|cc} & j & k \\ \hline k & 1 & \\ i & 1 & 1 \end{array}$$

Eg



$$R^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array}$$

Use 1st row and 1st column of R^0 to compute the matrix of R^1.

$r_{da}^{(0)} = 1 \quad r_{ba}^{(0)} = 1$ so $r_{db}^{(1)} = 1$

$$R^{1} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 0 \end{array}$$

Use the 2nd row & 2nd column of R^1 to compute R^2.

db/da, ab, ad = db

$r_{db}^{(1)} = 1, \; r_{ab}^{(1)} = 1, \; r_{bd}^{(1)} = 1, \; r_{db}^{(1)} = 1$

So $r_{ad}^{(2)} = 1$ So $r_{dd}^{(2)} = 1$

$r_{ab}^{(1)} = 1$, $r_{db}^{(1)} = 1$, $r_{bd}^{(1)} = 1$ So $r_{ad}^{(2)} = 1$, $r_{bd}^{(1)} = 1$ So $r_{dd}^{(2)} = 1$

$$R^{2} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

$r_{dc}^{(3)} = 1$

Use the 3rd row & 3rd colum of R^2 to compute $R3$.

$$R^{3} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

$r_{db}^{(3)} = 1$

$r_{ad}^{(3)} = 1$

$r_{dd}^{(3)} = 1$

Use the 4th row & 4th Column of R^3 to compute R_4.

$r_{da}^{(3)} = 1 \qquad r_{dc}^{(3)} = 1$

$r_{db}^{(3)} = 1$

$r^{(3)}_{ad} = 1$ $\quad r^{(3)}_{da} = 1$ $\quad \cancel{r^{(4)}_{aa} = 1}$ $\quad r^{(4)}_{aa} = 1$ $\quad r^{(4)}_{ba} = 1$ $\quad r^{(4)}_{da} = 1$

$r^{(3)}_{bd} = 1$ $\quad r^{(3)}_{db} = 1$ $\quad \cancel{r^{(4)}_{bb} = 1}$ $\quad r^{(4)}_{ab} = 1$ $\quad r^{(4)}_{bb} = 1$ $\quad r^{(4)}_{db} = 1$

$r^{(3)}_{dd} = 1$ $\quad r^{(3)}_{dc} = 1$ $\quad \cancel{r^{(4)}_{dc} = 1}$ $\quad r^{(4)}_{ac} = 1$ $\quad r^{(4)}_{bc} = 1$ $\quad r^{(4)}_{dc} = 1$

$r^{(3)}_{dd} = 1$ $\quad r^{(4)}_{ad} = 1$ $\quad r^{(4)}_{bd} = 1$ $\quad r^{(4)}_{dd} = 1$

$$R^{4} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

21/3/23

Algorithm.

Alg warshalls (A[1..n, 1..n])

// Implements warshalls algorithm to find the transitive closure for a digraph.

// i/p : Adjacency matrix A for a directed graph.

// o/p : Transitive closure of a digraph. - shows there existence path between every pair of vertices.

$R^{(0)} = A$

for $K = 1$ to n do

  for $i = 1$ to n do

    for $j = 1$ to n do

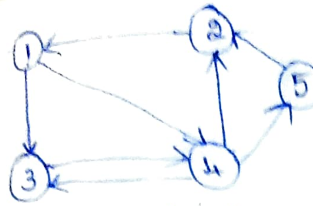
      $R^{(k)}_{ij} = R^{(k-1)}_{ij}$ or $R^{(k-1)}_{ik}$ and $R^{(k-1)}_{kj}$

return $R^{(n)}$.

Analysis

$$\boxed{C(n) = \Theta(n^3)}$$

$r^{(4)}_{da} = 1$
$r^{(4)}_{db} = 1$
$r^{(4)}_{dc} = 1$
$r^{(4)}_{dd} = 1$



Apply warshall's algorithm to find the transitive closure of the given digraph.

$$R^0 = \begin{array}{c|c} & \begin{matrix}1 & 2 & 3 & 4 & 5\end{matrix} \\ \hline \begin{matrix}1\\2\\3\\4\\5\end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}\end{array}$$

use 1st row and 1st column of R^0 to compute $R^{(1)}$.

$r^{(0)}_{21} = 1$ $\quad r^{(0)}_{13} = 1$ $\quad$ so $r^{(1)}_{23} = 1$ $\quad r^{(1)}_{24} = 1$

$r^{(0)}_{14} = 1$

here

$$R^1 = \begin{array}{c|c} & \begin{matrix}1 & 2 & 3 & 4 & 5\end{matrix} \\ \hline \begin{matrix}1\\2\\3\\4\\5\end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}\end{array}$$

use 2nd row and 2nd column of R^1 to compute $R^{(2)}$.

$r^{(1)}_{42} = 1$ $\quad r^{(1)}_{21} = 1$ $\quad$ so $r^{(2)}_{41} = 1$ $\quad r^{(2)}_{51} = 1$

$r^{(1)}_{23} = 1$ $\quad r^{(2)}_{43} = 1$ $\quad r^{(2)}_{44} = 1$

$r^{(1)}_{52} = 1$ $\quad r^{(1)}_{24} = 1$ $\quad r^{(2)}_{54} = 1$ $\quad r^{(2)}_{53} = 1$.

$$R^2 = \begin{array}{c|c} & \begin{matrix}1 & 2 & 3 & 4 & 5\end{matrix} \\ \hline \begin{matrix}1\\2\\3\\4\\5\end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}\end{array}$$

Use 3rd row & 3rd column of R^2 to compute R^3.

$r^{(2)}_{13} = 1$ $\quad r^{(2)}_{53} = 1$ $\quad r^{(2)}_{34} = 1$ $\quad$ so $r^{(3)}_{14} = 1$

$r^{(2)}_{23} = 1$ $\quad r^{(3)}_{24} = 1$

$r^{(2)}_{43} = 1$ $\quad r^{(3)}_{44} = 1$

$r^{(3)}_{54} = 1$

$$R^3 = \begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \hline \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Use 4^{th} row & 4^{th} column of R^3 to compute R^4

$r_{14}^{(3)} = 1$, $r_{24}^{(3)} = 1$, $r_{34}^{(3)} = 1$, $r_{44}^{(3)} = 1$, $r_{54}^{(3)} = 1$

$r_{41}^{(3)} = 1$, $r_{42}^{(3)} = 1$, $r_{43}^{(3)} = 1$, $r_{44}^{(3)} = 1$, $r_{45}^{(3)} = 1$

so

$r_{11}^{(4)} = 1$, $r_{12}^{(4)} = 1$, $r_{13}^{(4)} = 1$, $r_{14}^{(4)} = 1$, $r_{15}^{(4)} = 1$

$r_{21}^{(4)} = 1$, $r_{22}^{(4)} = 1$, $r_{23}^{(4)} = 1$, $r_{24}^{(4)} = 1$, $r_{25}^{(4)} = 1$

$r_{31}^{(4)} = 1$, $r_{32}^{(4)} = 1$, $r_{33}^{(4)} = 1$, $r_{34}^{(4)} = 1$, $r_{35}^{(4)} = 1$

$r_{41}^{(4)} = 1$, $r_{42}^{(4)} = 1$, $r_{43}^{(4)} = 1$, $r_{44}^{(4)} = 1$, $r_{45}^{(4)} = 1$

$r_{51}^{(4)} = 1$, $r_{52}^{(4)} = 1$, $r_{53}^{(4)} = 1$, $r_{54}^{(4)} = 1$, $r_{55}^{(4)} = 1$

$$R^4 = \begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \hline \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Use 5^{th} row & 5^{th} column of R^4 to compute R^5

$r_{15}^{(4)} = 1$, $r_{25}^{(4)} = 1$, $r_{35}^{(4)} = 1$, $r_{45}^{(4)} = 1$, $r_{55}^{(4)} = 1$

$r_{52}^{(4)} = 1$, $r_{53}^{(4)} = 1$, $r_{54}^{(4)} = 1$, $r_{55}^{(4)} = 1$, $r_{51}^{(4)} = 1$

so

$r_{11}^{(5)} = 1$, $r_{12}^{(5)} = 1$, $r_{13}^{(5)} = 1$, $r_{14}^{(5)} = 1$, $r_{15}^{(5)} = 1$

$r_{21}^{(5)} = 1$, $r_{22}^{(5)} = 1$, $r_{23}^{(5)} = 1$, $r_{24}^{(5)} = 1$, $r_{25}^{(5)} = 1$

$r_{31}^{(5)} = 1$, $r_{32}^{(5)} = 1$, $r_{33}^{(5)} = 1$, $r_{34}^{(5)} = 1$, $r_{35}^{(5)} = 1$

$r_{41}^{(5)} = 1$, $r_{42}^{(5)} = 1$, $r_{43}^{(5)} = 1$, $r_{44}^{(5)} = 1$, $r_{45}^{(5)} = 1$

$r_{51}^{(5)} = 1$, $r_{52}^{(5)} = 1$, $r_{53}^{(5)} = 1$, $r_{54}^{(5)} = 1$, $r_{55}^{(5)} = 1$

$$R^5 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 1 & 1 & 1 & 1 \end{array}$$

Floyd's algorithm

use to find the shortest paths between every pair of vertices in a weighted directed graph.

Used to solve all pairs shortest path problem.

It uses two matrix

1. weight matrix

In weight matrix, w_{ij} = the weight for the edge from the ~~vertices~~ i to j, If there is no edge from i to j, mark it with infinity.

2. distance matrix (D)

D_{ij} → Shortest path weight from vertex i to j.

The algorithm uses a series of matrices

$D^0, D^1, \ldots \ldots D^{k-1}, D^k \ldots D^n$ where D^n is a distance matrix

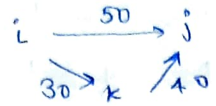
D^0 ↓ weight matrix

The matrix D^k is constructed from its predecessor $D^{(k-1)}$.

$d_{ij}^{(k)}$ → gives the path weight for the shortest path from i to j through the intermediate vertices. (1, 2, ..., k)

$$d_{ij}^{(k)} = \min \left\{ d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right\} \quad \text{for } k \geq 1$$

Eg. $i \xrightarrow{50} j$



$$d_{ij}^{(0)} = w_{ij}$$

Alg Floyd's (W[1..n, 1..n])

// Implements floyd's algorithm to find the shortest paths ~~to~~ between every pair of vertices in a directed weighted graph.

// i/p: The weight matrix W for a digraph.

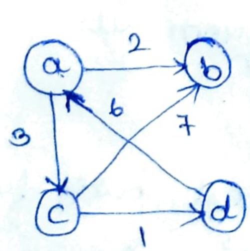
// o/p: Distance matrix D.

```
D = W
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n
            D[i, j] = min ( D[i, j], D[i, k] +
                                     D[k, j] )
return D.
```

Analysis

$$\boxed{C(n) = \Theta(n^3)}$$

Eg.



Apply floyd's algorithm to find the distance matrix for the given graph.

(or)

Apply floyd's algorithm & solve the all pairs shortest path problem for the given graph.

$$D^{(0)} = W = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

use 1st column & 1st row of D^0 to construct D^1.

$D_{ba} = 2$ $\quad D_{ac} = 3$ $\quad$ so $\quad d_{bc} = \min(\infty, 2+3) = 5$

$D_{da} = 6$ $\quad d_{dc} = \min(\infty, 6+3) = 9$

$$D^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

use 2nd column & 2nd row of D^1 to construct D^2

$d_{cb} = 7$ $\quad d_{ba} = 2$ $\quad$ so $\quad d_{ca} = \min(\infty, 7+2) = 9$

$d_{bc} = 5$ $\quad d_{cc} = \min(0, 7+5) = 0$

$$D^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & 9 & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

use 3rd column & 3rd row of D^2 to construct D^3.

$d_{ac} = 3$ $\quad d_{ca} = 9$ $\quad$ so $\quad d_{aa} = \min(0, 3+9) = 0$

$d_{bc} = 5$ $\quad d_{cb} = 7$ $\quad d_{ab} = \min(\infty, 3+7) = 10$

~~$d_{ac} = \min(3, 3+12) = 3$~~

$d_{ad} = \min(\infty, 3+1) = 4$

$d_{dc} = 9$ $\quad d_{cd} = 1$ $\quad d_{ba} = \min(2, 9+5) = 2$

$d_{bb} = \min(0, 5+7) = 0$

~~$d_{bc} = \min(5, 5+12) = 5$~~

$d_{bd} = \min(\infty, 5+1) = 6$

~~$d_{ca} = \min(9, 12+9) = 9$~~

~~$d_{cb} = \min(7, 12+7) = 7$~~

~~$d_{cc} = \min(12, 12+12) = 12$~~

~~$d_{cd} = \min(1, 12+1) = 1$~~

$d_{da} = \min(6, 9+9) = 6$

$d_{db} = \min(\infty, 9+7) = 16$

~~$d_{dc} = \min(9, 9+12) = 9$~~

$d_{dd} = \min(0, 9+1) = 0$

$$D^3 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Use 4^{th} column and 4^{th} row of D^3 to construct D^4

$d_{ad} = 4$ $\quad d_{da} = 6$

$d_{bd} = 6$ $\quad d_{db} = 16$

$d_{cd} = 1$ $\quad d_{dc} = 9$

$d_{aa} = \min(0, 4+6) = 0$

$d_{ab} = \min(10, 4+16) = 10$

$d_{ac} = \min(3, 4+9) = 3$

$d_{ba} = \min(2, 6+6) = 2$

$d_{bb} = \min(0, 6+16) = 0$

$d_{bc} = \min(5, 6+9) = 5$

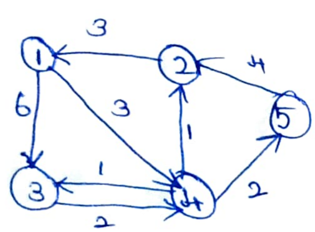
$d_{ca} = \min(9, 1+6) = 7$

$d_{cb} = \min(7, 1+16) = 7$

$d_{cc} = \min(0, 1+9) = 0$

$$D^4 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Eg: 2



$$D^0 = W = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 2 & \infty \\ \infty & 1 & 1 & 0 & 2 \\ \infty & 4 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Use 1st column and 1st row of D^0 to construct D^1

$d_{21} = 3$ $\quad d_{13} = 6$ $\quad d_{23} = \min(\infty, 3+6) = 9$

$d_{14} = 3$ $\quad d_{24} = \min(\infty, 3+3) = 6$

$$D^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ \infty & 1 & 1 & 0 & 2 \\ \infty & 4 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Use 2nd col & 2nd row of D^1 to construct D^2.

$d_{42} = 1$

$d_{52} = 4$

$d_{21} = 3$

$d_{23} = 9$

$d_{24} = 6$

$d_{41} = \min(\infty, 1+3) = 4$

$d_{43} = \min(1, 1+9) = 1$

$d_{44} = \min(0, 1+6) = 0$

$d_{51} = \min(\infty, 4+3) = 7$

$d_{53} = \min(\infty, 4+9) = 13$

$d_{54} = \min(\infty, 6+4) = 10$

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & 2 \\ 7 & 4 & 13 & 10 & 0 \end{bmatrix} \end{array}$$

Use 3rd col & 3rd row of D^2 to construct D^3.

$d_{13} = 6$

$d_{23} = 9$

$d_{43} = 1$

$d_{53} = 13$

$d_{34} = 2$

$d_{14} = \min(3, 6+2) = 3$

$d_{24} = \min(6, 9+2) = 6$

$d_{44} = \min(0, 1+2) = 0$

$d_{54} = \min(10, 13+2) = 10$

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & 2 \\ 7 & 4 & 13 & 10 & 0 \end{bmatrix} \end{array}$$

Use 4th col & 4th row of D^3 to construct D^4.

$d_{14} = 3$

$d_{24} = 6$

$d_{34} = 2$

$d_{54} = 10$

$d_{41} = 4$

$d_{42} = 1$

$d_{43} = 1$

$d_{45} = 2$

$d_{11} = \min(0, 3+4) = 0$

$d_{12} = \min(\infty, 3+1) = 4$

$d_{13} = \min(6, 3+1) = 4$

$d_{15} = \min(\infty, 3+2) = 5$

$d_{21} = \min(3, 6+4) = 3$

$d_{22} = \min(0, 6+1) = 0$

$d_{23} = \min(9, 6+1) = 7$

$d_{25} = \min(\infty, 6+2) = 8$

$d_{31} = \min(\infty, 2+4) = 6$

$d_{32} = \min(\infty, 2+1) = 3$

$d_{33} = \min(0, 2+1) = 0$

$d_{35} = \min(\infty, 2+2) = 4$

$d_{51} = \min(7, 10+4) = 7$

$d_{52} = \min(4, 10+1) = 4$

$d_{53} = \min(13, 10+1) = 11$

$d_{55} = \min(0, 10+2) = 0$

$$D^4 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 4 & 4 & 3 & 5 \\ 2 & 3 & 0 & 7 & 6 & 8 \\ 3 & 6 & 3 & 0 & 2 & 4 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 5 & 7 & 4 & 11 & 10 & 0 \end{array}$$

use 5th Col & 5th row of D^4 to compute D^5.

$d_{15} = 5$ $d_{51} = 7$

$d_{25} = 8$ $d_{52} = 4$

$d_{35} = 4$ $d_{53} = 11$

$d_{45} = 2$ $d_{54} = 10$

$d_{11} = \min(0, 5+7) = 0$

$d_{12} = \min(4, 5+4) = 4$

$d_{13} = \min(4, 5+11) = 4$

$d_{14} = \min(3, 5+10) = 3$

$d_{21} = \min(3, 8+7) = 3$

$d_{22} = \min(0, 8+4) = 0$

$d_{23} = \min(7, 8+11) = 7$

$d_{24} = \min(6, 8+10) = 6$

$d_{31} = \min(6, 4+7) = 6$

$d_{32} = \min(3, 4+4) = 3$

$d_{33} = \min(0, 4+11) = 0$

$d_{34} = \min(2, 4+10) = 2$

$d_{41} = \min(4, 2+7) = 4$

$d_{42} = \min(1, 2+4) = 1$

$d_{43} = \min(1, 2+11) = 1$

$d_{44} = \min(0, 2+10) = 0$

$$D^5 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 4 & 4 & 3 & 5 \\ 2 & 3 & 0 & 7 & 6 & 8 \\ 3 & 6 & 3 & 0 & 2 & 4 \\ 4 & 4 & 1 & 1 & 0 & 2 \\ 5 & 7 & 4 & 11 & 10 & 0 \end{array}$$

23/8/23

Greedy ~~algorithm~~ Technique

The technique applicable to solve optimization problem.

It is a technique used to construct a solution through a sequence of steps, each expanding a partial solution ~~so~~ obtained so for, until the complete solution to the entire problem is obtained.

The solution obtained in any step must be the choice with the following properties.

1. Feasible → satisfying ~~the~~ constraints.
2. Locally optimal → The best choice among the feasible choices in the current step.
3. Irrevocable → once chosen, it cannot be changed.

Few problems that used greedy technique.

1. prism algorithm
2. Krushkal's algorithm
3. Dijkshatra
4. Container loading problem
5. Huffmann trees
6. Fractional knapsach problem.

Huffman trees

A ~~the~~ text with symbol or characters from some n-symbol alphabet set can be coded by assigning some sequence of bits for each symbol of the test, to create a code word.

Eg

a p p l e can be coded as

1001 111 111 101 100

This can be done using Huffmann Trees.

It can be used in data compression algorithm also.

The process of finding the code for a text is called as encoding.

The code is decoded to retrieve the text back.

Two types of encoding

1. Fixed Length encoding - All characters have the ~~code~~ code of Same size.

2. variable Length encoding - the characters have different code size - needs a technique to differentiate the characters.

Procedure to create huffman trees.

Step 1: create n - ~~symbol~~ single noded trees with each characters frequency as a roots.

Step 2: Combine two trees with minimum frequency and create a new tree with the Some of frequencies as the new root. Attach the two ~~sss~~ tree as the left & right subtrees.

Step 3: Repeat step 2, until a single tree is obtained.

Step 4: Assign 0 to the left subtree and 1 to the right subtree, in each node.

step 5: Collect 0's & 1's from the root to each leaf and generate the code for each corresponding character at the leaf.

27/3/23

Example.

1)

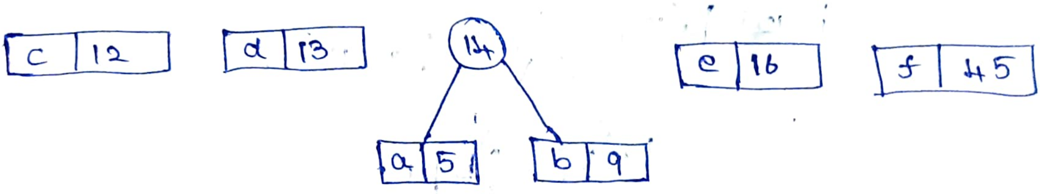
characters	a	b	c	d	e	f
Frequency	5	9	12	13	16	45

generate the hauffman trees & hauffman codes for the given characters.

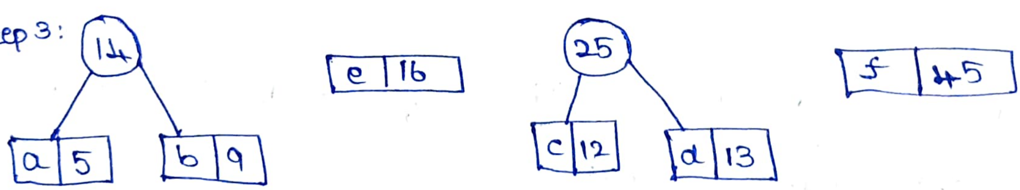
soln

step 1: a 5 | b 9 | c 12 | d 13 | e 16 | f 45

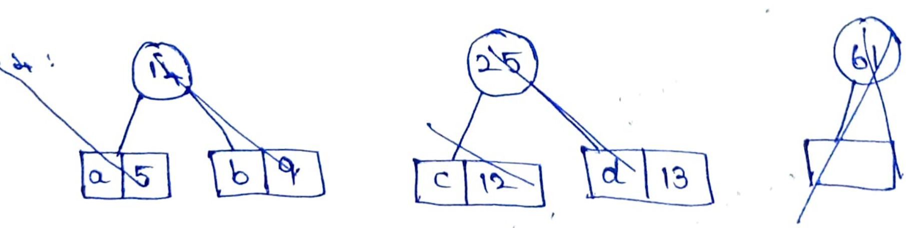
step 2:



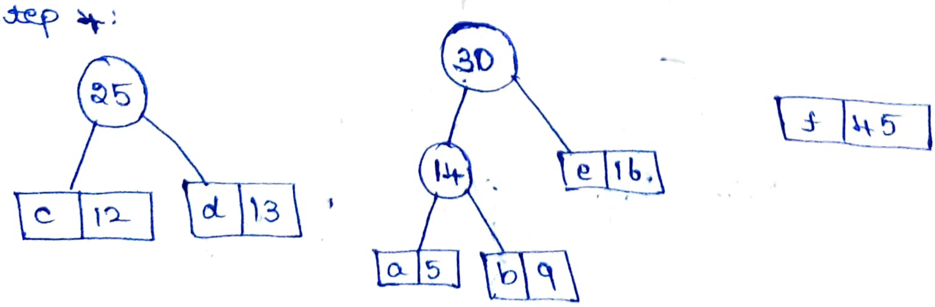
step 3:



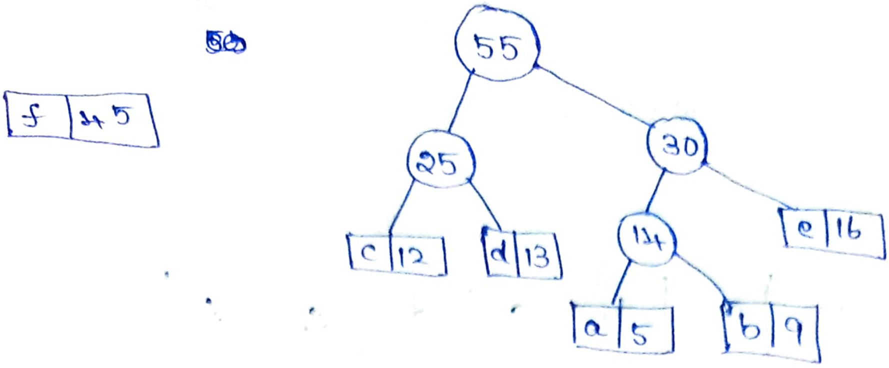
step 4:



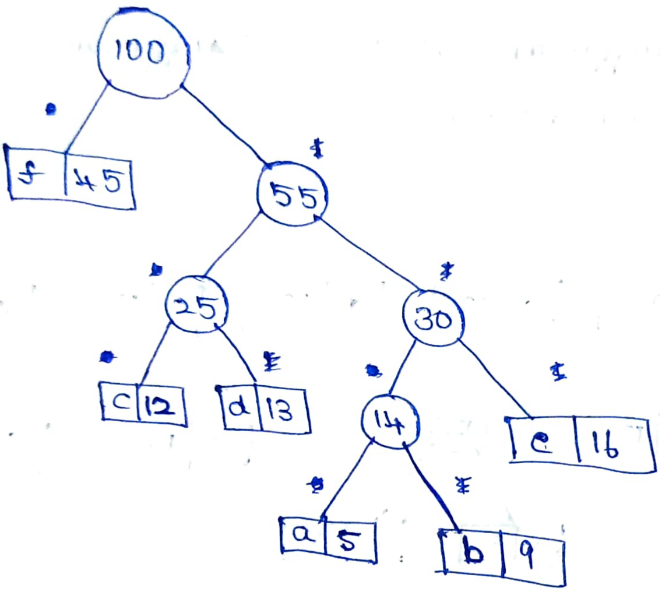
step 4:



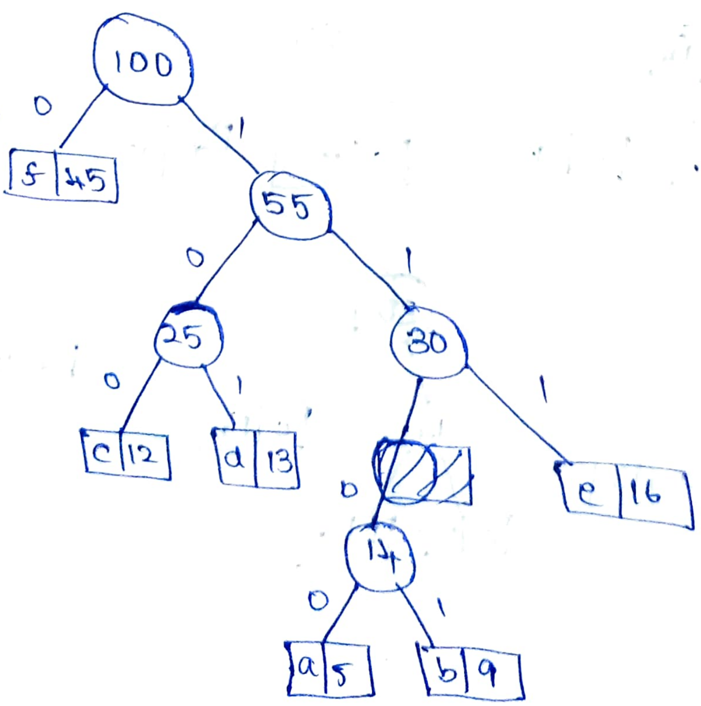
Step 5 :



Step 6 :



Now we assign 0 to the left subtrees and 1 to the right subtrees.



characters	a	b	c	d	e	f
frequency	1100	1101	100	101	111	0

2) Generate the huffman code for the following data.

a : 1 b : 1 c : 2 , d : 3 e : 5 f : 8 g : 13 h : 21

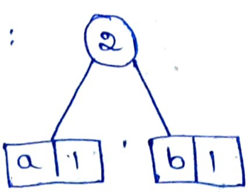
compressing of alphabets & frequency.

characters	a	b	c	d	e	f	g	h
frequency	1	1	2	3	5	8	13	21

soln

step 1 : a|1 b|1 c|2 d|3 e|5 f|8 g|13 h|21

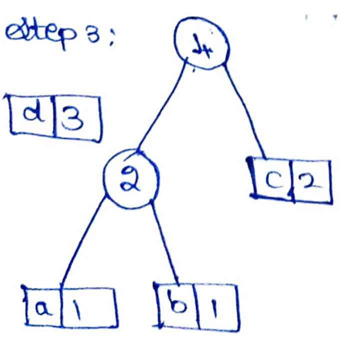
step 2 :



c|2 d|3 e|5 f|8 g|13 h|21

step 3 :

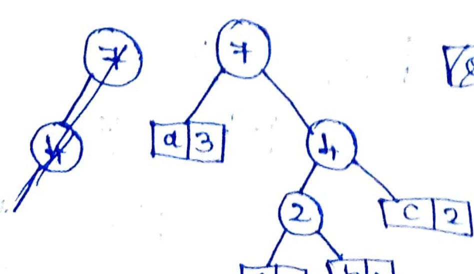
d|3



~~c|3~~ e|5 f|8 g|13 h|21

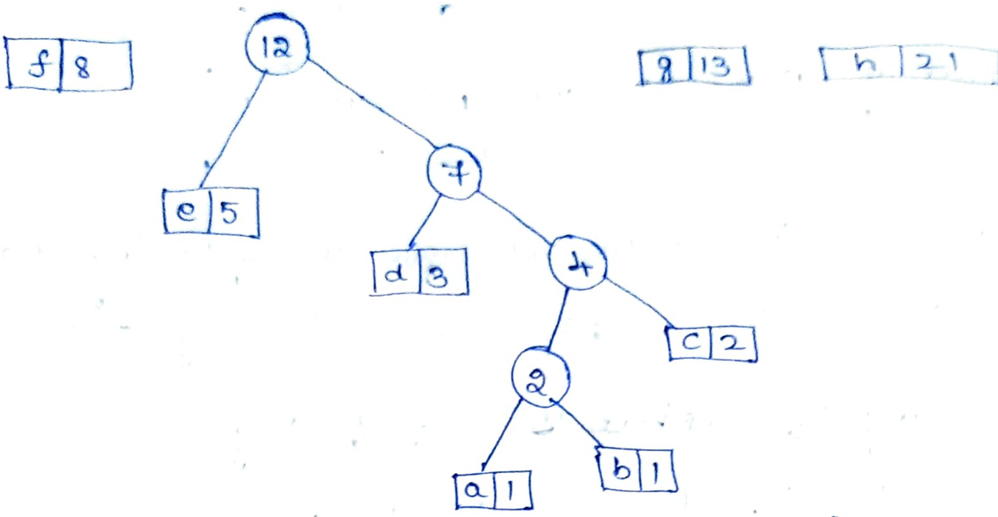
step 4 :

e|5

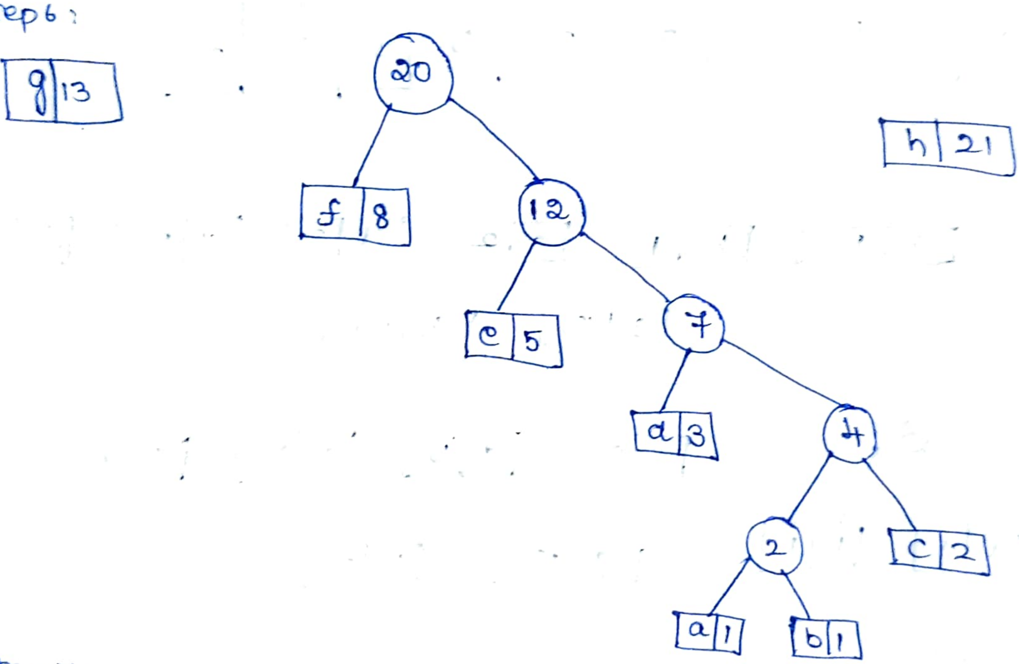


~~e|5~~ f|8 g|13 h|21

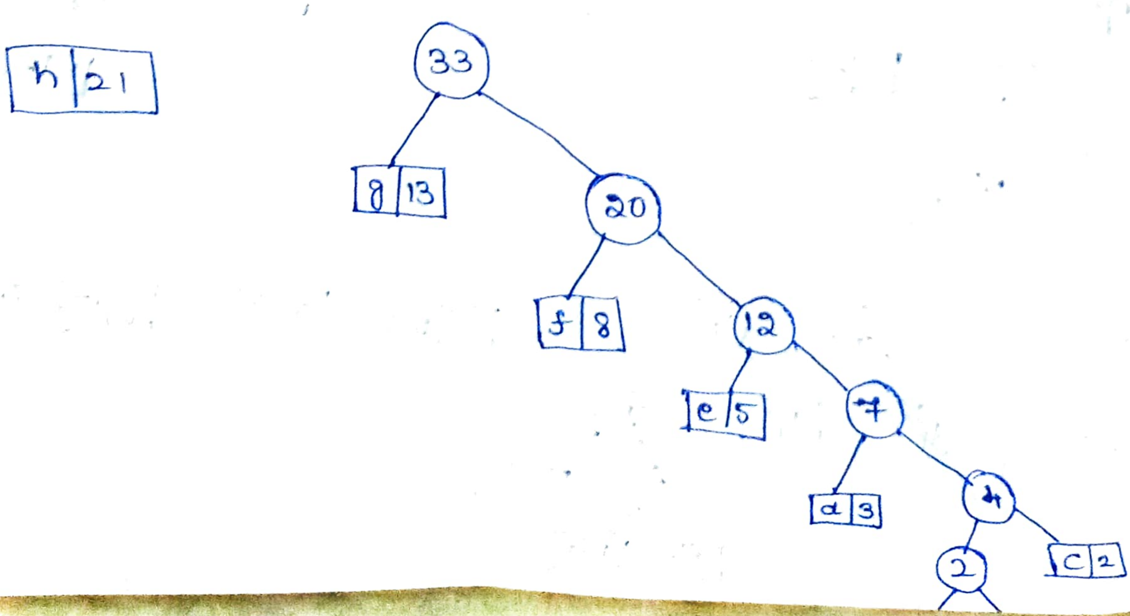
step 5:



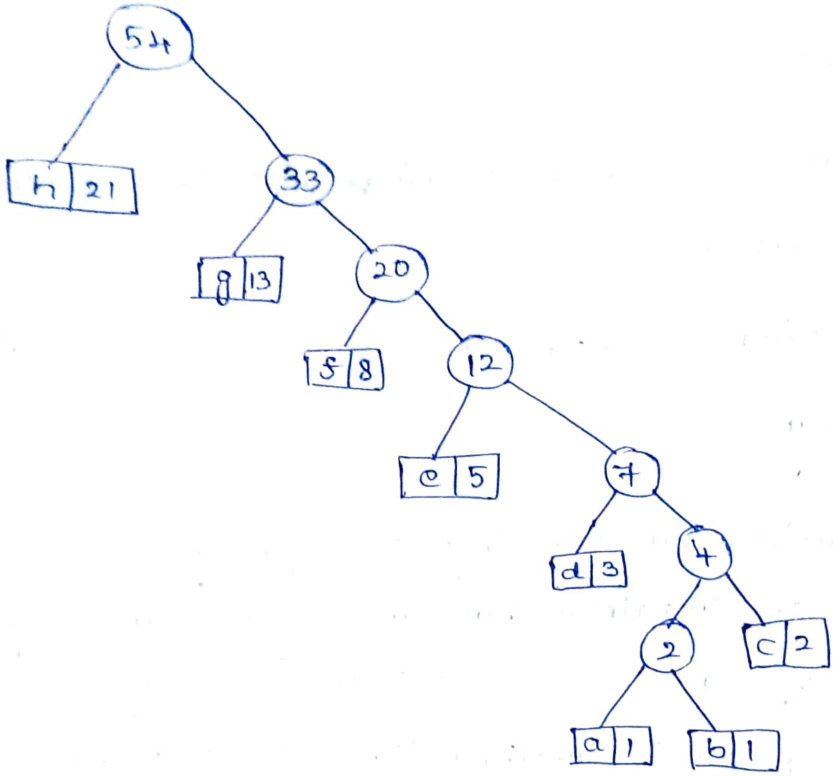
step 6:



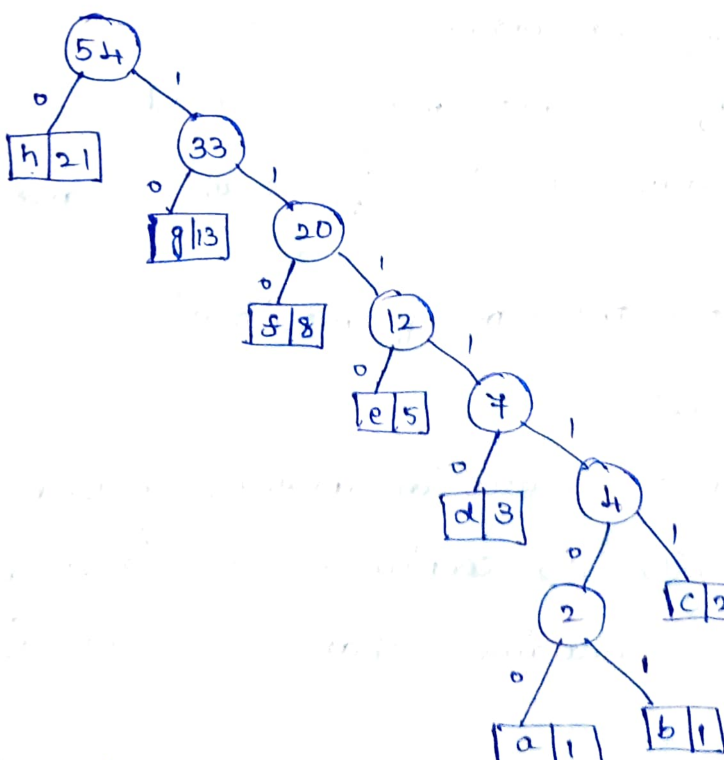
step 7:



Step 8.



Assign 0 to the left subtree and 1 to the right subtree.



Character	a	b	c	d	e	f	g	h
frequency	1111100	1111101	111111	11110	1110	110	10	0

Container loading problem.

A large , containers are of same size, but with different weights.

Objective : to load a ship with the maximum no. of containers.

constraint : without exceeding the cargo ship's weight capacity.

Parameters : Let the cargo's capacity be c.

Each ith container has an associated weight w_i.

Sum of the loaded container's weights must be $\leq c$.

Let x_i denote whether the ith container is loaded or not.

$x_i \in \{0, 1\}$

$x_i = 0$, if the container is not loaded

$x_i = 1$, if the container is loaded

Assign values to x_i $\ni$, $\sum_{i=1}^{n} x_i w_i \leq c$ and $\sum_{i=1}^{n} x_i$ is maximised.

n is the total no. of container

Procedure

Load containers in increasing order of weight (start with the container with least weight) until we get to a container that does not fit.

Example

Let the capacity of the cargo be $c = 400$ & the maximum no. of containers available, $n = 8$.

weight of each container is given below.

container	1	2	3	4	5	6	7	8
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
	100	200	50	150	90	50	20	80

find the optimal set of containers that can be loaded.

Soln

Initially, total weight $=0$

start loading with the container with minimum weight.

- Load the container C_7 with $w_7 = 20$.

so the total weight $= 0+20 = 20 < C$.

so load the container C_7 and make $x_7 = 1$

step	Loaded containers	weight (w_i)	Total weight	x_i
1	7	$w_7 = 20$	$0+20 = 20 < 400$	$x_7 = 1$
2	3	$w_3 = 50$	$20+50 = 70 < 400$	$x_3 = 1$
3	6	$w_6 = 50$	$70+50 = 120 < 400$	$x_6 = 1$
4	8	$w_8 = 80$	$120+80 = 200 < 400$	$x_8 = 1$
5	5	$w_5 = 90$	$200+90 = 290 < 400$	$x_5 = 1$
6	1	$w_1 = 100$	$290+100 = 390 < 400$	$x_1 = 1$
7	4	$w_4 = 150$	$390+150 = 540 > 400$	$x_4 = 0$

stop the loading process as the total

weight exceeds the capacity, c.

Container	1	2	3	4	5	6	7	8
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
	100	200	50	150	90	50	20	80
x_i	1	0	1	0	1	1	1	1

$$\sum_{i=1}^{n} x_i w_i = 1\times100 + 0 + 1\times50 + 0 + 1\times90 + 1\times50 + 1\times20 + 1\times80$$

$$= 100 + 50 + 90 + 50 + 20 + 80$$

$$= 390 \leq 400 \, (c)$$

$$\sum_{i=1}^{n} x_i = 1+0+1+0+1+1+1+1 = 6.$$

Maximum no. of containers loaded = 6.

2) suppose you have 7 containers whose weights are 50, 10, 30, 20, 70, 60 & 5 & a ship whose capacity is 110, find an optimal solution to this instance of container loading problem.

Container	1	2	3	4	5	6	7
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7
	50	10	30	20	70	60	5

Load the container C7 with $w_7 = 5$

so the, total weight $= 0+5 = 5 < C$

so load the container C7 and make $x_7 = 1$

stop	loaded containers	weight	Total weight	x_i
1	7	$w_7 = 5$	$0+5 = 5 < 110$	$x_7 = 1$
2	2	$w_2 = 10$	$5+10 = 15 < 110$	$x_2 = 1$
3	4	$w_4 = 20$	$15+20 = 35 < 110$	$x_4 = 1$
4	3	$w_3 = 30$	$35+30 = 65 < 110$	$x_3 = 1$
5.	1	$w_1 = 50$	$65+50 = 115 > 110$	$x_1 = 0$

stop the loading process as the total weight exceeds the capacity, c.

container	1	2	3	4	5	6	7
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7
	50	10	30	20	70	60	5
x_i	0	1	1	1	0	0	1

$$\sum_{i=1}^{n} x_i w_i = 0 + 1\times10 + 1\times30 + 1\times20 + 1\times5$$

$$= 10 + 30 + 20 + 5$$

$$= 65 \leq 110 \ (C)$$

$$\sum_{i=1}^{n} x_i = 1+1+1+1 = 4$$

Maximum no. of containers loaded = 6.

3) solve the Container loading problem for a ship with maximum capacity of 175 & 7 containers with weights given below.

	50	10	30	20	70	60	5
x_i	0	1	1	1	0	0	1

$$\sum_{i=1}^{n} x_i w_i = 0 + 1 \times 10 + 1 \times 30 + 1 \times 20 + 1 \times 5$$

$$= 10 + 30 + 20 + 5$$

$$= 65 \leq 110 \ (C)$$

$$\sum_{i=1}^{n} x_i = 1 + 1 + 1 + 1 = 4$$

Maximum no. of containers loaded = 6.

3) solve the container loading problem for a ship with maximum capacity of 175 & 7 containers with weights given below.

Container	1	2	3	4	5	6	7
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7
	80	35	20	30	45	60	25

Load the Container C3 with $w_3 = 20$

so the total weight $= 0 + 20 = 20 < C$

so load the container C3 and make $x_3 = 1$

step	loaded containers	weight	Total weight	x_i
1	3	$w_3 = 20$	$0 + 20 = 20 < 175$	$x_3 = 1$
2	7	$w_7 = 25$	$20 + 25 = 45 < 175$	$x_7 = 1$
3	4	$w_4 = 30$	$45 + 30 = 75 < 175$	$x_4 = 1$
4	2	$w_2 = 35$	$75 + 35 = 110 < 175$	$x_2 = 1$
5	5	$w_5 = 45$	$110 + 45 = 155 < 175$	$x_5 = 1$
6	6	$w_6 = 60$	$155 + 60 = 215 > 175$	$x_6 = 0$

stop the loading process as the total weight exceeds the capacity, C.

container	1	2	3	4	5	6	7
weight	w_1	w_2	w_3	w_4	w_5	w_6	w_7
	80	35	20	30	45	60	25
x_i	0	1	1	1	1	0	1

$$\sum_{i=1}^{n} x_i w_i = 0 + 1 \times 35 + 1 \times 20 + 1 \times 30 + 1 \times 45 + 0 + 1 \times 25$$

$$= 35 + 20 + 30 + 45 + 25 = 155$$

$$\sum_{i=1}^{n} x_i = 1+1+1+1+1 = 5$$

28/3/23

optimal binary search tree

To construct an optimal binary search tree that has the minimum no. of comparisons for a ~~the~~ key value searched in the tree.

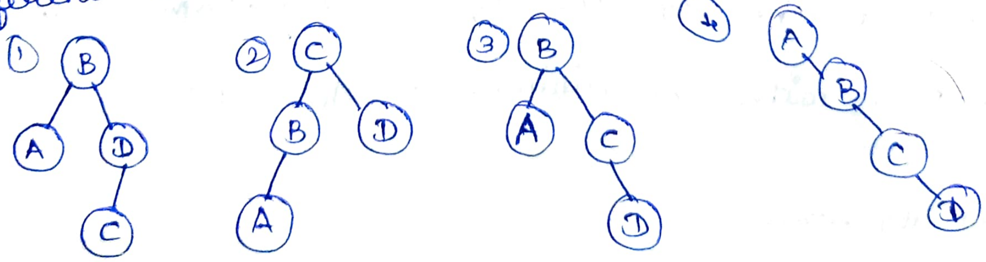
The binary search tree comparisons depend on the position of the key. and the probability of searching for that key.

Unsuccessful searches must also be considered.

Application

1. To implement ~~for~~ file search.
2. google search.
3. a directory.
4. If $n = 4$, a BST with 4 nodes, it can have 14 different BST.

Eg



5. If the search probability are

prob(A) = 0.1

prob(B) = 0.2

prob(C) = 0.4

prob(d) = 0.3

then for tree 1, the total no. of comparisons =

$2 \times 0.1 + 1 \times 0.2 + 3 \times 0.4 + 2 \times 0.3$

$= 0.2 + 0.2 + 1.2 + 0.6$

$= 2.2$

for tree 4, the total no. of comparisons =

$1 \times 0.1 + 2 \times 0.2 + 3 \times 0.4 + 4 \times 0.3$

$= 0.1 + 0.4 + 1.2 + 1.2$

$= 2.9$

for tree 2, the total no. of comparisons =

$3 \times 0.1 + 2 \times 0.2 + 1 \times 0.4 + 2 \times 0.3$

$= 0.3 + 0.4 + 0.4 + 0.6$

$= 1.7$

for tree 3, the total no. of comparisons =

$2 \times 0.1 + 1 \times 0.2 + 2 \times 0.4 + 3 \times 0.3$

$= 0.2 + 0.2 + 0.8 + 0.9$

$= 2.1$

For the above 4 trees the total no. of comparisons vary.

so that the objectives of OBST algorithm is to find the BST with the minimum no. of comparisons.

OBST algorithm

Let $a_1, a_2 \ldots a_n$ be the keys in order.

Let $P_1, P_2 \ldots P_n$ be the probabilities of searching for the keys, respectively.

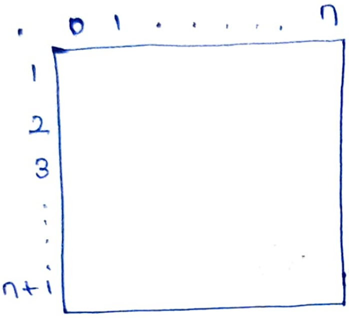
Let $C(i, j)$ be the no. of comparisons made on a average for a successful search in the binary search tree. T_i^j

so the no. of comparisons $c(i,j) = \min\limits_{i \le k \le j} \{ c(i, k-1) + c(k+1, j) + \sum_{s=i}^{j} p_s \}$

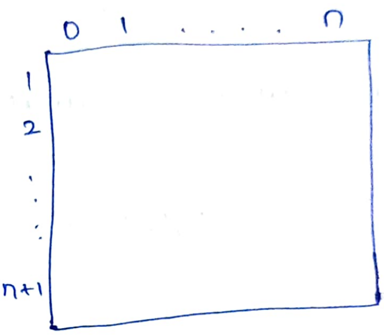
for $1 \le i \le j \le n$

The algorithm uses two tables in matrix form.

1. Main Table.



2. Root Table



n=4

The main table ~~it~~ will give the no. of comparison $c(i,j)$.

The root table will give root for every subtree as $R(i,j)$

start filling the ~~diag~~ diagonal in main table with 0 & the probabilities in the cells just above the diagonal. & The root values as $R(i,i) = i$.

Example

Construct an OBST for 4 nodes A, B, C, D given the probabilities as, $P(A) = 0.1$, $P(B) = 0.2$, $P(C) = 0.4$, $P(D) = 0.3$.

1. Main table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

2. Root table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

The values to be calculated

$C(1,2)$ $\quad C(1,3)$ $\quad C(1,4)$

$C(2,3)$ $\quad C(2,4)$

$C(3,4)$

The values have to be filled in diagonally.

$$C(1,2) = \min \begin{cases} k=1 : C(1,0) + C(2,2) + P_1 + P_2 \\ k=2 : C(1,1) + C(3,2) + P_1 + P_2 \end{cases}$$

$$= \min\{0 + 0.2 + (0.1 + 0.2),\ 0.1 + 0 + (0.1 + 0.2)\}$$

$$= \min(0.5, 0.4) = 0.4$$

$R(1,2) = 2.$

$$C(2,3) = \min \begin{cases} k=2 : C(2,1) + C(3,3) + P_2 + P_3 \\ k=3 : C(2,2) + C(4,3) + P_2 + P_3 \end{cases}$$

$$= \min\{0 + 0.4 + (0.2 + 0.4),\ 0.2 + 0 + (0.2 + 0.4)\}$$

$$= \min\{1.0, 0.8\}$$

$$= 0.8$$

$R(2,3) = 3$

$$C(3,4) = \min \begin{cases} k=3: C(3,2) + C(4,4) + P_3 + P_4 \\ k=4: C(3,3) + C(5,4) + P_3 + P_4 \end{cases}$$

$$= \min\{0 + 0.3 + (0.4 + 0.3),\ 0.4 + 0 + (0.4 + 0.3)\}$$

$$= \min(1, 1.1\}$$

$$= 1.0$$

$$R(3,4) = 3$$

$$C(1,3) = \min \begin{cases} k=1: C(1,0) + C(2,3) + P_1 + P_2 + P_3 \\ k=2: C(1,1) + C(3,3) + P_1 + P_2 + P_3 \\ k=3: C(1,2) + C(4,3) + P_1 + P_2 + P_3 \end{cases}$$

$$= \min(0 + 0.8 + 0.1 + 0.2 + 0.4,\ 0.1 + 0.4 + 0.1 + 0.2 + 0.4,\ 0.4 + 0 + 0.1 + 0.2 + 0.4)$$

$$= \min(1.5, 1.2, 1.1)$$

$$= 1.1$$

$$R(1,3) = 3$$

$$C(2,4) = \min \begin{cases} k=2: C(2,1) + C(3,4) + P_2 + P_3 + P_4 \\ k=3: C(2,2) + C(4,4) + P_2 + P_3 + P_4 \\ k=4: C(2,3) + C(5,4) + P_2 + P_3 + P_4 \end{cases}$$

$$= \min\{0 + 1.0 + 0.2 + 0.4 + 0.3,\ 0.2 + 0.3 + 0.2 + 0.4 + 0.3,\ 0.8 + 0 + 0.2 + 0.4 + 0.3\}$$

$$= \min(1.9, 1.4, 1.7)$$

$$C(2,4) = 1.4$$

$$R(2,4) = 3$$

$$C(1,4) = \min \begin{cases} k=1: C(1,0) + C(2,4) + P_1 + P_2 + P_3 + P_4 \\ k=2: C(1,1) + C(3,4) + P_1 + P_2 + P_3 + P_4 \\ k=3: C(1,2) + C(4,4) + P_1 + P_2 + P_3 + P_4 \\ k=4: C(1,3) + C(5,4) + P_1 + P_2 + P_3 + P_4 \end{cases}$$

$$= \min \left\{ \begin{array}{l} 0 + 1.4 + 0.1 + 0.2 + 0.4 + 0.3, \\ 0.1 + 1.0 + 0.1 + 0.2 + 0.4 + 0.3; \\ 0.4 + 0.3 + 0.1 + 0.2 + 0.4 + 0.3; \\ 1.1 + 0 + 0.1 + 0.2 + 0.4 + 0.3 \end{array} \right\}$$

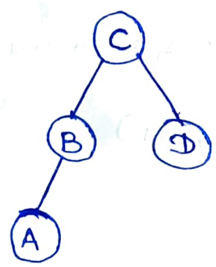
$$= \min(2.4, 2.1, 1.7, 2.1)$$

$C(1,4) = 1.7$

$R(1,4) = 3$

$R(1,4) = 3 \longrightarrow$ Root of the tree

$\left.\begin{array}{l} R(1,2) = 2 \\ R(4,4) = 4 \end{array}\right\}$ Root of the subtree



The average no. of comparison for the constructed BST $= C(1,4) = 1.7$.

29/3/23

Knapsack problem

using greedy technique

Given a set of objects, there weights, there worth the objective is to collect the subset of them in an knapsack. in such a way that the total weight does not exceed the knapsack capacity C & ~~H~~ overall worth is maximised.

1. knapsack problem → The object is considered as a hole - the object has to be

2.

included completely

2. Fractional knapsack → If an object cannot be included completely a portion of it (fractions) can be included in the knapsack.

so that the total weight = C

solved using greedy technique.

parameters.

* n items (or) objects, * knapsack capacity C or W.
* The objects : 1, 2, . . . , n
* The profit / value / worth $P_1, P_2, \ldots P_n$
* The fraction of object included $x_1, x_2, \ldots x_n$.
* The weights of object $w_1, w_2, \ldots w_n$.

solving procedure

1. Compute, P_i / w_i for each item

2. Rearrange the items in decreasing order of $\frac{P_i}{w_i}$.

3. Include the items with the highest P_i/w_i if $w_i \leq$ the remaining capacity.

4. If items i is completely included, $x_i = 1$. otherwise include a fraction m out of w_i in the knapsack & assign $x_i = m/w_i$ (this happens for the last item included)

5. Repeat the above steps until either all or few items are included. so that the total weights becomes equal to W (or) C.

6. Compute $\sum_{i=1}^{n} x_i w_i$ this must be

equal to W (or) C.

compute $\sum_{i=1}^{n} x_i p_i$ to find the overall profit of the items included.

Example

W = 15

Object	1	2	3	4	5	6	7
weight	10	5	15	7	6	18	3
Profit	2	3	5	7	1	4	1

Soln

object	1	2	3	4	5	6	7
~~weight~~ profit	10	5	15	7	6	18	3
~~profit~~ weight	2	3	5	7	1	4	1
Profit / weight	5	1.6	3	1	6	4.5	3

Rearrange the items

object	5	1	6	3	7	2	4
Profit	6	10	18	15	3	5	7
weight	1	2	4	5	1	3	7
Profit / weight	6	5	4.5	3	3	1.6	1

object	Profit p_i	weight w_i	x_i	Remaining capacity
5	6	1	1	15 − 1 = 14
1	10	2	1	14 − 2 = 12
6	18	4	1	12 − 4 = 8
3	15	5	1	8 − 5 = 3
7	3	1	1	3 − 1 = 2
2	5	2	2/3	2 − 2 = 0

$$\sum_{i=1}^{7} x_i w_i = 1\times 2 + \frac{2}{3}\times 3 + 1\times 5 + 1\times 1 + 1\times 4 + 1\times 1$$

$$= 2+2+5+1+4+1$$

$$= 15$$

$$\sum_{i=1}^{7} x_i p_i = 1\times 10 + \frac{2}{3}\times 5 + 1\times 15 + 1\times 6 + 1\times 18 + 1\times 3$$

$$= 10 + 3.3 + 15 + 6 + 18 + 3$$

$$= 55.3$$

The optimal subset of objects included in the knapsack = (1, 2, 3, 5, 6, 7), where a fraction of object 2 is included.

The overall profit of the objects in the knapsack = 55.3