Date: / / /

Huffman Codes

Huffman codes are used to compress a text file. The compression ratio is about 20%-90%. It uses a table of frequencies of occurences of characters in the text file.

Codes: a → 010, b → 01001, c → 01

Fixed length Code	Variable length Code
a → 000	a → 0
b → 001	b → 1
c → 010	c → 00
d → 011	d → 01
e → 100	e → 10
f → 101	f → 11
g → 110	g → 000
h → 111	h → 001
Total size = 8×3 = 24 bits	Total bits = 16 bits

$$\text{Ratio} = \frac{\text{Changes}}{\text{original}} \times 100 \qquad \frac{\text{max code length} - \text{avg code length}}{\text{max code length}} \times 100$$

$$= \frac{24-16}{24} \times 100$$

Q Design a Huffman tree for the following characters -

A	B	C	D	E	F
15	3	45	10	12	9
1	11	0	01	00	10

for fixed length code = 288 bits
for Variable length code = 128 bits

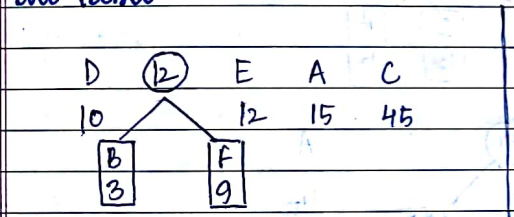
Date: / / /

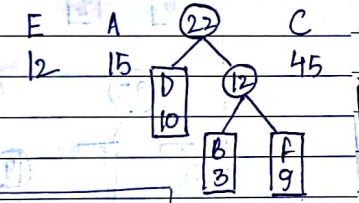
Huffman Tree

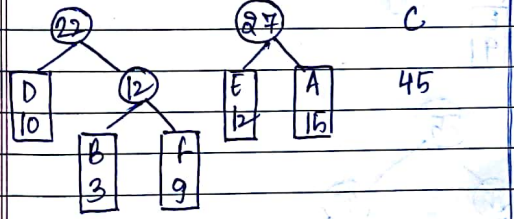
Step 1 Arrange the given characters in increasing order of frequencies.

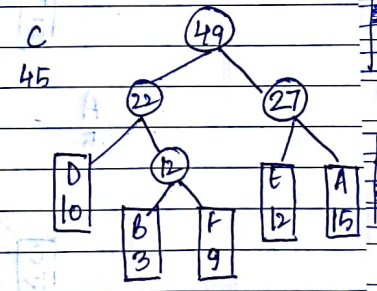
B	F	D	E	A	C
3	9	10	12	15	45

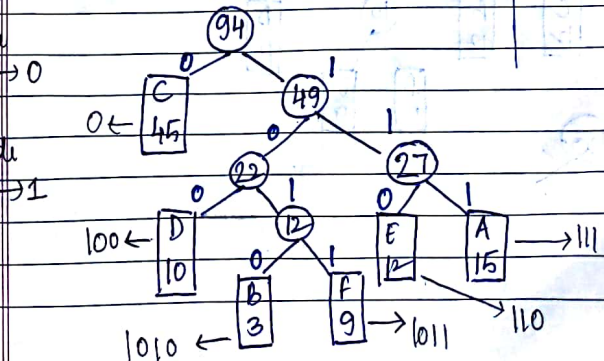
Step 2 Add the two smallest frequencies and rearrange the table







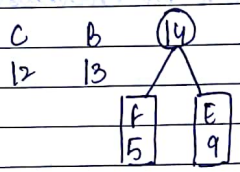


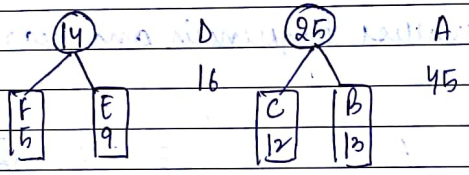


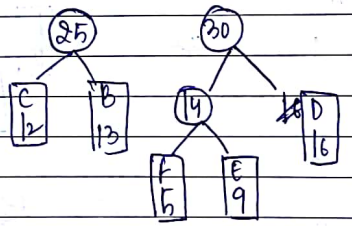
Date : / / /

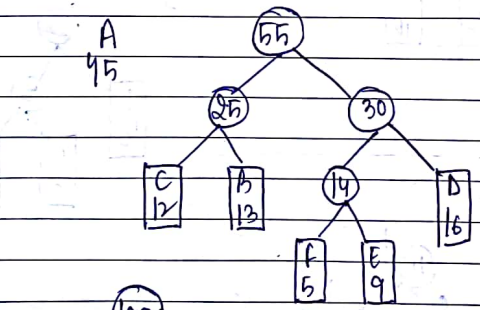
A	B	C	D	E	F
45	13	12	16	9	5

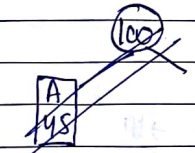
F	E	C	B	D	A
5	9	12	13	16	45

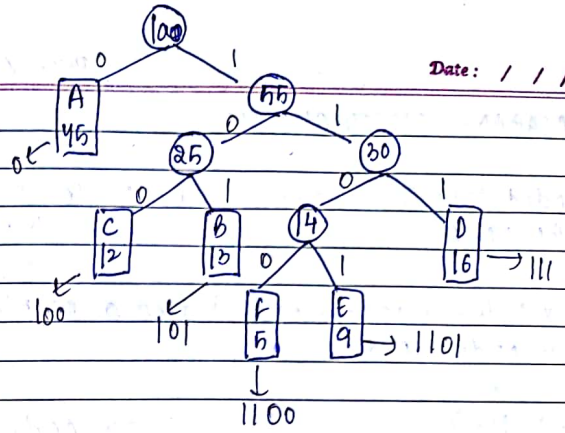










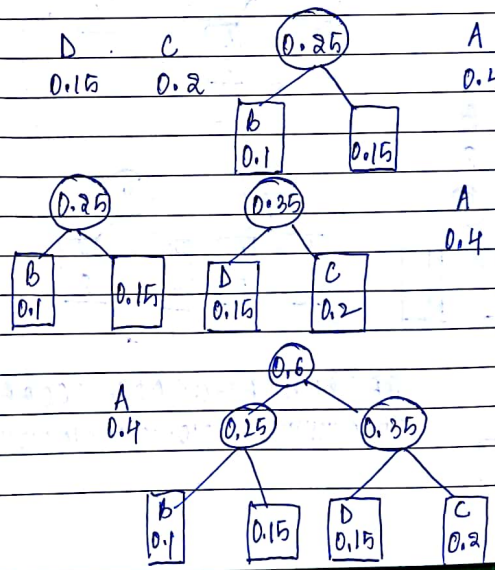


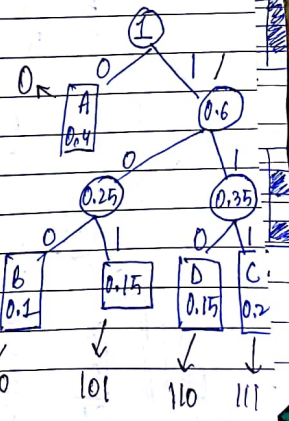
Questions based on Huffman

Character	A	B	C	D	-
Probability	0.4	0.1	0.2	0.15	0.15

- Encode the text ABACABAD using the code.
- Decode the text whose encoding is 100010111001010.

Ascending: B - D C A
0.1 0.15 0.15 0.2 0.4.





Date: / / /

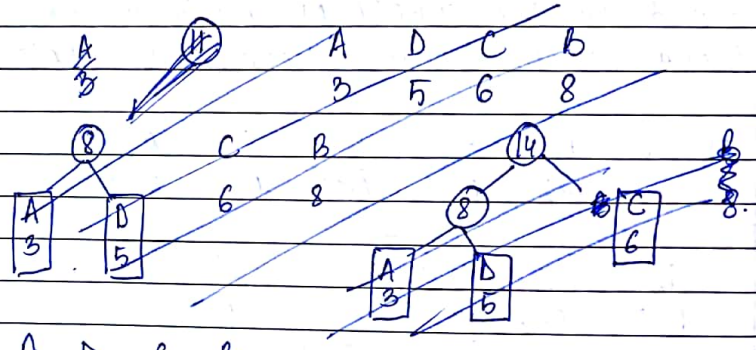
ABACABAD- 0|000|11|0|000|10

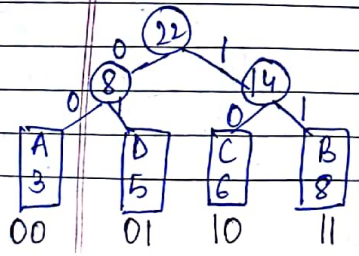
Decoded text for encoded text 1000|0|11|000|10|0
BA-DA-A

Prefix code - no code word is also a prefix of some other code word.

Encode the following string using huffman coding & find the percentage of compression to store the string.
AABBBBBBABCCDDDCCCCBBDD

Char	A	B	C	D
Freq.	3	8	6	5





AABBBBBBABCCDDDCCCCBBDD
0000111111111001101010010101101010101111010101

By using Substitution Method. Date: / / /

$T(n) = 2T(n/2) + n$
$T(n) = O(n \log n)$
$\Rightarrow T(n) \le c \cdot n \log n$
Assume that, $T(n/2) \le c \cdot \frac{n}{2} \log(n/2)$
$T(n) \le \frac{cn}{2} \log \frac{n}{2} + n$

Solve: ① $T(n) = 2T(n/2 + 16) + n$
② $T(n) = T(n/10) + T(9n/10) + n$

Maximum Sum Sub Array Problem

Target array. | 1 | -4 | 3 | 2 |

1 = 1	1 -4 = -3	1 -4 3 = 0	1 -4 3 2 = 2
-4 = -4	-4 3 = -1	-4 3 2 = 1	
3 = 3	3 2 = 5		
2 = 2			

```
left sum = -∞
Right sum = -∞
Sum = 0
for(i=m; i>0; i--)
{ sum = sum + a[i];
   if(sum > left sum)
      left sum = sum;
}
sum = 0;
for(i=m+1; i<n; i++)
{
  sum += a[i];
  if (sum > rightsum)
     rightsum = sum; }
```

```
int function (arr, l, s)
{
  if (n == 1)
     return arr[0]; }
LEFTSUM = function(arr, 0, m);
RIGHTSUM = function(arr, m+1, n);
a = LEFTSUM + RIGHTSUM;

RETURN MAX(LEFTSUM,
  RIGHTSUM, a);
```

$T(n) = 2T(n/2) + n$
⇓
Time Complex
$= 2(2T(n/2))$

Date: / / /

Merge Sort

```
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m-l+1;
    int n2 = r-m;

    int L[n1], R[n2];

    for(i=0; i<n1; i++)
    L[i] = arr[l+i];
    for(j=0; j<n2; j++)
      R[j] = arr[m+1+j];

    i=0;
    j=0;
    k=l;
    while (i<n1 && j<n2)
    {
      if(L[i]<=R[j])
      {
        arr[k] = L[i];
        i++;
      }
    else
    {
      arr[k] = R[j];
        j++;
    }
    while(i<n1)
    {
      arr[k] = L[i];
        i++;
        k++;
    }
    while(j<n2)
    {
      arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[],
 int l, int r)
{
    if(l<r)
    {
    int m = l+(r-1)/2;

    mergeSort(arr, l, m);
    mergeSort(arr, m+1, r);

    merge(arr, l, m, r);
}
```

Date: / / /

Selection Sort

```
void selectionsort(int arr[], int n)
{
    int i, j, min_idx;
    for(i=0; i<n-1; i++)
    {
      min_idx = i;
      for(j=i+1; j<n; j++)
        if(arr[j] < arr[min_idx])
          min_idx = j;
      swap(arr[min_idx], arr[i]);
    }
}
```

$T(n) = T(n-1) + n$

$T(n) \approx n^2$

$\theta(n^2)$

Bubble Sort

```
int c=0;
for(i=0; i<n-1; i++)
{
    for(j=0; j<n-i-1; j++)
    {
      if(arr[j]>arr[j+1])
      {
        swap(arr[j], arr[j+1]);
        c++; }
    if(c==0)
      break;
}
```

$T(n) = T(n-1) + n$

$T(n) \approx n^2$

$\theta(n^2)$

Oct 22, 2018
Monday

Date : / / /

Dynamic Programming (Diff. b/w D&C and dynamic prog.)

1. Dynamic Prog. like the divide and conquer method solves problems by combining the solutions of the sub-problems.
2. D & C algorithm partition the prob. into disjoint sub prob, solve the subprob recursively & then combine their solns to solve the original prob.
3. Dynamic Prog. applies when the subproblems overlap i.e when sub prob. share some prob.
4. D & C solves the prob in top down fashion whereas dynamic prog. solves the prob in bottom up fashion.
5. Dynamic prog. algo. solves each sub prob just once and save the soln in a table, thereby avoiding the work of recomputing the soln every time if the same subproblem occcurs.
6. We apply dynamic prog. to optimization prob.

① Matrix Chain Multiplication
② Longest common subsequence (LCS)
③ 0/1 Knapsack Prob.
④ Optimal BST.

1. Matrix Chain Multiplication

$A_{2\times3} \times B_{3\times4} \Rightarrow$ Total multiplications $\Rightarrow 2\times3\times4 = 24$

$((A_{2\times3}\ B_{3\times4})\ C_{4\times5}) \Rightarrow 2\times3\times4 + 2\times4\times5 = 64$

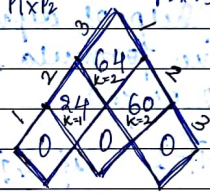
$[A_i]_{P_0\times P_i}$

Date : / / /

Given a chain of matrices $\langle A_i\ A_{i+1}\ A_{i+2} \ldots A_j \rangle$ where matrix A_i is having the dimension of $P_{i-1} \times P_i$ we need to fully parenthesise the product from A_i to A_j in such a way that minimises the no. of scalar multiplications.

$$m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \le k < j} \{ m[i,k] + m[k+1,j] + P_{i-1} \times P_k \times P_j \} & \text{if } i \ne j \end{cases}$$

$A_{2\times3}\ B_{3\times4}\ C_{4\times5}$ $P_0\times P_1$ $P_1\times P_2$ $P_2\times P_3$



$m[1,2] = m[1,1] + m[2,2] + P_0 \times P_1 \times P_2$

$= 0 + 0 + 2\times3\times4 = 24$

$m[2,3] = m[2,2] + m[3,3] + P_1 \times P_2 \times P_3$

$= 0 + 0 + 3\times4\times5 = 60$

$m[1,3] = m[1,1] + m[2,3] + P_0 \times P_1 \times P_3$ (k=1)

$= 0 + 60 + 2\times3\times5 = 90$

$m[1,3] = m[1,2] + m[3,3] + P_0 \times P_2 \times P_3$ (k=2)

$= 24 + 0 + 2\times4\times5 = 64$ [min] ✓

$((A\ B)\ C)$ Ans $\because k=2$

Date: / / /

② Longest Common Subsequence (LCS Problem)

Str = abcdef
Substring = bcd
Subsequence = bdf [not necessary to be continuous]

In LCS problem we are given two sequences.
$X = \langle x_1, x_2, x_3 \ldots x_n \rangle$ and $Y = \langle y_1, y_2, y_3 \ldots y_m \rangle$
we wish to find a max^m length common subsequence of x and y.
To find the soln. with the help of dynamic programming, a recursive formula is used

$$C[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ C[i-1, j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(C[i,j-1], C[i-1,j]) & \text{if } i,j>0 \text{ and } x_i \neq y_j \end{cases}$$

X=ABCDE
Y=AECF
LCS(X,Y) = AC

		Y→	A	E	C	F
X↓	i↓	0	1	2	3	4
	0	0	0	0	0	0
A	1	0	1↖	1←	1←	1←
B	2	0	1↑	1↑	1↑	1↑
C	3	0	1↑	1↑	2↖	2←
D	4	0	1↑	1↑	2↑	2↑
E	5	0	1↑	2↖	2↑	2↑

AC

(AC) ✓

N queen problem

Date...../...../.....

n×n chess board
n queens

The problem is to place n queens on an n×n chess board so that no two queens attack each other by i.e. the same row, column or diagonal.

for ex: 4 queen problem
n=4.

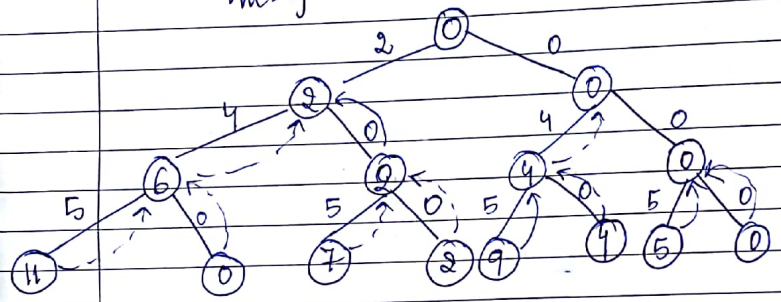
Q Design a st. space tree for 4 queen problem.

Date...../...../.....

Sum of subset problem.

- Given a set of (n) different +ve nos, we need all the subsets of it having sum of its class exactly equal to a given no. (m).

for ex. {2,4,5} or {5,10,12,13,15,18}
m=9



Back tracking Problem

1) N-Queen Problem
2) Sum of subset problem
3) Graph Coloring problem.

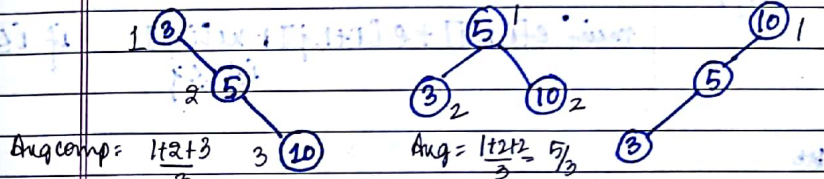
X= ABCBDAB , Y= BDCABA

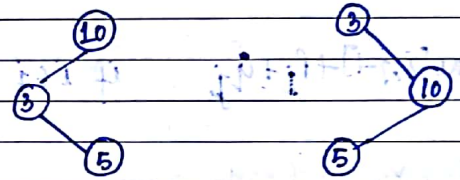
Date: / / /

③ Optimal Binary Search Tree

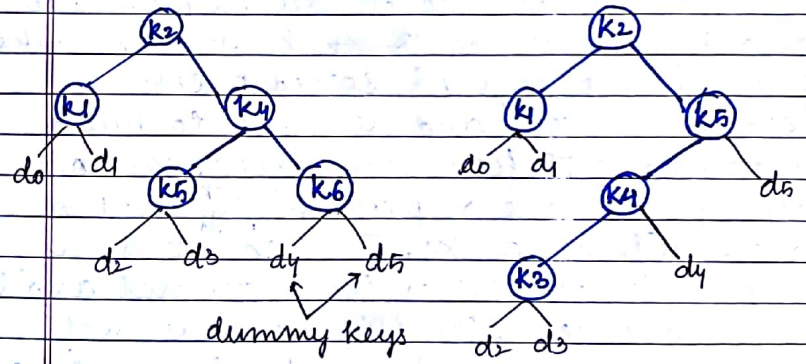
If there are 'n' no. of keys, then no. of binary search tree will be $\frac{2nC_n}{n+1}$

3 keys: 3, 5, 10





n=5, k1, k2, k3, k4, k5



n=5

Date: / / /

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1 \\ \min_{i \le r \le j} \{ e[i,r-1] + e[r+1,j] + w[i,j] \} & \text{if } i \le j \end{cases}$$

$$w[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1 \\ w[i,j-1] + p_j + q_j & \text{if } i \le j \end{cases}$$

there are n+1 dummy keys where n = no. of nodes.

In optimal Binary Search Tree Problem, we are given a sequence $k = \langle k_1, k_2, k_3 \ldots k_n \rangle$ of n distinct keys in sorted order $k_1 < k_2 < k_3 \ldots < k_n$ and we wish to build an optimal BST from these keys. For each key k_i we have a probability p_i that a search will be for k_i. There can be some searches for the values those are not available in the sequence k. For this we have n+1 dummy keys $d_0, d_1, d_2 \ldots d_n$ representing the values. Those are not in k. d_0 represents all the values less than k_1 and d_n represent all the values greater than k_n. For any dummy key d_i it represents all the values between k_i and k_{i+1} $[k_i < d_i < k_{i+1}]$ For each dummy key we have a probability p_i that represents the search will be for d_i.

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$$

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1 \\ \min_{i \le r \le j} \{ e[i,r-1] + e[r+1,j] + w[i,j] \} & \text{if } i \le j \end{cases}$$

$$w[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1 \\ w[i,j-1] + p_j + q_j & \text{if } i \le j \end{cases}$$

i	0	1	2	3
p_i		0.25	0.20	0.15
q_i	0.10	0.15	0.10	0.05

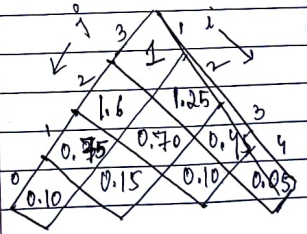
$w[1,1] + p_2 + q_2$

$0.5 + 0.3$

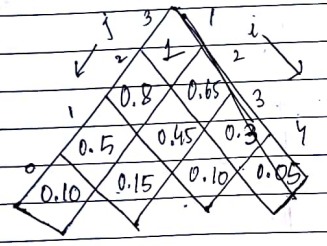
$e[2,2] = e[2,1] + e[3,2] + w[2,2]$

$= 0.15 + 0.10 + 0.45$

$= 0.70$



$e[i,j]$



$w[i,j]$

$e[3,3] = e[3,2] + e[4,3] + w[3,3]$

$= 0.1 + 0.05 + 0.3$

$= 0$

Date: / / /

$$e[1,2] = \begin{cases} k=1 & e[1,0]+e[2,2]+w[1,2] \Rightarrow 0.10+0.70+0.8 = 1.6 \\ k=2 & e[1,1]+e[3,2]+w[1,2] = 1.65 \end{cases}$$

$$e[2,3] = \begin{cases} k=2 & e[2,1]+e[3,3]+w[2,3] = 0.15+0.45+0.65 = 1.25 \\ k=3 & e[2,2]+e[4,3]+w[2,3] = 0.7+0.05+0.65 = 1.4 \end{cases}$$

$$e[1,3] = \begin{cases} k=1 & e[1,0]+e[2,3]+w[1,3] = 0.10+1.25+0.65 = 2.00 \\ k=2 & e[1,1]+e[3,3]+w[1,3] = 0.75+0.45+1 = 2.2 \\ k=3 & e[1,2]+e[4,3]+w[1,3] = 1.6+0.05+1 = 2.65 \end{cases}$$

i	0	1	2	3	4
P_i		0.15	0.1	0.15	0.2
q_i	0.05	0.1	0.05	0.05	0.15

0-1 Knapsack Problem

Date: / / /

$$B[k,w] = \begin{cases} 0 & \text{if } k=0 \text{ or } w=0 \\ B[k-1,w] & \text{if } w < w_k \rightarrow \text{For start pt.} \\ \max\{B[k-1,w],\ B[k-1,w-w_k]+b_k\} & \text{if } w \geq w_k \end{cases}$$

Let's run our algo on the foll. data -

$n=4$ (# of elements)

$w=5$ (max wts.)

Elements (wt, benefits)

(2,3), (3,4), (4,5), (5,6)

k=1 k=2 k=3 k=4

$w < w_1$ i.e. $2 < 2$

$w < w_2$ i.e. $1 < 3$

k →

w ↓	0	1	2	3	4	
0	0	0	0	0	0	
1	0	0	0	0	0	$i=3$
2	0	0	3	3	3	$b_i=5$
3	0	3	4	4	4	$w_i=4$
4	0	3	4	5	5	$w=5$
5	0	3	7	7	7	

k=1, w=3 (1,3) ✗

$B[1,1] = B[0,1]$

$B[1,2] = B[0,2]$

$B[1,2] =$

$B[0,2]$ $B[0,0]+3$

0 3.

$b[k-1, w-w_k]$

Start Point

<1,1,0,0>

Nov 26, 18
Monday

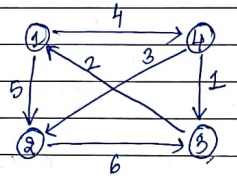
Date: / / /

All Pair Shortest Path

Imp — **Floyd Warshall Algorithm**

All Pair Shortest path algo is used to compute the shortest path from each vertex v to every other vertex u. In this we wish to compute the output in a tabular form where the entry in the a^{th} row and b^{th} column should be the wt. of the shortest path b/w A and B.

Floyd Marshall Algo is used to compute the shortest path in a weighted directed graph. In this negative wt. edges are allowed but there must be no negative cycle. The running time is $\theta(V^3)$ where V = no. of vertices.



Λ = Nil

d_{ij}^0 =

	1	2	3	4
1	0	5	∞	4
2	∞	0	6	∞
3	2	∞	0	∞
4	∞	3	1	0

P_{ij}^0 =

	1	2	3	4
1	Λ	1	Λ	1
2	Λ	Λ	2	Λ
3	3	Λ	Λ	Λ
4	Λ	4	4	Λ

d_{ij}^1 = (node 1 should be intermediate for calculating)

	1	2	3	4
1	0	5	∞	4
2	∞	0	6	∞
3	2	(7)	0	(6)
4	∞	3	1	0

P_{ij}^1 =

	1	2	3	4
1	Λ	1	Λ	1
2	Λ	Λ	2	Λ
3	3	1	1	1
4	Λ	4	4	Λ

Second last node visited

d_{ij}^2 =

	1	2	3	4
1	0	5	(11)	4
2	∞	0	6	∞
3	2	7	0	6
4	∞	3	1	0

P_{ij}^2 =

	1	2	3	4
1	Λ	1	2	1
2	Λ	Λ	2	Λ
3	3	1	Λ	1
4	Λ	4	4	Λ

d_{ij}^3 =

	1	2	3	4
1	0	5	11	4
2	(8)	0	6	(12)
3	2	7	0	6
4	(3)	3	1	0

P_{ij}^3 =

	1	2	3	4
1	Λ	1	2	1
2	3	Λ	2	1
3	3	1	1	1
4	3	4	4	Λ

d_{ij}^4 =

	1	2	3	4
1	0	5	(5)	4
2	8	0	6	12
3	2	7	0	6
4	3	3	1	0

P_{ij}^4 =

	1	2	3	4
1	Λ	1	4	1
2	3	Λ	2	1
3	3	1	1	1
4	3	4	4	Λ

To find distance b/w (1) to (3)

(1) → (4) → (5)

4 will be second last node

Last node 1

Recursive formula for

$$d_{ij}^k = \begin{cases} w_{ij} & \text{if } k=0 \\ \min\left\{ d_{ij}^{k-1},\ d_{ik}^{k-1} + d_{kj}^{k-1} \right\} & \text{if } k \geq 1 \end{cases}$$

$d_{ij}^{k-1} + d_{ik}^{k-1} + d_{kj}^{k-1}$

$d_{ij}^{k-1} \leq d_{ik}^{k-1} + d_{kj}^{k-1}$ if d_{ij}^{k-1}

$d_{ij}^{k-1} > d_{ik}^{k-1} + d_{kj}^{k-1}$ if d_{ij}^{k-1}

P_{ij}^{k-1} P_{ij}^{k-1}

P_{ij}^k =

$$w_{ij} = \begin{cases} 0 & \text{if } i=j \\ \infty & \text{if } i \neq j \text{ and } (i,j) \notin E \\ w(i,j) & \text{if } i \neq j \text{ and } (i,j) \in E \end{cases}$$

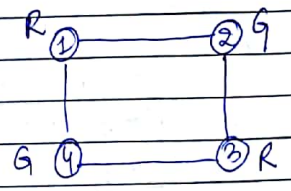
$$P_{ij}^0 = \begin{cases} \Lambda\ (\text{NIL}) & \text{if } i=j \text{ or } w_{ij}=\infty \\ i & \text{if } i \neq j \text{ and } w_{ij} \neq \infty \end{cases}$$

Date: / / /

GRAPH COLORING PROBLEM

No adjacent vertices can have same colour.
Task is ~~the~~ to reduce the no. of colours

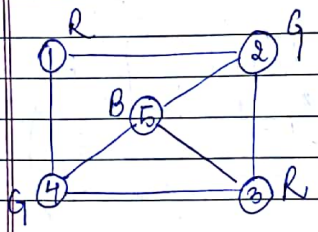
Example



$m = 3$

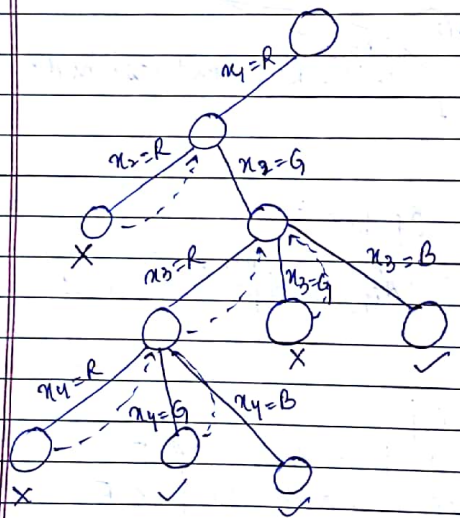
$\{R, G, B\}$

Chromatic no = 2



Chromatic no = 3

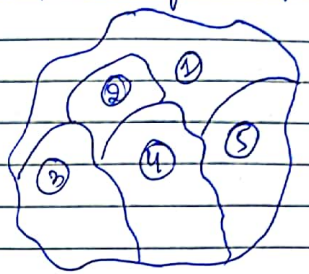
Decision Tree

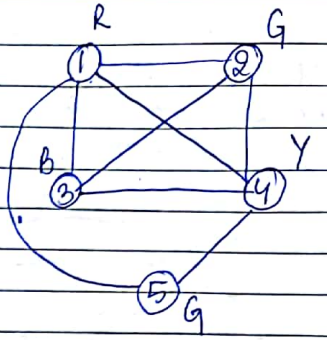


Application

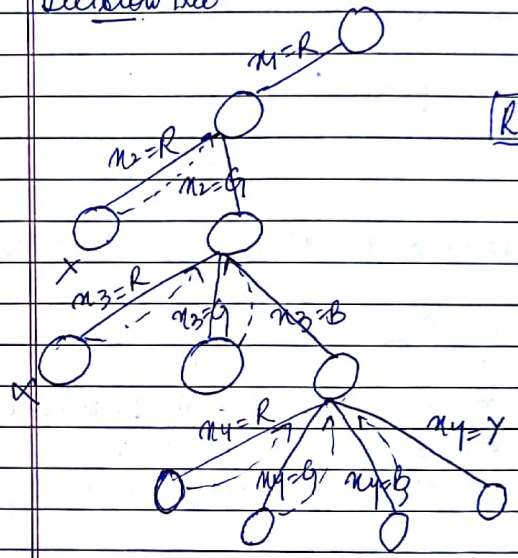
- MAP coloring

Map Colouring Example





Decision Tree



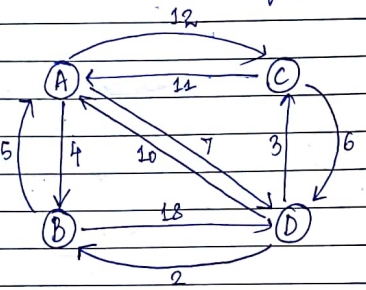
R, G, B, Y, G

1. Intro. to Branch and Bound
2. Diff b/w Branch and Bound & dynamic programming
3. Diff. b/w Branch and Bound and Backtracking

Travelling Salesman Problem

Date: / / /

Given a set of cities and distance b/w every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once & returns to the starting point.



Step 1: Initial cost Matrix and Red^n of Initial Cost Matrix

Initial Cost Matrix

	A	B	C	D
A	∞	4	12	7
B	5	∞	∞	18
C	11	∞	∞	6
D	10	2	3	∞

$C_{ii} = \infty$

$C_{ij} = \infty$, if there is no direct path b/w city i and city j

∞	4	12	7	4
5	∞	∞	18	5
11	∞	∞	6	6
10	2	3	∞	2

Row Red^n →

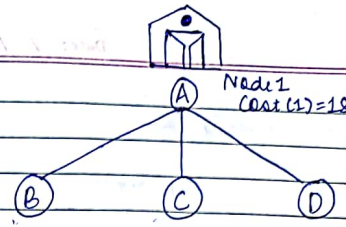
∞	0	8	3
0	∞	∞	13
5	∞	∞	0
8	0	1	∞

0 0 1 0

↓ Column Red^n

∞	0	7	3
0	∞	∞	13
5	∞	∞	0
8	0	0	∞

Cost of A = 4+5+6+2+0+0+1+0
= 18



Step 2: Choosing to go to vertex B: Node 2 (Path A→B)
- From the reduced matrix of Step 1 M[A,B] = 0
- Set row A and column B to ∞
- Set M[B,A] = ∞
- Now, resulting cost matrix is -

∞	∞	∞	∞
∞	∞	∞	13
5	∞	∞	0
8	∞	0	8

- Now, we reduce the matrix and find the cost of node 2

We have

∞	∞	∞	∞	-
∞	∞	∞	13	13
5	∞	∞	0	-
8	∞	0	∞	-

After row Red^n →

∞	∞	∞	∞
∞	∞	∞	0
5	∞	∞	0
8	∞	0	∞

5 - 0 0

↓ After column Red^n

∞	∞	∞	∞
∞	∞	∞	0
0	∞	∞	0
3	∞	0	∞

Thus cost of node 2 is
Cost(2) = Cost(1) + Red^n + M[A,B]
= 18 + 18 + 0
= 36

Step 3: Choosing to go to vertex C: Node 3 (Path A→C)
- From the reduced matrix of Step 1 M[A,C] = 7
- Set row A and col C to ∞

Date: / / /

- Set M[C,A] = ∞
- Now, resulting cost matrix B-

∞	∞	∞	∞
0	∞	∞	13
∞	∞	∞	0
8	0	∞	∞

This matrix is already reduced.

Cost of node 3 is -

Thus Cost(3) = cost(1) + Redn + M[A,C]

= 18 + 0 + 7 = 25

A → C → D

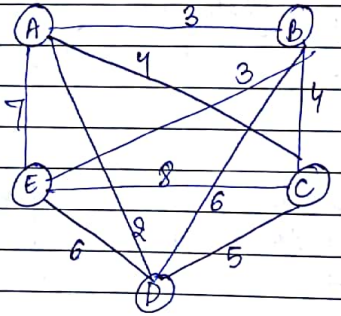
M[C,D] = 0

Set row C and column D = ∞

Set M[D,A] = ∞

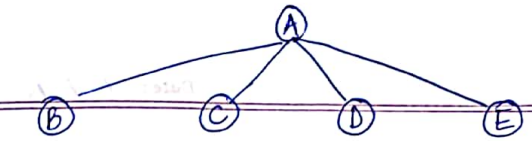
Q Draw a soln space tree. Consider first vertex as start vertex. [10 Marks]

1) Last Year Ques
2)
3)



Q Draw a soln state space tree to solve the following instance of the subset sum problem using backtracking set = {3, 5, 7, 10, 12, 15} , sum = 15

[10 Marks]



Date: / / /

	A	B	C	D	E	
A	∞	3	4	2	7	2
B	3	∞	4	6	3	3
C	4	4	∞	5	8	4
D	2	6	5	∞	6	2
E	7	3	8	6	∞	3

→

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

0 0 1 0 0

↓

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

Cost of A = 2+3+4+2+3+1

= 15

10. a) A

	A	B	C	D	
A	∞	10	15	20	10
B	5	∞	9	10	5
C	6	13	∞	12	6
D	8	8	9	∞	8

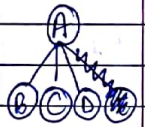
→

∞	0	5	10	
0	∞	4	5	
0	7	∞	6	
0	0	1	∞	

0 0 1 5

→

∞	0	4	5	
0	∞	3	0	
0	7	∞	1	
0	0	0	∞	



Cost of A = 10+5+6+8+1+5 = 35.

Path A → B

M[A,B] = 0 Set row A col B ∞ and Set M[B,A] = ∞.

∞	∞	∞	∞	-
∞	∞	3	0	0
0	∞	∞	1	0
0	∞	0	∞	0

→

∞	∞	∞	∞	
∞	∞	3	0	
0	∞	∞	1	
0	∞	0	∞	

0 - 0 0

→

∞	∞	∞	∞	
∞	∞	3	0	
∞	∞	∞	1	
0	∞	0	∞	

Cost of B = 35+0+0

= 35.

Date: / / /

Path A → C

M[A,C] = ~~4~~.

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 0 \\ \infty & 7 & \infty & 1 \\ 0 & 0 & \infty & \infty \end{bmatrix} \begin{matrix} - \\ 0 \\ 1 \\ 0 \end{matrix} \rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 0 \\ \infty & 6 & \infty & 0 \\ 0 & 0 & \infty & \infty \end{bmatrix}$$

0 0 − 0.

Cost of C = ~~41~~ 35+4+1 = 40.

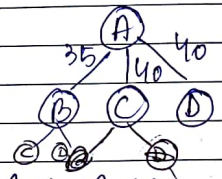
Path A → D

M[A,D] = 5.

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty \\ 0 & 7 & \infty & \infty \\ \infty & 0 & 0 & \infty \end{bmatrix} \begin{matrix} - \\ 0 \\ 0 \\ 0 \end{matrix}$$

0 0 0 −

Cost = 35+5 = 40.



Path A → C → B. Path A → B → A → B → C

M[C,B] = M[B,C] = 3 Set M[C,A] = ∞.

$$\begin{bmatrix} \infty & 0 & \infty & 5 \\ \infty & \infty & \infty & \infty \\ \infty & 7 & \infty & 1 \\ 0 & 0 & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ - \\ 1 \\ 0 \end{matrix} \rightarrow \begin{bmatrix} \infty & 0 & \infty & 5 \\ \infty & \infty & \infty & \infty \\ \infty & 6 & \infty & 0 \\ 0 & 0 & \infty & \infty \end{bmatrix}$$

0 0 − 0

Cost = 35+3+1 = 39.

Path A → B → D.

M[B,D] = 0 Set M[D,A] = ∞

$$\begin{bmatrix} \infty & 0 & 4 & \infty \\ \infty & \infty & \infty & \infty \\ 0 & 7 & \infty & \infty \\ \infty & 0 & 0 & \infty \end{bmatrix} \begin{matrix} 0 \\ - \\ 0 \\ 0 \end{matrix}$$

0 0 0 −

Cost = 35+0+0 = 35.

A → B → D → C → A

Path A → B → D → C

Date: / / /

M[D,C] = 0 Set M[C,A] = ∞.

$$\begin{bmatrix} \infty & 0 & \infty & 5 \\ 0 & \infty & \infty & 0 \\ \infty & 7 & \infty & 1 \\ \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 1 \\ - \end{matrix} \rightarrow \begin{bmatrix} \infty & 0 & \infty & 5 \\ 0 & \infty & \infty & 0 \\ \infty & 6 & \infty & 0 \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

0 0 − −

Cost = 35+1+0 = 36

Path A → B → C.

M[B,C] = 3 Set M[C,A] = ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 1 \\ 0 & \infty & \infty & \infty \end{bmatrix} \begin{matrix} - \\ - \\ 1 \\ 0 \end{matrix} \rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 \\ 0 & \infty & \infty & \infty \end{bmatrix}$$

0 − − 0

Cost = 35+1+3 = 39.

Path A → B → D.

M[B,D] = 0 Set M[D,A] = ∞.

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty \end{bmatrix} \begin{matrix} - \\ - \\ 0 \\ 0 \end{matrix}$$

0 − 0 −

Cost = 35+0+0 = 35.

Path A → B → D → C.

M[D,C] = 0 Set M[C,A] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} - \\ 0 \\ 7 \\ - \end{matrix} \rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

~~35+7 = 42~~

35