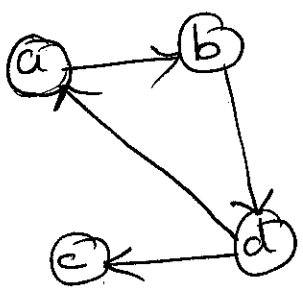
UNIT - III

Dynamic programming

Dynamic programming is a technique for solving problems with overlapping subproblems. It suggests solving each of the smaller sub problems only once and recording the results in a table form which we can then obtain a solution to the original problem.

Warshall's Algorithm

→ for computing transitive closure of a directed graph.



Diagraph

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

Transitive closure of a directed graph is the boolean matrix that has 1 in its ith row and jth column if and only if there is a directed path from ith vertex to the jth vertex, otherwise 0.

$R^{(0)}$ =

	a (1)	b (2)	c (3)	d (4)
a (1)	0	1	0	0
b (2)	0	0	0	1
c (3)	0	0	0	0
d (4)	1	0	1	0

$k = 1$

$R[1,1] = 0$ or $R[0,1]$ and $R[1,1]$

$i\ j$

$= 0$ or $0 \& 0 = 0$ $\quad R[1,1] = 0$

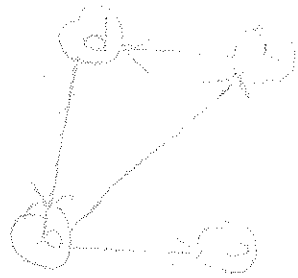
$R[1,2] = 1$ or $[R[0,1] \& R[1,2]]$

$i\ j$

$= 1$ or $[0 \& 1] = 1$ $\quad R[1,2] = 1$

$R[1,3] = 0$ or $[0 \& 0] = 0$ $\quad R[1,3] = 0$

$R[1,4] = 0$ or $[0 \& 0] = 0$ $\quad R[1,4] = 0$



$R[2,1] = 0$ or $0 \& 0 = 0$

$R[2,2] = 0$ or $0 = 0$

$R[2,3] = 0$ or $0 = 0$

$R[2,4] = 1$ or $0 \& 0 = 1$

$R[3,1] = 0$ or $0 = 0$

$R[3,2] = 0$ or $0 = 0$

$R[3,3] = 0$ or $0 = 0$

$R[3,4] = 0$ or $0 = 0$

$R[4,1] = 1$ or $1 \& 0 = 1$ ✓

$R[4,2] = 0$ or $1 \& 1 = 1$ ✓

$R[4,3] = 1$ or $= 1$ ✓

$R[4,4] = 0$ or $1 \& 0 = 0$

$R^{(1)} =$

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

K=2

$R(1,1) = 0$ or $1 \& 0 = 0$

$R(1,2) = 1$ or $1 \& 0 = 1$

$R(1,3) = 0$ or $1 \& 0 = 0$

$R(1,4) = 0$ or $1 \& 1 = 1$

$R(2,1) = 0$ or $0 \; 0 = 0$

$R(2,2) = 0$ or $0 \; 0 = 0$

$R(2,3) = 0$ or $0 \; 0 = 0$

$R(2,4) = 1$ or $0 \; 1 = 1$

$R(3,1) = 0$ or $0 \; 0 = 0$

$R(3,2) = 0$ or $0 \; 0 = 0$

$R(3,3) = 0$ or $0 \; 0 = 0$

$R(3,4) = 0$ or $0 \; 1 = 0$

$R(4,1) = 1$ or $1 \; 0 = 1 = 1$

$R(4,2) = 1$ or $1 \; 0 = 1 = 1$

$R(4,3) = 1$ or $1 \; 0 = 1 = 1$

$R(4,4) = 0$ or $1 \; 1 = 1 = 1$

$R^{(2)} =$

0	1	0	1
0	0	0	1
0	0	0	0
1	1	1	1

$K=3$

$$R^{(3)} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$K=4$

$R(1,1) = 0 \vee 1 \wedge 1 = 1$

$R(1,2) = 1 \quad = 1$

$R(1,3) = 0 \vee 1$

$R(1,4) = 1 \quad = 1$

$R(2,1) = 0 \vee 1 \wedge 1 = 1$

$R(2,2) = 0 \vee 1 \wedge 1 = 1$

$R(2,3) = 0 \vee 1 \wedge 1 = 1$

$R(2,4) = 1 \quad = 1$

$R(3,1) = 0 \vee 0 = 0$

$R(3,2) = 0 \vee 0 = 0$

$R(3,3) = 0 \vee 0 = 0$

$R(3,4) = 0 \vee 0 = 0$

$R(4,1) = 1 \quad = 1$

$R(4,2) = 1 \quad = 1$

$R(4,3) = 1 \quad = 1$

$R(4,4) = 1 \quad = 1$

$R^{(4)}$ = Transitive closure

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Algorithm

Warshall (A [1..n, 1..n])

$R^{(0)} \leftarrow A$

For k = 1 to n do

  For i = 1 to n do

    For j = 1 to n do

$R^{(k)}[i,j] = R^{k-1}[i,j]$ (or) $R^{(k-1)}[i,k]$ and $R^{k-1}[k,j]$

return $R^{(n)}$

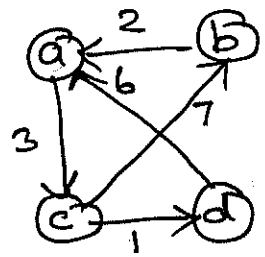
Time efficiency is $O(n^3)$

i/p → adjustancy matrix

o/p → transitive closure

Floyd's algorithm

Given a weighted connected graph (undirected or directed), the all-pairs shortest paths problem to find the distances (the lengths of the shortest paths) from each vertex to all other vertices.



$$W = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & 6 \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

(a) Diagraph (b) weight matrix

$\infty \rightarrow$ no direct path.

$0 \rightarrow$ self node

$d_{ij} \rightarrow$ direct path from i^{th} to j^{th}.

i/p → weight matrix

o/p → Distance matrix of the shortest path.

Algorithm

```
Floyd(W[1..n, 1..n])

D ← W
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
```

$$D[i,j] = \min\{D[i,j], D[i,k] + D[k,j]\}$$

return D.

$$D^{(0)} = \begin{array}{c} \\ a\,(1) \\ b\,(2) \\ c\,(3) \\ d\,(4) \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \\ 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$n = 4$

$D^{(4)} = ?$

Self node always 0.

$k = 1$

$$D^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \boxed{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \boxed{9} & 0 \end{bmatrix} \end{array}$$

$D[1,2] = \min\{\infty, 0+\infty\}$

$D[1,3] = \min\{3, 0+3\}$

$D[2,1] = \min\{2, 2+0\} = 2$

$D[2,3] = \min\{\infty, 2+3\} = 5$

$D[4,3] = \min(\infty, 6+3) = 9$

$K=2$

$$D^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \boxed{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

$D[3,1] = \min(\infty, 7+2) = 9$

$K=3$

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

$D(1,2) = \min(\infty, 3+7) = 10$

$D(1,4) = \min(\infty, 3+1) = 4$

$D(2,4) = \min(\infty, 5+1) = 6$

$D(4,2) = \min(\infty, 9+7) = 16$

$k = 4$:

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

$$D(3,1) = \min(9, 1+6) = 7$$

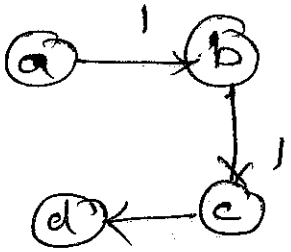
Time efficiency of Floyd's algorithm is cubic.

$$T(n) = O(n^3)$$

Exercise:

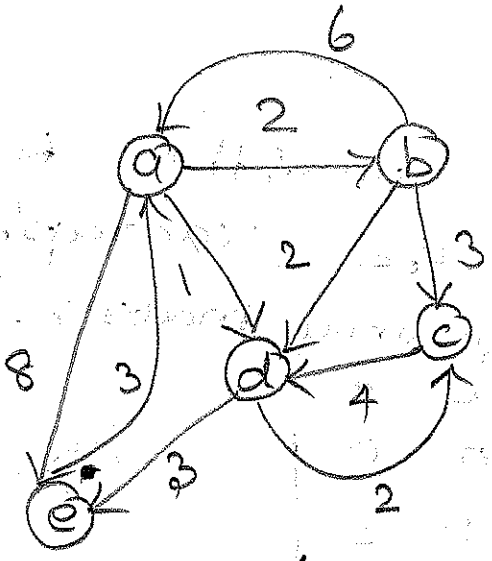
1) Apply Warshall's algorithm to find the transitive closure of the diagraph defined by the following adjacency matrix.

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

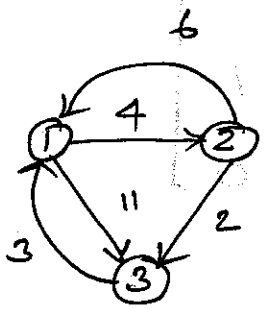


② Solve the all-pairs shortest path problem for the diagraph with the weight matrix

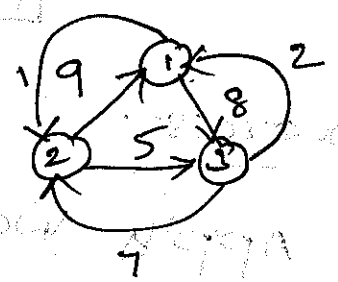
	a	b	c	d	e
a	0	2	∞	1	8
b	6	0	3	2	∞
c	∞	∞	0	4	∞
d	∞	∞	2	0	3
e	3	∞	∞	∞	0



③



④



$$D^{(1)} = \begin{bmatrix} 0 & 1 & 8 \\ 9 & 0 & 5 \\ 2 & 3 & 0 \end{bmatrix}$$

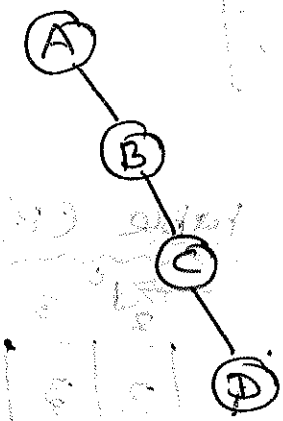
$$D^{(2)} = \begin{bmatrix} 0 & 1 & 6 \\ 9 & 0 & 5 \\ 2 & 3 & 0 \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & 1 & 6 \\ 7 & 0 & 5 \\ 2 & 3 & 0 \end{bmatrix}$$

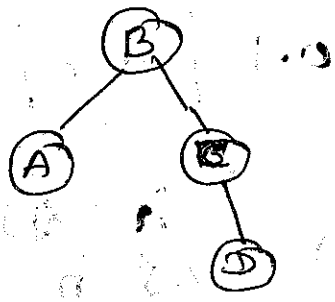
optimal Binary Search Tree

→ minimizing the average number of key comparisons in a search operation.

Ex:



(a) Binary search Tree



(b) optimal binary Search Tree.

Key = D.

no of comparisions in (a) = 4

no of " " (b) = 3.

(b) is optimal than (a)

Construct optimal Binary Search Tree (OBST) for the following key sets.

Key	A	B	C	D
probability	0.1	0.2	0.4	0.3

$n = 4$

Main table (C)

i° \ j°	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
n+1 = 5					0

Root table (R)

i° \ j°	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
n+1 = 5					

$i° \Rightarrow 1$ to $n+1$, $j° \Rightarrow 0$ to n.

→ All diagonal value in C is 0

→ Next to diagonal in C is probability.

→ For single node the same node is root in R.

$d = 1$ to 3 // no. of diagonals

$d = 1$

$c[1,2]$ $\quad c[i,j] = c[i,k-1] + c[k+1,j]$

$i^{\circ}\ j^{\circ}$

k is i to j, $k = 1$ to 2 $\quad p(s) = p(1) + p(2)$

$p(s) = 0.3$.

$k=1$ $\quad c[1,2] = c[1,0] + c[2,2] + p(s)$

$= 0 + 0.2 + 0.3 = 0.5$

$k=2$ ✓ $\quad c[1,2] = c[1,1] + c[3,2] + p(s)$

$= 0.1 + 0 + 0.3 = 0.4$ minimum

$c[1,2]$ is minimum when $k=2$

$c[1,2] = 0.4 \quad R[1,2] = 2$

$c[2,3]$, $k = 2, 3$ $\quad p(s) = p(2) + p(3) = 0.6$

$k=2$ $\quad c[2,3] = c[2,1] + c[3,3] + p(s)$

$= 0 + 0.4 + 0.6 = 1$

$k=3$ ✓ $\quad c[2,3] = c[2,2] + c[4,3] + 0.6$

$= 0.2 + 0 + 0.6 = 0.8$ minimum

$c[2,3] = 0.8 \quad R[2,3] = 3$

$c[3,4]$, $k = 3, 4$ $\quad p(s) = p(3) + p(4) = 0.7$

$k=3$ ✓ $\quad c[3,4] = c[3,2] + c[4,4] + 0.7$

$= 0 + 0.3 + 0.7 = 1$ minimum

$k=4$ $\quad c[3,4] = c[3,3] + c[5,4] + 0.7$

$= 0.4 + 0 + 0.7 = 1.1$

$c[3,4] = 1 \quad R[3,4] = 3$

d=2

c[1,3], K = 1 to 3 K = 1, 2, 3 p(3) = p(1)+p(2)+p(3)

p(3) = 0.7

K=1

$c[1,3] = c[1,0] + c[2,3] + 0.7$

$= 0 + 0.8 + 0.7 = 1.5$

K=2

$c[1,3] = c[1,1] + c[3,3] + 0.7$

$= 0.1 + 0.4 + 0.7 = 1.2$

K=3 ✓

$c[1,3] = c[1,2] + c[4,3] + 0.7$

$= 0.4 + 0 + 0.7 =$ **1.1** minimum

c[1,3] = 1.1 R(1,3) = 3

c[2,4] K = 2, 3, 4, K ← 2 to 4 p(3) = p(2)+p(3)+p(4)

p(3) = 0.9

K=2

$c[2,4] = c[2,1] + c[3,4] + 0.9$

$= 0 + 1.0 + 0.9 = 1.9$

K=3 ✓

$c[2,4] = c[2,2] + c[4,4] + 0.9$

$= 0.2 + 0.3 + 0.9 =$ **1.4** minimum

K=4

$c[2,4] = c[2,3] + c[5,4] + 0.9$

$= 0.8 + 0 + 0.9 = 1.7$

c[2,4] = 1.4 R[2,4] = 3

$d = 3$

$c[1,4]$, $k = 1$ to 4 $p(3) = p(1) + p(2) + p(3) + p(4)$

$= 1.$

$k = 1$

$c[1,4] = c[1,0] + c[2,4] + 1$

$= 0 + 1.4 + 1 = 2.4$

$k = 2$

$c[1,4] = c[1,1] + c[3,4] + 1$

$= 0.1 + 1.0 + 1 = 2.1$

$k = 3$

$c[1,4] = c[1,2] + c[4,4] + 1$

$= 0.4 + 0.3 + 1 = 1.7$ minimum

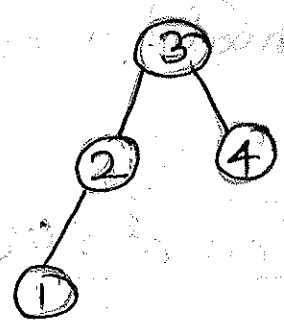
$k = 4$

$c[1,4] = c[1,3] + c[5,4] + 1$

$= 1.1 + 0 + 1 = 2.1$

$c[1,4] = 1.7$, $R[1,4] = 3$

optimal binary search Tree from R



Algorithm

Optimal BST (P[1..n])

```
For i = 1 to n do
    c[i, i-1] = 0
    c[i, j] = p[i]
    R[i, i] = i
c[n+1, n] = 0
For d = 1 to n-1 do
    For i = 1 to n-d do
        j = i+d
        minval = ∞
        For k = i to j do
            if c[i, k-1] + c[k+1, j] < minval
            minval = c[i, k-1] + c[k+1, j]; kmin = k
        R[i, j] = k
        Sum = p[i]; For s = i+1 to j do
                        Sum = Sum + p(s)
        c[i, j] = minval + Sum
return c[1, n], R
```

Analysis

The space efficiency is $O(n^2)$ quadratic

The time efficiency is cubic $T(n) = O(n^3)$

Computing a Binomial Coefficient

Binomial coefficient is denoted by $C(n,k)$ or $\binom{n}{k}$, is the number of combinations (subsets) of k elements from an n-element set $(0 \le k \le n)$.

The following binomial formula contains the "binomial coefficients".

$$(a+b)^n = C(n,0)a^n + \cdots + C(n,i)a^{n-i}b^i + \cdots + C(n,n)b^n.$$

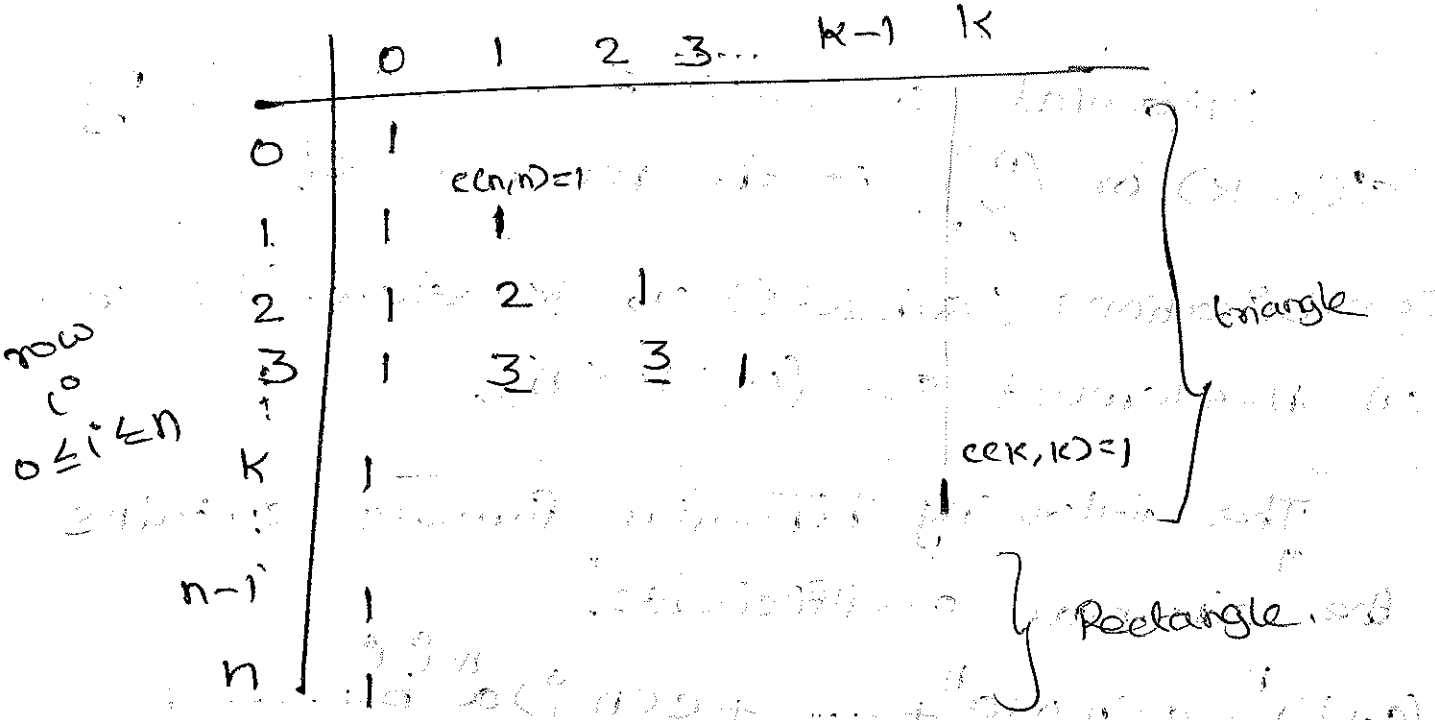
The two properties of binomial coefficients are

1. $C(n,k) = C(n-1,k-1) + C(n-1,k)$, for $n>k>0$
2. $C(n,0) = C(n,n) = 1$

The problem of constructing $C(n,k)$ in terms of the smaller and overlapping problems of computing $C(n-1,k-1)$ and $C(n-1,k)$ (from property 1) by dynamic programming technique.

Record the values of binomial co-efficients in a table of $n+1$ rows & $k+1$ columns. numbered from 0 to n & from 0 to k.

column, j
$0 \le j \le k$



$C(n,k) = C(n-1,k-1) + C(n-1,k), \quad n > k > 0$

$C(2,1) = C(1,0) + C(1,1)$

$= 1 + 1 = 2$

$C(3,1) = C(2,0) + C(2,1)$

$= 1 + 2 = 3$

$C(3,2) = C(2,1) + C(2,2)$

$= 2 + 1$

$= 3$ //

Algorithm

Binomial (n, k)

For $i = 0$ to n do

For $j = 0$ to $\min(i, k)$ do

if $j = 0$ or $j = k$

$C[i, j] = 1$

else

$C[i, j] = C[i-1, j-1] + C[i-1, j]$

return $C[n, k]$

Analysis

Basic operation addition.

$A(n, k)$ = no. of additions made by algorithm in computing $C(n, k)$

First $k+1$ rows of the table form a triangle while remaining $n-k$ rows form a rectangle.

$$A(n,k) = \sum_{i=1}^{k} \sum_{j=1}^{i-1} 1 + \sum_{i=k+1}^{n} \sum_{j=1}^{k} 1$$

$$= \sum_{i=1}^{k} (i-1+1) + \sum_{i=k+1}^{n} (k-1+1)$$

$$= \sum_{i=1}^{k} i - \sum_{i=1}^{k} 1 + k \sum_{i=k+1}^{n} 1$$

$$= \frac{k(k+1)}{2} - (k-1+1) + k(n-k+1-1)$$

$$= \frac{k(k+1)}{2} + k + k(n-k) =$$

$$= \frac{k(k+1)}{2} - k + k(n-k)$$

$$= k\left[\frac{k+1}{2} - 1 + (n-k)\right]$$

$$= k\left[\frac{k+1-2+2n-2k}{2}\right]$$

$$= k\left[\frac{2n-k-1}{2}\right] = \frac{2kn-k^2-k}{2}$$

$$A(n,k) = \frac{2kn-k^2-k}{2}$$

$$\boxed{A(n,k) = O(nk)}$$

---x---

Find OBST

node	A	B	C	D	E
probability	0.2	0.3	0.1	0.3	0.1

Knapsack problem

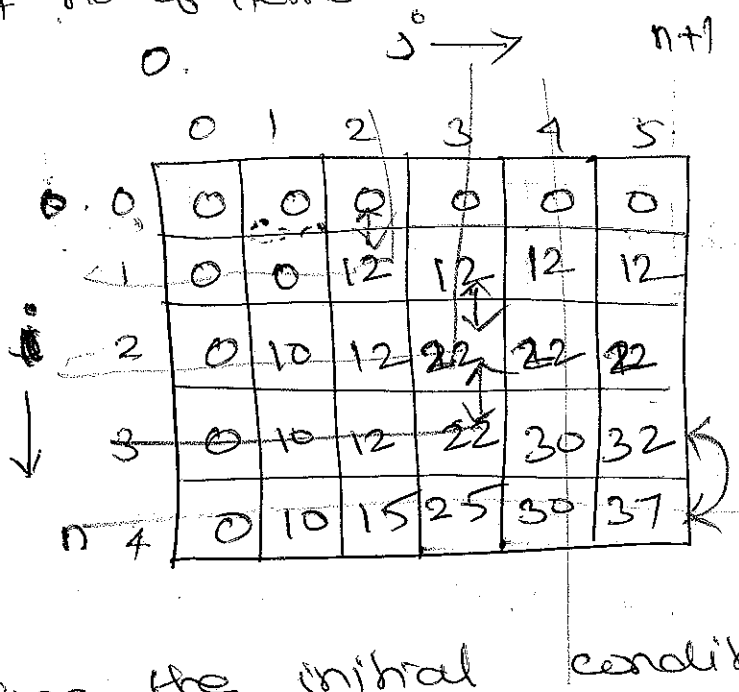
Given n items of known weights $w_1, w_2, \ldots w_n$ and values $v_1, v_2, \ldots v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.

(1)

Item	weight	value/ profit
1	2	12
2	1	10
3	3	20
4	2	15

capacity w=5. Find the optimal solution for knapsack problem?

n=4 no. of items



① Define the inibal condition.

row i^0, $i \to 0$ to n

column j^0, $j \to 0$ to n+1

$P(0, j^0) = 0$, $j^0 >= 0$

$P(i^0, 0) = 0$, $j^0 >= 0$

② calculate $P(i^0, j^0)$

$$P(i^0, j^0) = \max\{P(i^0-1, j^0), P_{i^0} + P(i-1, j-w_{i^0})\}$$

if $j - w_{i^0} \geq 0$.

$P(i, j) = P(i-1, j)$, if $j - w_i < 0$.

① P(1,1), i=1, j=1

$w_1 = 2 \quad P_1 = 12$

$j - w_i = 1 - 2 = -1 < 0$

$P(1,1) = P(0,1) = 0$

$\boxed{P(1,1) = 0}$

② P(1,2) i=1, j=2

$w_1 = 2 \quad P_1 = 12$

$j - w_i = 2 - 2 = 0 \geq 0$

$P(1,2) = \max \{ P(0,2), 12 + P(0,0) \}$

$= \max \{ 0, 12 + 0 \} = 12$

$\boxed{P(1,2) = 12}$

③ P(1,3) i=1, j=3

$w_1 = 2 \quad P_1 = 12$

$j - w_i = 1 \geq 0$

$P(1,3) = \max \{ P(0,3), 12 + P(0,1) \}$

$= \max \{ 0, 12 \} = 12$

$\boxed{P(1,3) = 12}$

④ P(1,4), i=1, j=4

$P_1 = 12 \quad w_1 = 2 \quad j - w_i = 4 - 2 = 2 \geq 0$

$P(1,4) = \max \{ P(0,4), 12 + P(0,2) \}$

$= \max\{0, 12\} = 12$

$P(1,4) = 12$

⑤ $P(1,5)$, $i=1$, $j=5$

$w_1 = 2 \quad P_1 = 12$

$j - w_i = 3 \geq 0$

$P(1,5) = \max\{P(0,5), 12 + P(0,3)\}$

$= \max\{0, 12\} = 12$

$P(1,5) = 12$

⑥ $P(2,1)$ $i=2$, $j=1$

$w_2 = 1 \quad P_2 = 10$

$j - w_i = \geq 0$

$P(2,1) = \max\{P(1,1), 10 + P(1,0)\}$

$= \max\{0, 10+0\} = 10$

$P(2,1) = 10$

⑦ $P(2,2)$ $i=2, j=2$

$w_2 = 1 \quad P_2 = 10$

$j - w_i = 1 \geq 0$

$P(2,2) = \max\{P(1,2), 10 + P(1,1)\}$

$= \max\{12, 10+0\} = 12$

$P(2,2) = 12$

(8) $P(2,3)$ $i=2,\ j=3$

$w_i = 1 \quad P_i = 10$

$j - w_i = 2 \geq 0$

$P(2,3) = \max\{P(1,3), 10 + P(1,2)\}$

$= \max\{12, 10+12\} = 22$

$P(2,3) = 22$

(9) $P(2,4)$ $i=2,\ j=4$

$w_i = 1 \quad P_1 = 10$

$j - w_i = 3 \geq 0$

$P(2,4) = \max\{P(1,4), 10 + P(1,3)\}$

$= \max\{12, 10+12\}$

$P(2,4) = 22$

(10) $P(2,5)$ $i=2\ \ j=5$

$w_2 = 1 \quad P_2 = 10$

$j - w_i = 2 \geq 0$

$P(2,5) = \max\{P(1,5), 10 + P(1,4)\}$

$= \max\{12, 10+12\} = 22$

$P(2,5) = 22$

(11) $P(3,1)$, $i=3$, $j=1$

$w_3=3 \quad P_3=20$

$j-w_i = 1-3=-2 < 0$

$P(i,j) = P(i-1,j)$

$= P(2,1)$

$= 10$

$\boxed{P(3,1)=10}$

(12) $P(3,2)$, $i=3$, $j=2$

$w_3=3 \quad P_3=20$

$j-w_i = 2-3=-1 < 0$

$P(3,2) = P(2,2) = 12$

$\boxed{P(3,2)=12}$

(13) $P(3,3)$ $i=3$, $j=3$

$w_3=3 \quad P_3=20$

$j-w_i = 0 \geq 0$

$P(3,3) = \max\{P(2,3), 20+P(2,0)\}$

$= \max\{22, 20+0\} = 22$

$\boxed{P(3,3)=22}$

(14) $P(3,4)$ $i=3$, $j=4$

$w_3 = 3$ $P_3 = 20$

$j - w_i = 1 \geq 0$

$P(3,4) = \max\{P(2,4),\ 20 + P(2,1)\}$

$= \max\{22,\ 20+10\} = 30$

$\boxed{P(3,4) = 30}$

(15) $P(3,5)$ = $i=3$, $j=5$

$w_3 = 3$ $P_3 = 20$

$j - w_i = 2 \geq 0$

$P(3,5) = \max\{P(2,5), 20 + P(2,2)\}$

$= \max\{22,\ 20+12\} = 32$

$\boxed{P(3,5) = 32}$

(16) $P(4,1)$, $i=4$ $j=1$

$w_i = 2$ $P_i = 15$

$j - w_i = 1-2 = -1 < 0$

$P(4,1) = P(3,1) = 10$

$\boxed{P(4,1) = 10}$

(17) $P(4,2)$ $i=4$, $j=2$

$j - w_i = 2-2 = 0 \geq 0$

$P(4,2) = \max\{P(3,2),\ 15 + P(3,0)\}$

$= \max\{12,\ 15+0\} = 15$

$P(4,2) = 15$

(18) $P(4,3) = \quad i=4 \quad , j=3$

$w_4 = 2 \quad P_4 = 15$

$j - w_i = 3-2 \neq 0$

$P(4,3) = \max \{ P(3,3), 15 + P(3,1) \}$

$= \max \{ 22, 15+10 \} = \max \{22, 25\}$

$P(4,3) = 25$

(19) $P(4,4) \quad i=4 \quad j=4 \quad , w_4 = 2 \quad P_4 = 15$

$j - w_i = 4-2 = 2 \geq 0$

$P(4,4) = \max \{ P(3,4), 15 + P(3,2) \}$

$= \max \{ 30, 15+12 \}$

$= \max \{ 30, 27 \} = 30$

$P(4,4) = 30$

(28) $P(4,5)$ = $i=4$, $j=5$ $w_4=2$ $P_4=15$

$j-w_i = 5-2=3$ $P_4=15$

$P(4,5) = \max\{P(3,5), 15+P(3,3)\}$

$= \max\{32, 15+22\}$

$= \max\{32, 37\}$

$P(4,5)=37$

1. compare $P(4,5)$ with $P(3,5)$

$P(4,5) \neq P(3,5)$

not equal means, item 4 is included in the solution set, then delete 4th row & 4th column. $W_R = 5-2 = 3$ (capacity W − item weight)

item set = {4, 2, 1}

item1	2
item2	1
item4	2

$W=5$

(2) compare $P(3,5)$ and $P(2,3)$

$P(3,3) = P(2,3)$

item 3 is not included in the set. delete only only 3rd row

(3) $P(2,3) \neq P(1,3)$ $W_R = 3-1=2$

include item2. delete 2nd row, 3rd colu

(4) $P(1,2) \neq P(0,2)$, add item1, delete RRC

$W_R = 2-2=0$

Feasible subset = {1, 2, 4}

optimal solution = 37 (last cell in table)

(2)

item	weight	profit
1	2	10
2	4	10
3	6	12
4	9	18

capacity W = 10. Find the feasible subset using dynamic programming.

(3)

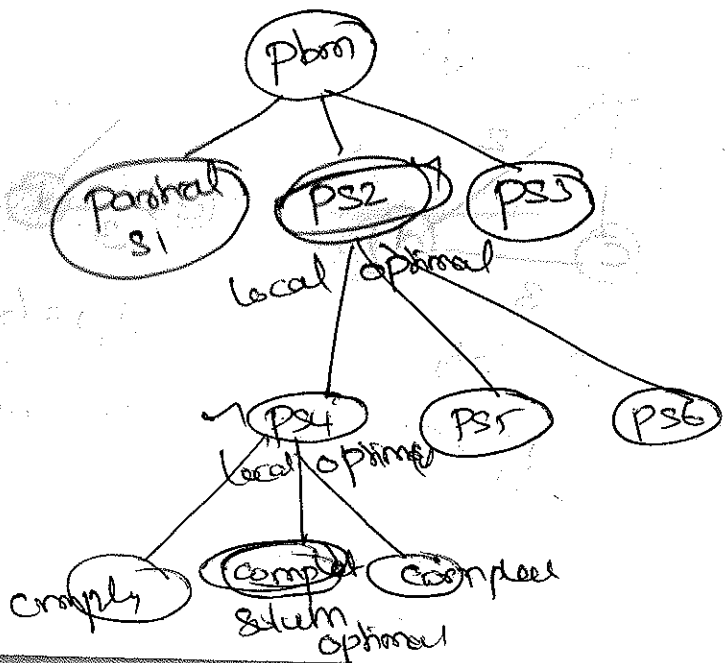
item	weight	value
1	3	25
2	2	20
3	1	15
4	4	40
5	5	50

capacity W = 6.

Greedy Technique

The greedy approach suggests ~~that~~ constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. On each step the choice made must be

1. Feasible → It has to satisfy the problem's constraint
2. locally optimal → it has to be the best local choice among all feasible choices available on that step.
3. Irrevocable - once made, it can't be changed on subsequent steps.



prim's Algorithm

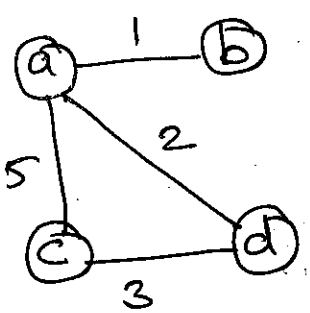
→ to construct minimum spanning tree.

Spanning Tree

A spanning tree of a weighted, connected graph is its connected acyclic subgraph (tree) that contains all the vertices of the graph.

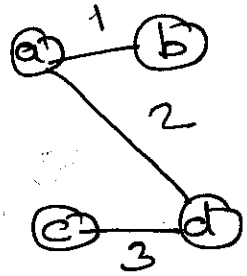
Minimum Spanning Tree

A minimum spanning tree of a weighted connected graph is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges.



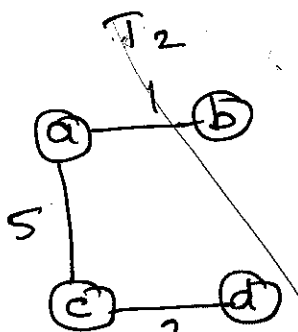
graph

T_1



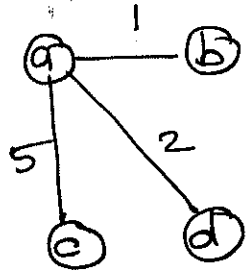
$W(T_1) = 6$

T_2



$W(T_2) = 9$

T_3

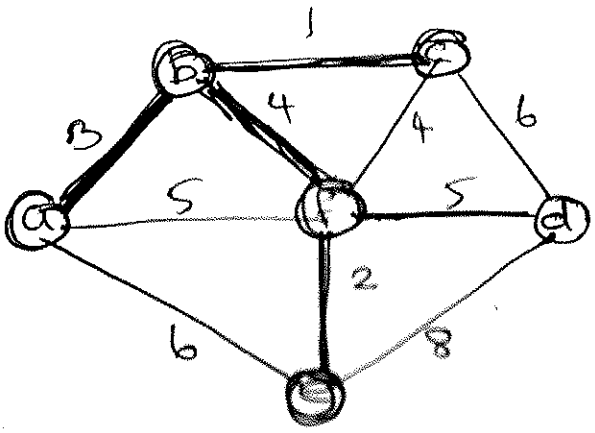


$W(T_3) = 8$

T_1, T_2, T_3 → spanning tree

T_1 → minimum spanning tree

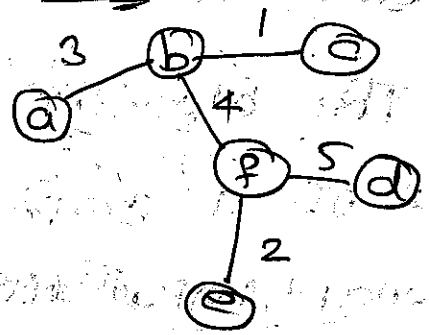
(1)



Tree vertices	Remaining vertices (minimum → b(a,3))
a(–,–)	b(a,3) f(a,5), e(a,6) c(–,∞), d(–,∞)
b(a,3)	c(b,1) f(b,4), e(a,6) d(–,∞)
c(b,1)	d(c,6), f(b,4), e(a,6)
f(b,4)	d(f,5), e(f,2)
e(f,2)	d(f,5)
d(f,5)	

Minimum Spanning Tree

MST



W(T) = 15

Algorithm prim(G)

prim(G)

$V_T = \{v_0\}$

$E_T = \phi$

for $i = 1$ to $|V| - 1$ do

find a minimum weight edge $e^* = (v^*, u^*)$

among all the edges (v, u), such that

v is in V_T and u is in $V - V_T$.

$V_T = V_T \cup \{u^*\}$

$E_T = E_T \cup \{e^*\}$

return E_T

Analysis

1. If a graph is represented by its weight matrix and the priority queue is implemented as an ordered array,

The algorithm's running time is $O(|V|^2)$.

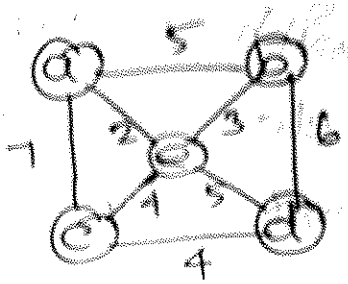
2. If a graph is represented by its adjacency linked lists and the priority queue is

implemented as a min-heap.

Running time of algorithm is $O(|E| \log |V|)$

$$T(n) = O(|E| \cdot \log |V|)$$

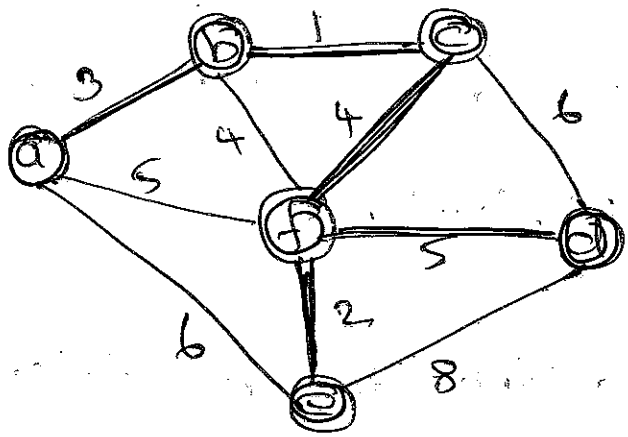
(2) Apply prim's algorithm to the following graph.



Kruskal's Algorithm

→ to find minimum spanning tree.

(1) Apply Kruskal's algorithm to find minimum spanning tree for the following graph



① write the edges with weight

ab	bc	cd	de	ea	af	bf	ef
3	1	6	8	6	5	4	4
*	X	*	*	*	*	*	*

df	ef
5	2
*	X

② Sort the edges in ascending order based on their e

bc	ef	ab	bf	ef	af	df	cd	ea
1	2	3	4	4	5	5	6	6
					X		X	X

cd
8

Tree edges

① select bc, no cycle, add in tree edges

1

(2) Selat cf, no cycle, add in tree edges

(3) Selat ab, no cycle add in tree edges

(4) Selat bf, no cycle, add in tree edges

(5) Selat ef, no cycle, add in tree edges

(6) Select af, Form a cycle, can't be added

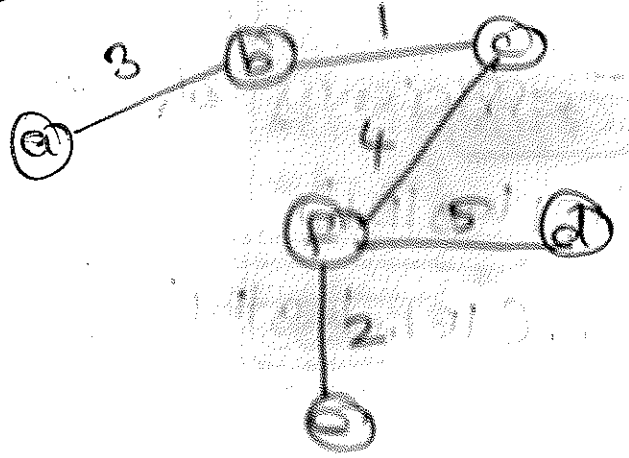
(7) Select df, no cycle, add in tree edges

(8) Select cd, Form a cycle, no add

(9) Selat ea, cycle,

(10) Selat ed, cycle.

Minimum Spanning Tree



Total weight = 15,

Algorithm

```
kruskal (G)
    Sort E in increasing order of the
    edge weight
    E_T = ϕ
    k = 0 , encounter = 0
    while encounter < |V| - 1
        k = k + 1
        if E_T U {e_ik} is acyclic
            E_T = E_T U e_ik
            encounter = encounter + 1
    return E_T.
```

Analysis

The time efficiency of kruskals algorithm is $O(|E| \log |E|)$.

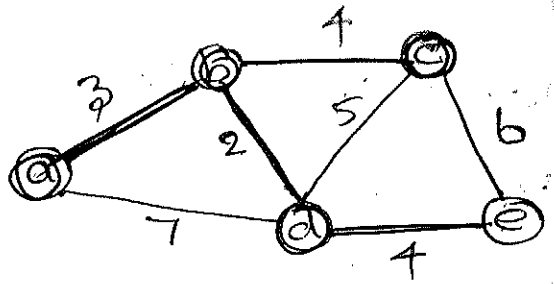
$$TCN = O(|E| \cdot \log |E|)$$

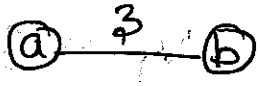
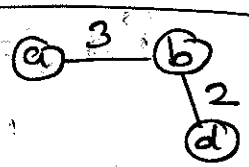
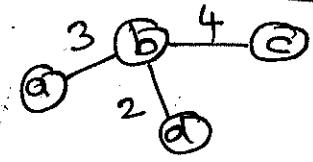
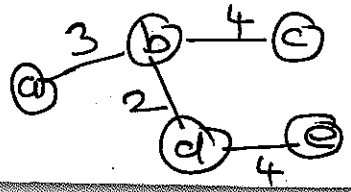
Dijkstra's Algorithm

→ Single source shortest paths problem.

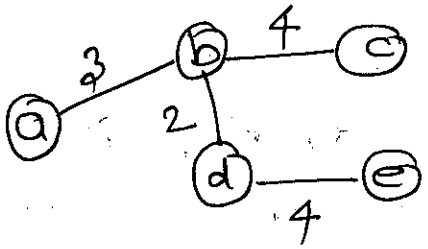
→ For a given vertex called source in a weighted connected graph, find shortest paths to all its other vertices.

① Apply Dijkstra's algorithm to find the shortest path from source node a:



Tree vertices	Remaining vertices	Illustration
a(-,0)	**b(a,3)** d(a,7) c(-,∞) e(-,∞)	
b(a,3)	c(b,7), **d(b,5)** e(-,∞)	
d(b,5)	**c(b,7)** e(d,9)	
e(b,7)	e(d,9)	

Shortest path to all nodes from Source a



Algorithm

```
Dijkstra(G, s)
Initialize (Q)
  For every vertex v in V do
      dv = ∞ , Pv = null
      Insert(Q, V, dv)
  ds = 0
  Decrease (Q , s, ds)
  VT = ϕ
   For i = 0 to |V|-1 do
        u* = DeleteMin(Q)
        VT = VT ∪ u*
        For every vertex u in V-VT that is
            adjacent to u* do
            if du* + w(u*, u) < du
                du = du* + w(u*, u)
                Pu = u*
```

Decrease (Q, v, dv)

Analysis

① weight matrix and priority queue is used for implementation

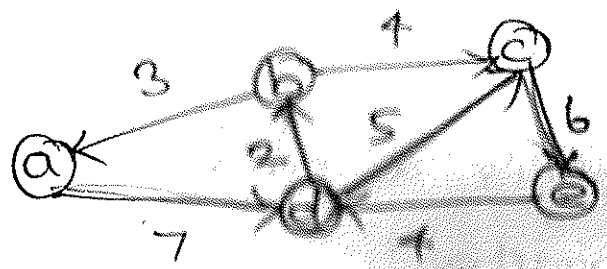
Time complexity

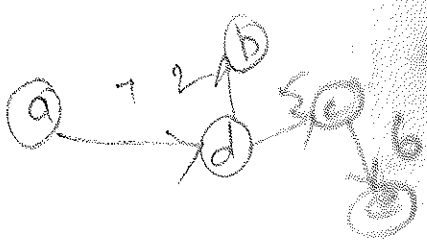
$$T(n) = O(|V|^2)$$

② Linked list and priority queue is used for implementation

Time complexity $T(n) = O(|E| \cdot \log|V|)$

② Solve the following instances of the single source shortest paths problem with vertex a as the source.





Huffman Trees

Text replaced by code word. Text contains n characters. Assigning to each of the text's characters some sequence of bits called codeword. Ex: A B → 0110 111 (character → codeword)

Fixed length encoding

(Text) n = 2 [A encoded to 4 bits, B encoded to 3 bits] codeword

(A bit string) or codeword of the same length. M is assigned to each character of text.

Ex: A B → 110 011

m = 3,

Variable length encoding

Assigning codeword of different lengths to different characters.

Ex: AB → 11 011

Condition

→ No codeword is a prefix of a codword of another character.

Ex: A B → 11 110 ✗ wrong

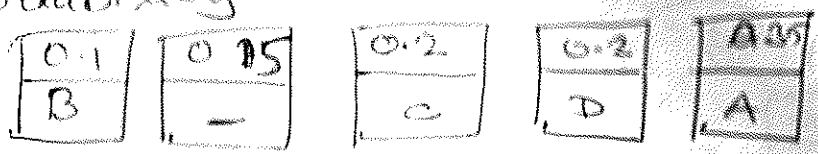
codeword of A, 11 is prefix of codeword of B.

(5) consider the five character alphabet {A, B, C, D, _} with the following occurrence probabilities.

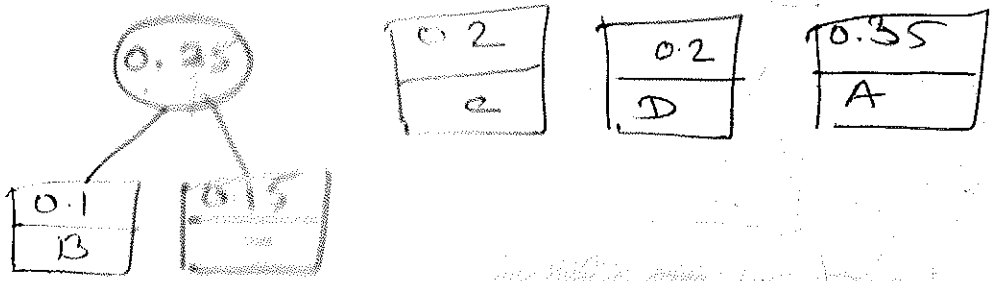
character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15

Huffman tree

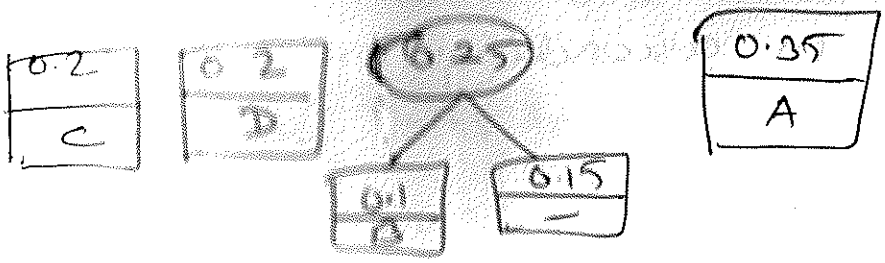
(1) Sort the characters in ascending order of probability



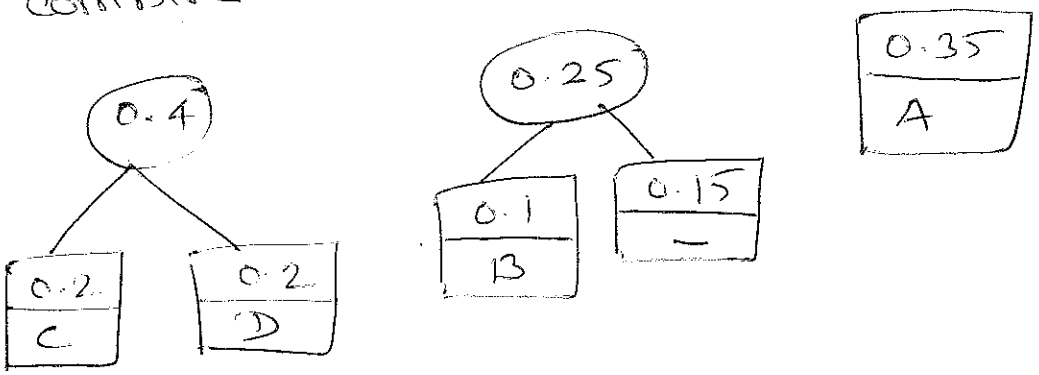
(2) combine first two minimum characters



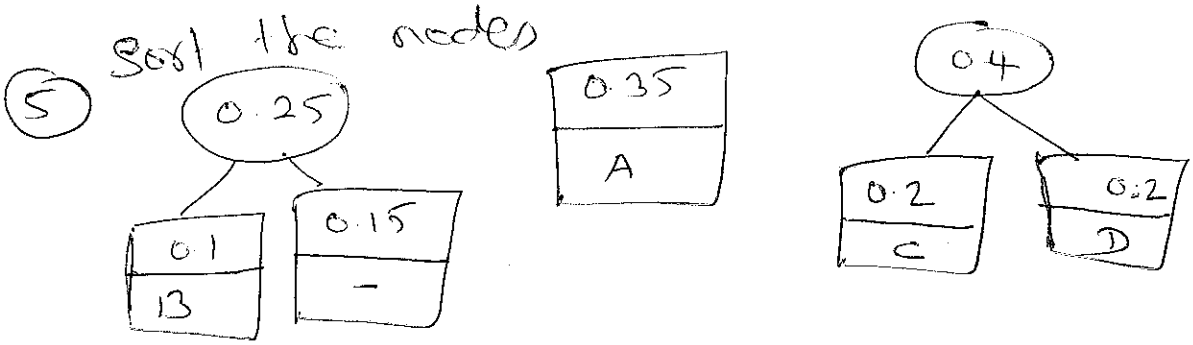
(3) Sort the nodes



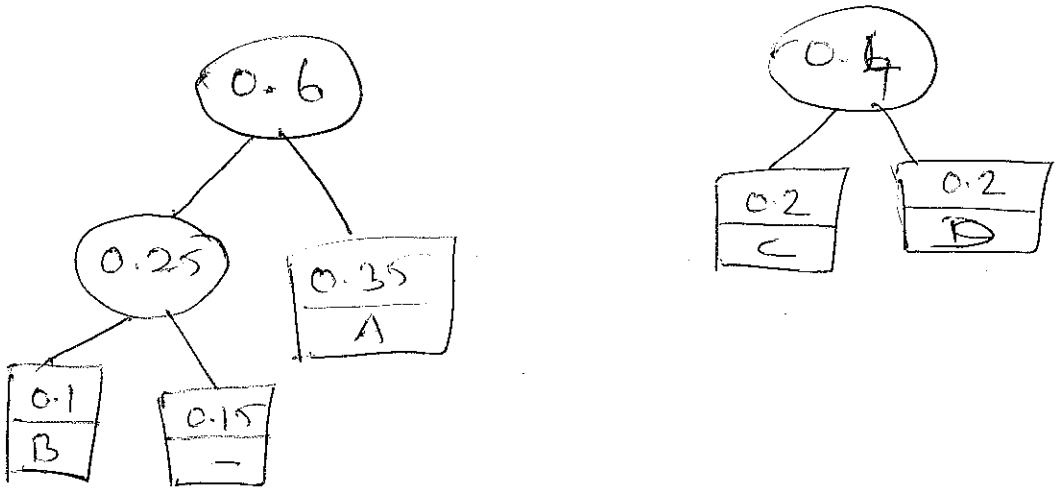
④ combine first 2 minimum



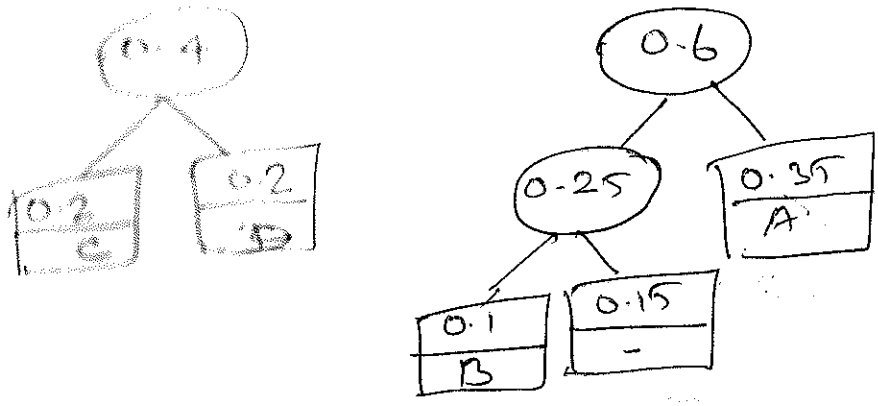
⑤ Sort the nodes



⑥ Combine first 2 minimum

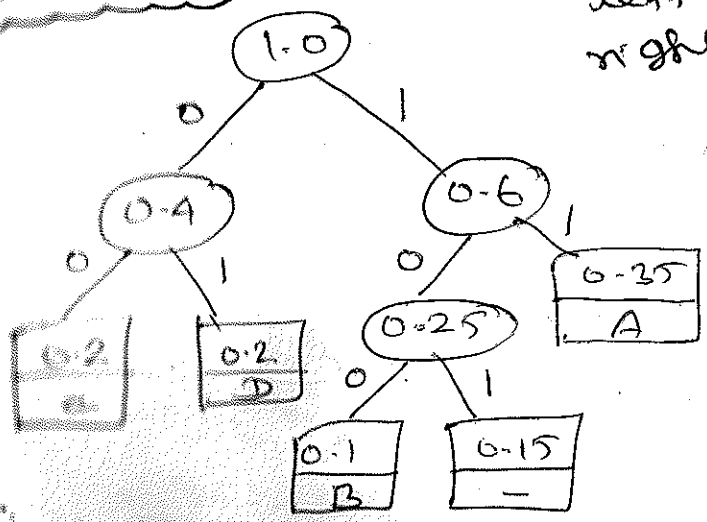


⑦ Sort in ascending order of node value



(8) combine two nodes. Assign left edge = 0 right edge = 1

Huffman Tree



codeword:

character	A	B	C	D	—
codeword	11	100	00	01	101
Probability	0.35	0.1	0.2	0.2	0.15

Maximum number of bits in the codeword is calculated as $= 2 \times 0.35 + 3 \times 0.1 + 2 \times 0.2 + 3 \times 0.2 + 3 \times 0.15$

$= 2.25 =$ (3) bits in codeword.

Ex: DAD is encoded as 01101

100110110101 is decoded as

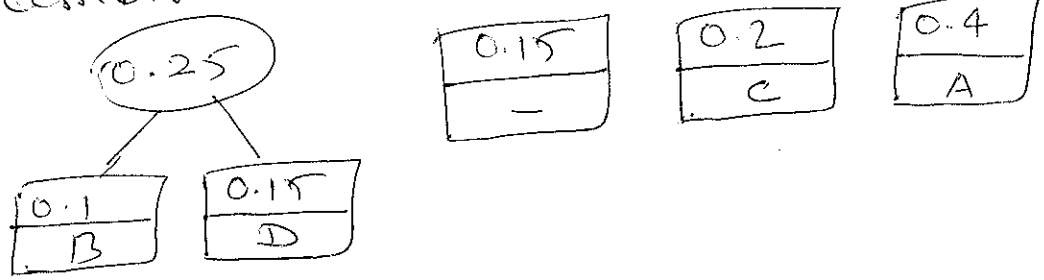
= BAD-AD

(2) construct a huffman code for the following data.

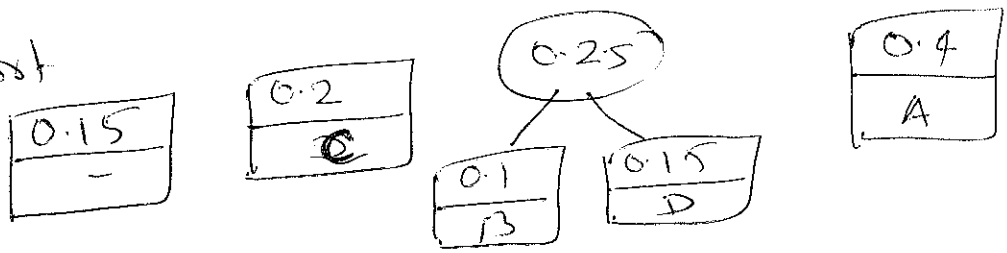
char	A	B	C	D	-
prob	0.4	0.1	0.2	0.15	0.15

(1) sorting

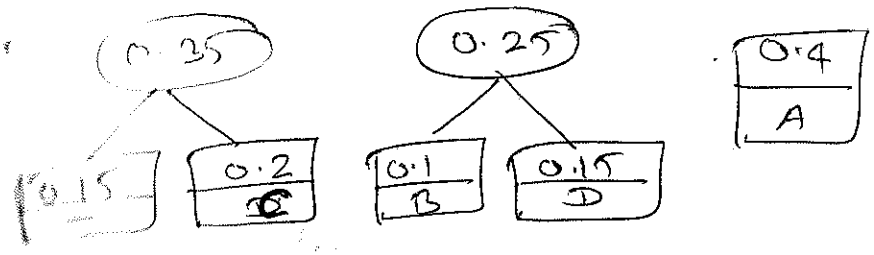
(2) combine



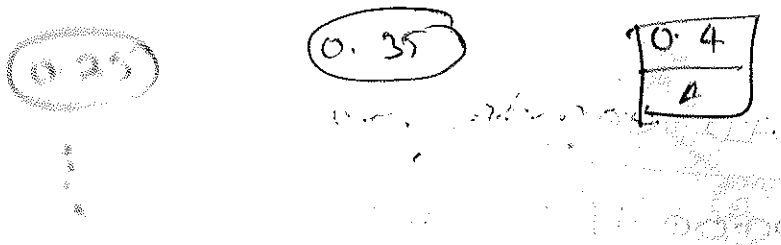
(3) sort



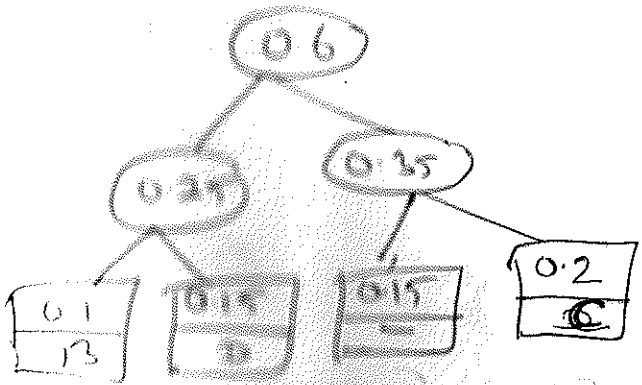
(4) combine



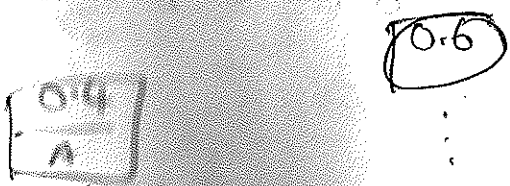
(5) sort



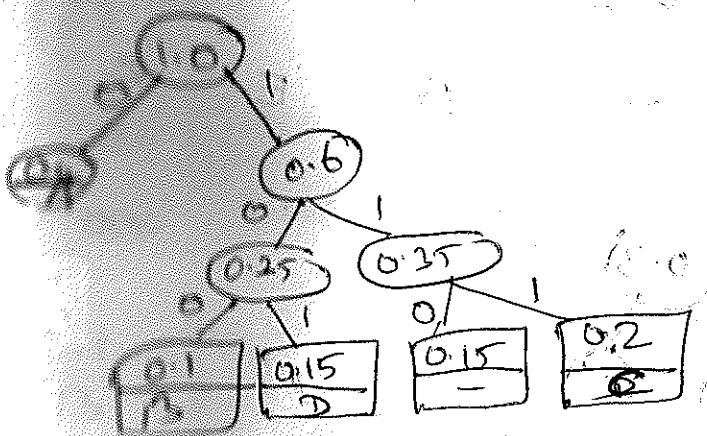
(6) combine



(7) sort



(8) combine



char	A	B	C	D	_
code word	0	100	111	101	110

Encode

a) ABACABAD encoded as

0100 0111 0100 0101
A B A C A B A D

Decode

b) 1000101110010 encoded as

1 000 101 110 010
B A D _ A D A

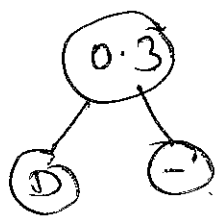
BAD_ADA

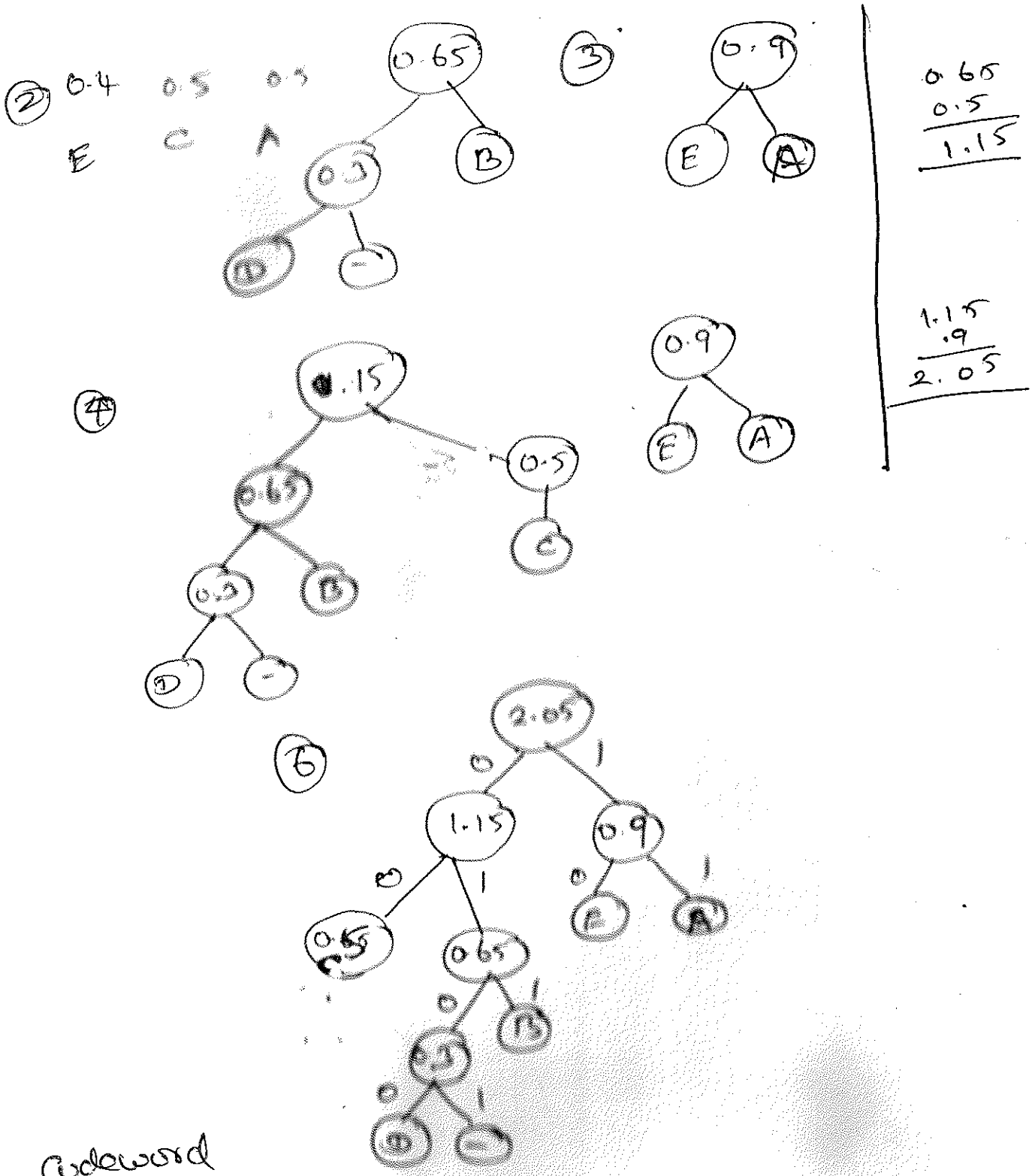
Prob

(3)

	A	B	C	D	E	_	
	0.5	0.35	0.5	0.1	0.4	0.2	=2.05
	0.1	0.2	0.35	0.4	0.5	0.5	
	D	_	B	E	C	~~A~~	

(1)





Codeword

A	B	C	D	E	-
11	011	00	0100	10	0101

o/p → 11 00 11 011 11

A C A B A