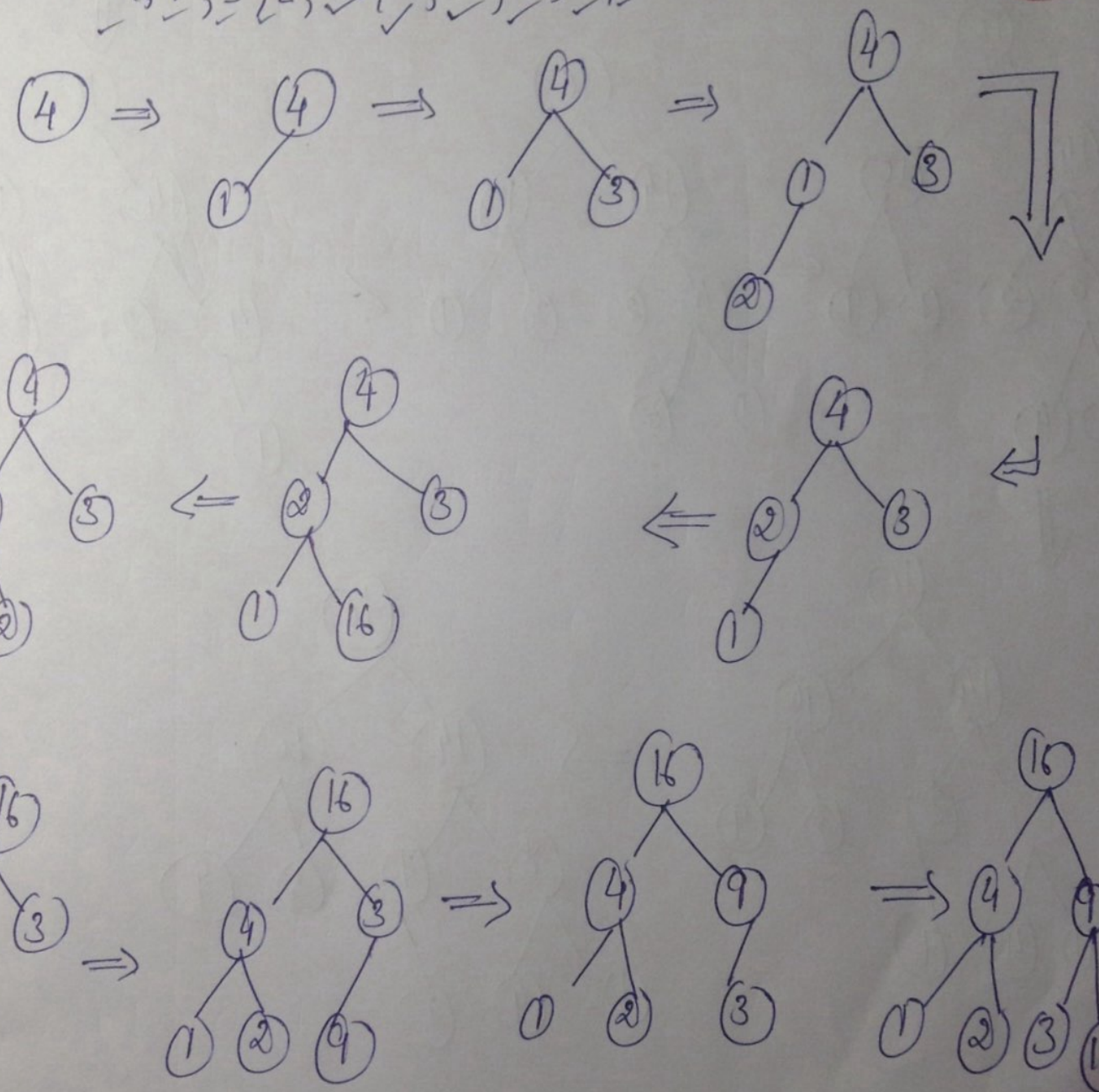
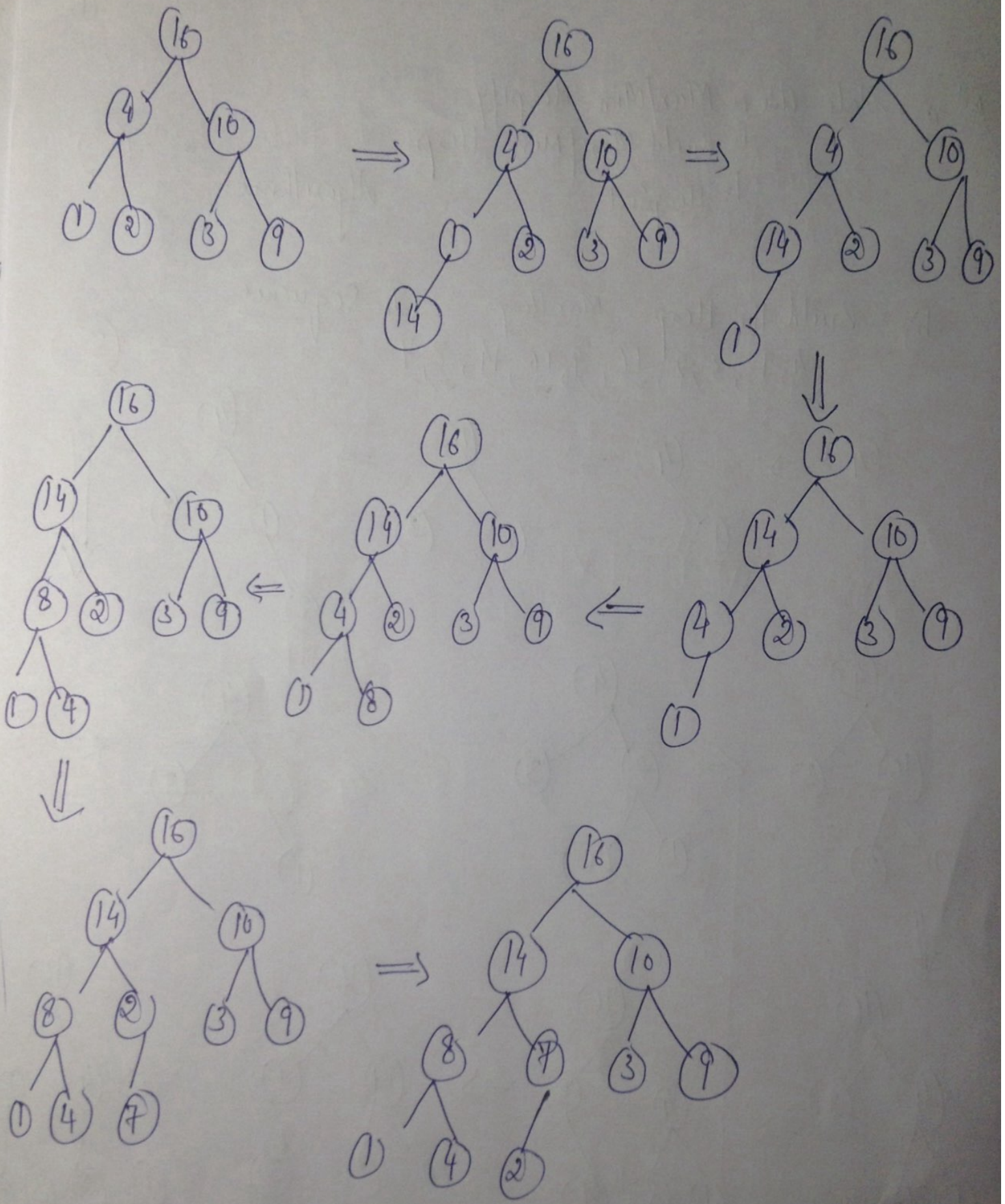
Q.1. a) State the Algorithms. (5)

↳ Max/Min-Heapify
↳ Build Max/Min-Heap
↳ Heapsort

b) Build the Heap. MaxHeap In Sequence (5)

4, 1, 3, 2, 16, 9, 10, 14, 8, 7





Q.3(b). OR.

(i) Kruskal's Technique:

↳ Sorts E in nondecreasing order of edge wts: $w(e_i) \leq \cdots \leq w(e_{i_{|E|}})$

↳ Checks for $E_T \cup \{e_{ik}\}$ is acyclic.

↳ Explain the concept of Disjoint Sets. — (2)

↳ makeset

↳ Union

↳ findset

to achieve the test for acyclic cond^n.

ALGO

// I/P: Wtd. connected graph $G=(V,E)$.

// O/P: E_T: set of edges comprising MST of G.

Sort such that $w(e_i) \leq \ldots \leq w(e_{i_{|E|}})$ — (3)

$E_T \leftarrow \phi$; ecounter $\leftarrow 0$.

$k \leftarrow 0$

while ecounter $< |V|-1$ do

$k \leftarrow k+1$

if $E_T \cup \{e_{ik}\}$ is acyclic

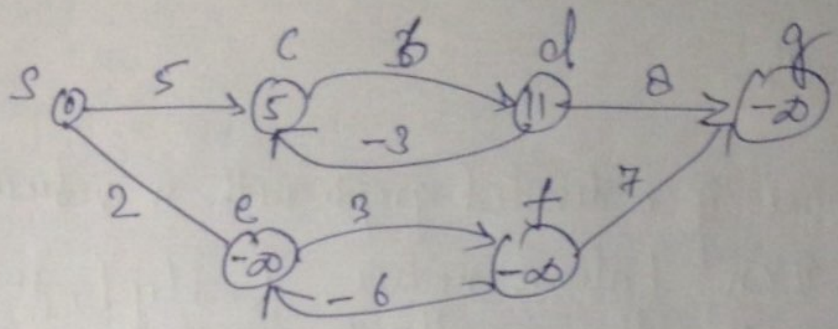
$E_T \leftarrow E_T \cup \{e_{ik}\}$; ecounter ++;

return E_T.

(ii) Negative weight edges

If the $G=(V,E)$ contains no (-ve) wt. cycle reachable from the source s, then $\forall\ v \in V$, the shortest path wt. $\delta(s,v)$ remains well-defined, even if it has a (-ve) wt. value.

eg:



(3)

(1) $\langle s, c\rangle$ => paths $\langle s, c\rangle$, $\langle s, c, d, c\rangle$, $\langle s, c, d, c, d, c\rangle$ and so on } ∞ly many paths.

Cycle $\langle cdc\rangle$ has wt = 6 − 3 = 3.

∴ Shortest path $\langle s, c\rangle = \delta(s, c) = 5$.

(2) s to f: => $\langle s, e, f\rangle$, $\langle s, e, f, e, f\rangle$ } ∞ly many.

so on.

∴ $\delta(s, f) = -\infty$

∵ Cycle $\langle efe\rangle$ = 3 − 6 = −3 Negative.

Bellman-Ford :- Solves the prob. of single source Shortest Path.

Unlike Dijkstra which has (+ve) wt. edges; Bellman-ford considers (−ve) wt. edges as well.

Hence, uses the concept of Negative wt. edges/cycle

(2)

Q4. Warshall's -

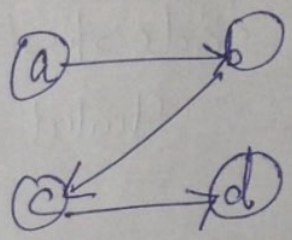
Transitive closure of a directed graph with n vertices can be defined as $n \times n$ boolean matrix $T = \{t_{ij}\}$, in which the elem. in the i^{th} row, j^{th} col $(1 \le i \le n;\ 1 \le j \le n)$ is 1 if there exists a non-trivial directed path from i^{th} to j^{th} vertex; otherwise $t_{ij} = 0$.

(2)

Given Adjacency Matrix

$R^{(0)}$ =

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



(8)

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ OR } (r_{ik}^{(k-1)} \text{ AND } r_{kj}^{(k-1)})$$

$R^{(1)}$ =

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

$R^{(2)}$ =

	1	2	3	4
1	0	1	(1)	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

$R^{(3)}$ =

	1	2	3	4
1	0	1	1	(1)
2	0	0	1	(1)
3	0	0	0	1
4	0	0	0	0

$R^{(4)}$ =

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

Q5.

↳ Discuss the Dynamic Programming Technique explaining what kind of problems it solves and how.

↳ Take an example of Multistage Graph Prob.

↳ Explain the prob.

↳ Solve it using Greedy, DP & brute force

↳ Compare. (10)

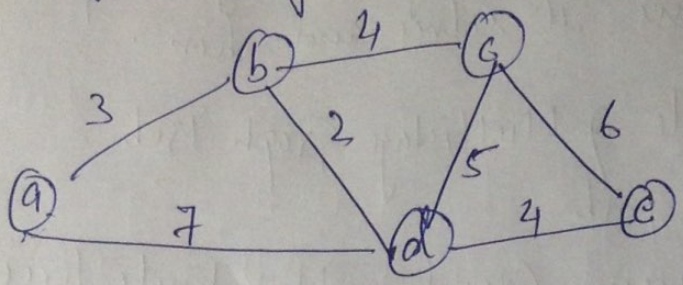
OR.

Prefix Codes: No codeword is also a prefix of some other codeword. Desirably as they simplify coding & are unambiguous. (2)

```
                                  // C: set of chars.
HUFFMAN(C)                        // f[c] :- defined freq.
                                  // Q :- Min-Priority Q keyed on f.
1.  n ← |C|
2.  Q ← C
3.  for i ← 1 to (n-1)
4.      do allocate a new node z
5.          left[z] ← x ← EXTRACT_MIN(Q)
6.          right[z] ← y ← EXTRACT_MIN(Q)
7.          f[z] ← f[x] + f[y]
8.          INSERT(Q, z)
9.  return EXTRACT_MIN(Q).
```

(8)

Q6. Dijkstra's Algo.



↳ INITIALIZE-SINGLE-SOURCE (G, s)
↳ RELAX (u, v, w)
↳ DIJKSTRA (G, w, s)

} Algos. — (5)

Solution: Shortest paths

a – b : 3
a – b – d : 5
a – b – c : 7
a – b – d – e : 9

} Detail the steps. — (5)

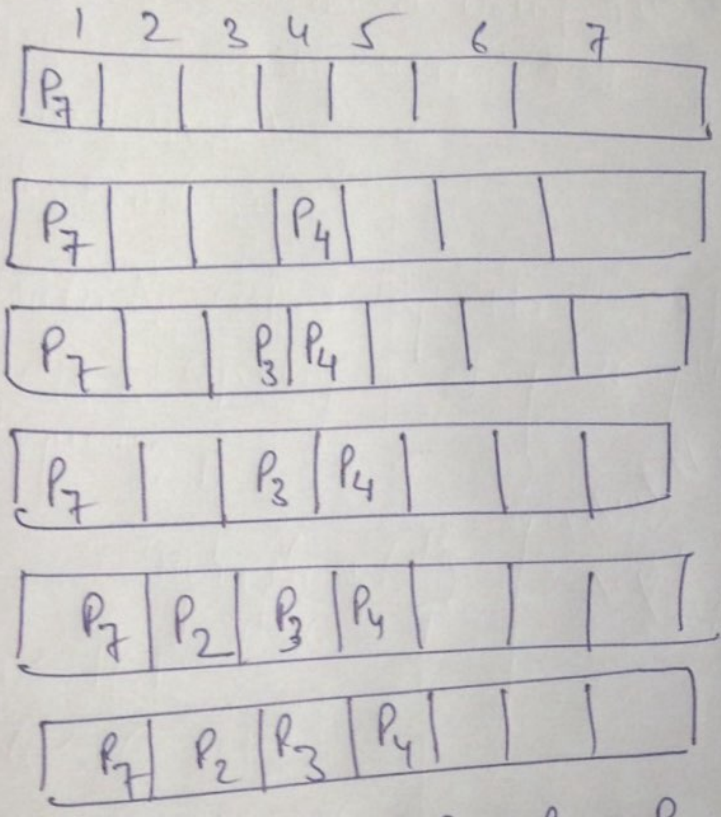
Q.7(a) (6)

n=7. Job Sequencing with deadlines

Profit:	38	20	18	6	5	3	1
Job:	P_7	P_4	P_3	P_5	P_2	P_1	P_6
Deadline:	1	4	3	1	3	1	2

% Array J[].

(5)



Final seq: → $P_7 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$.

Profit = 38 + 5 + 18 + 20 = 81

b). Solve using the Dynamic Programming approach.

(5)

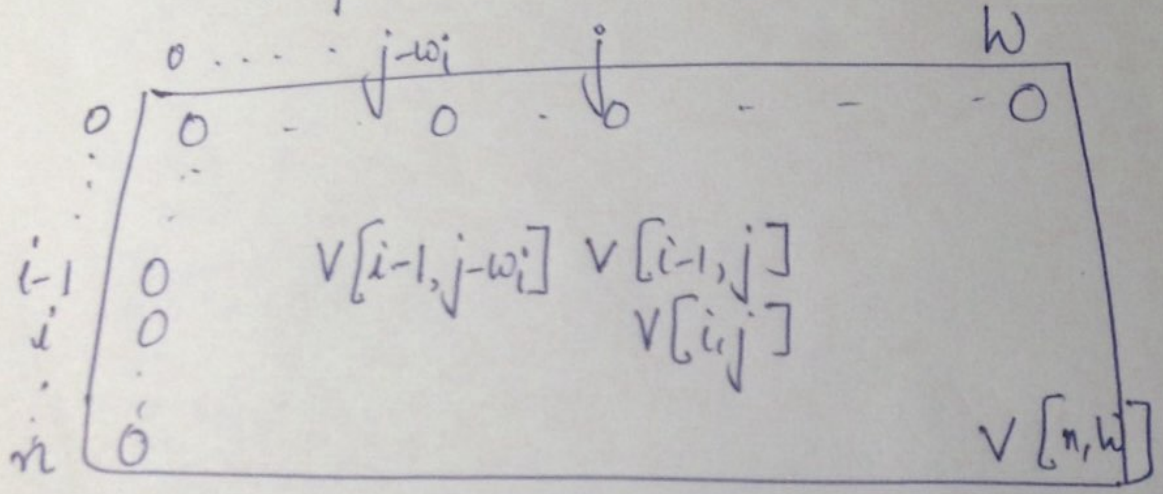
D_0	D_1	D_2
1	2	2
$300	$300	$400

Reliability $\pi = 0.648$

Cost incurred $1050

Q.8(a). 0/1 Knapsack Problem. W = 5 mins.

Question	Time	Points
1	2	2
2	2	5
3	3	8
4	1	1



i	capacity j: 0	1	2	3	4
0	0	0	0	0	0
1 ($w_1=2, v_1=2$)	0	0	2	2	2
2 ($w_2=2, v_2=5$)	0	0	5	7	7
3 ($w_3=3, v_3=8$)	0	0	5	13	15
4 ($w_4=1, v_4=1$)	0	1	5	13	15