Unit III – Dynamic Programming

- Dynamic Programming
  - Computing a Binomial Coefficient
  - Warshall’s algorithm
  - **Floyd’s algorithm**
  - **Optimal Binary Search Trees**
  - Knapsack Problem and Memory functions

Floyd's algorithm

- Weighted connected graph – all pair shortest path
- Algorithm

```
ALGORITHM  Floyd(W[1..n, 1..n])
    //Implements Floyd's algorithm for the all-pairs shortest-paths problem
    //Input: The weight matrix W of a graph with no negative-length cycle
    //Output: The distance matrix of the shortest paths' lengths
    D ← W  //is not necessary if W can be overwritten
    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
```

- Time Complexity – $O(n^3)$

Optimal Binary Search Tree

Cost Matrix

C[i ,i-1] = 0

C[i, i] = 0

C[i, j] = formula

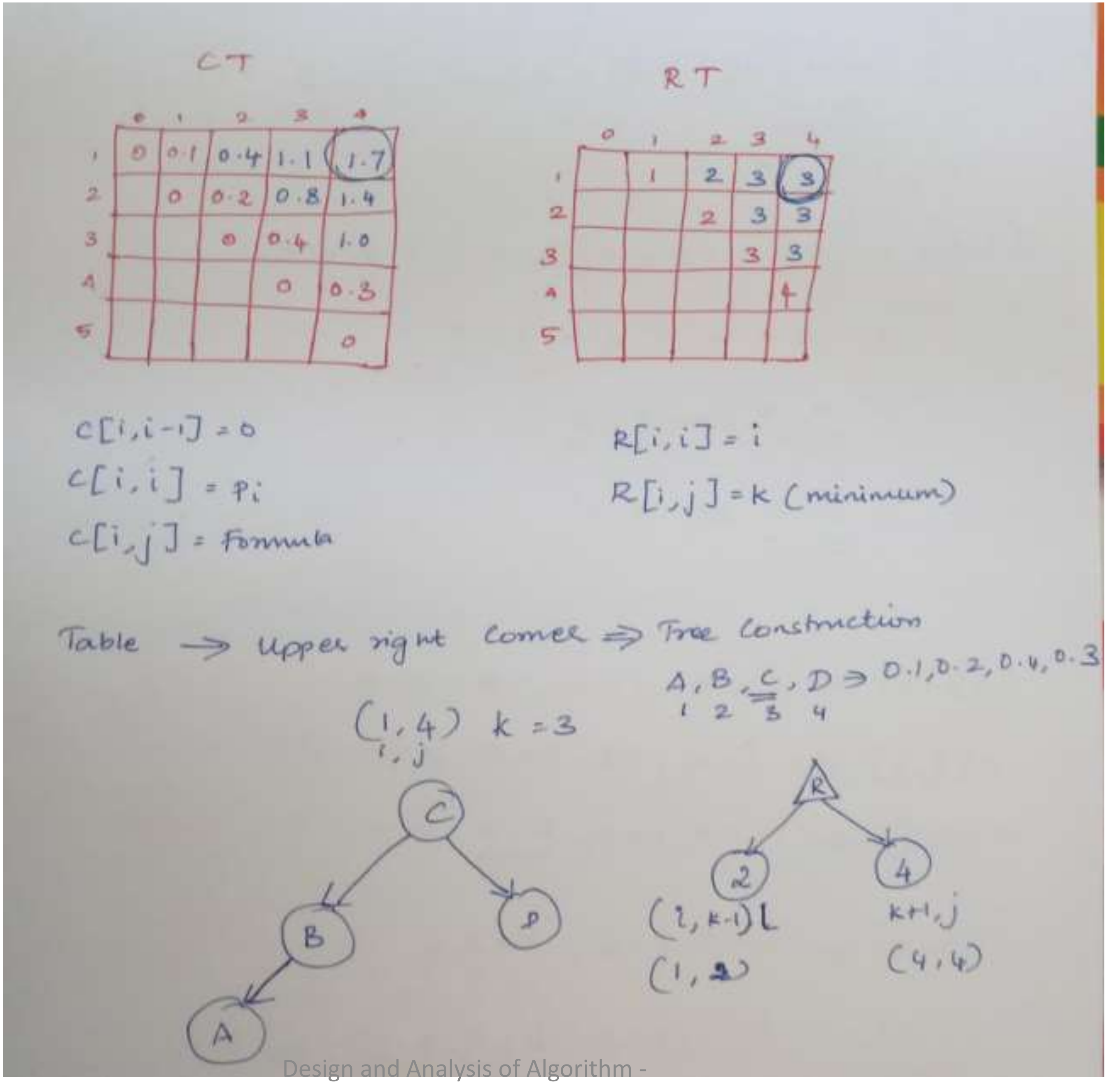
Root Matrix

R[i, i] = i

R[i, j] = k (min)

	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**

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



CT

	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

RT

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

c[i, i-1] = 0
c[i, i] = p_i
c[i, j] = Formula

R[i, i] = i
R[i, j] = k (minimum)

Table → upper right corner ⇒ Tree construction

A, B, C, D ⇒ 0.1, 0.2, 0.4, 0.3
1 2 3 4

(1, 4) k = 3
i, j