5/1)

P[i][j]=

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 4 | 5 | 6 | 8 | 16 | 32 |
| 0 | 0 | 1 | 2 | 4 | 12 | 28 |
| 0 | 0 | 0 | 1 | 3 | 11 | 27 |
| 0 | 0 | 0 | 0 | 2 | 10 | 26 |
| 0 | 0 | 0 | 0 | 0 | 8 | 24 |
| 0 | 0 | 0 | 0 | 0 | 0 | 16 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

W[i][j]=

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 4 | 6 | 9 | 15 | 31 | 63 |
| 0 | 0 | 1 | 3 | 7 | 19 | 47 |
| 0 | 0 | 0 | 1 | 4 | 15 | 42 |
| 0 | 0 | 0 | 0 | 2 | 12 | 38 |
| 0 | 0 | 0 | 0 | 0 | 8 | 32 |
| 0 | 0 | 0 | 0 | 0 | 0 | 16 |

R[i][j]=

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 5 | 5 |
| 0 | 2 | 2 | 3 | 5 | 6 |
| 0 | 0 | 3 | 4 | 5 | 6 |
| 0 | 0 | 0 | 4 | 5 | 6 |
| 0 | 0 | 0 | 0 | 5 | 6 |

13

21

1

8

7

-5

There are total 8 possibilities of the optimal binary search trees. Because there will be three coordinates for which value of r can be selected from any of the two.

3/2) Yes, DP approach can be used to construct the worst O\_BST. Everything in the algorithm will remain same. We just have to make some changes in the probability values given.

3/3) Because in Huffman coding we build bunary search tree comparing frequencies (can be compared with probabilities), but in optimal O\_BST, we use keys as the comparison criteria

3/4)

**Commonalities:-** Both have same search, delete operations and space operation complexities. Both follow binary search tree properties.

**Differences:-** Insert time in splay is O(n). But in O\_BST, insertion complexity is O(n2).

After inserting all the elements in increasing order, splay tree is:-

-5

1

7

8

13

21

After finding elem. 21 16 times splay tree is

-5

1

7

8

13

21

After finding 13 4 times, splay tree is:-

21

-5

1

7

8

13

After finding 8 2 times, splay tree is:-

21

13

-5

1

7

8

After finding 7 1 time, splay tree is :-

21

13

8

-5

1

7

After finding 1 1 time, splay tree is:-

21

13

8

7

-5

1

After finding -5 4 times, splay tree is:-

21

13

8

7

1

-5

5/5)

(b) def swap(str1,str2):

no\_of\_swap=0

str1=list(str2)

print str1

for i in range(len(str2)):

if(str1[i] != str2[i]):

for j in range(i+1,len(str2)):

if str1[j]==str2[i]:

str1[j],str1[i]=str1[i],str1[j]

no\_of\_swap+=1

break

return no\_of\_swap

5/6)

(b) In case of heterogeneous k-d tree, if we have to compare coordinate values then number of leaves to be compared will become twice. Hence total number of leaves to be compared in case of partial match query will be n1/2.