**Practical No.09**

**Title :**

Given sequence k = k1 <k2 < … <kn of n sorted keys, with a search probability pi for each key ki . Build the Binary search tree that has the least search cost given the access probability for each key?

**Objective :**

* To build a binary tree that has the least search cost given the access probability for each key.

**Source Code :**

def optimal\_bst(keys, probabilities):

n = len(keys)

cost = [[0] \* (n + 1) for \_ in range(n + 2)]

for i in range(1, n + 1):

cost[i][i] = probabilities[i - 1]

for l in range(2, n + 1):

for i in range(1, n - l + 2):

j = i + l - 1

cost[i][j] = float('inf')

for k in range(i, j + 1):

temp = cost[i][k - 1] + cost[k + 1][j]

temp += sum(probabilities[i - 1:j])

if temp < cost[i][j]:

cost[i][j] = temp

return cost[1][n]

def main():

keys = input("Enter keys separated by space: ").split()

probabilities = input("Enter probabilities separated by space: ").split()

keys = [int(key) for key in keys]

probabilities = [float(prob) for prob in probabilities]

min\_cost = optimal\_bst(keys, probabilities)

print("Minimum search cost:", min\_cost)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output :**

