Data Analysis and Algorithm

027\_ABHISHEK\_OJHA

**Experiment No – 4 Date of Experiment : 22th September 2021**

**Program : -** Program to implementing hiring problem and analyze its complexity.

# Input :-

Input: Four Jobs with following

deadlines and profits JobID Deadline Profit

a 4 20 b 1 10 c 1 40 d 1 30

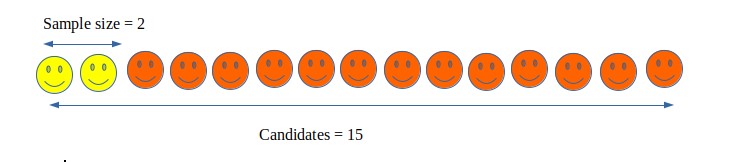
Output: Following is maximum profit sequence of jobs c, a

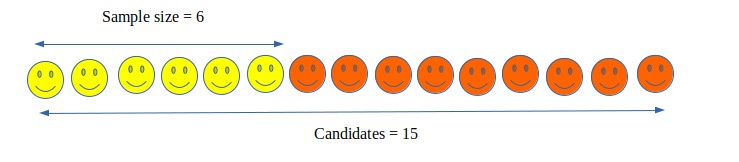
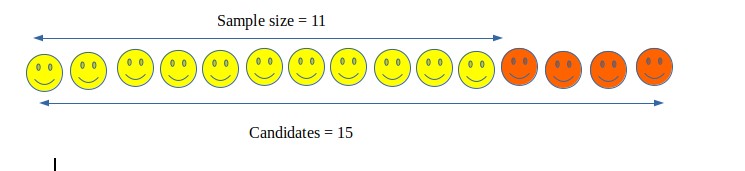
# Algorithm :-

Begin

1. Sort all the jobs based on profit Pi so
2. P1 > P2 > P3 …………………………….>=Pn
3. d = maximum deadline of job in A
4. Create array S[1,…………………,d]
5. For i=1 to n do
6. Find the largest job x
7. For j=i to 1
8. If ((S[j] = 0) and (x deadline<= d))
9. Then
10. S[x] = i;
11. Break;
12. End if
13. End for
14. End for
15. End

# Fig :-

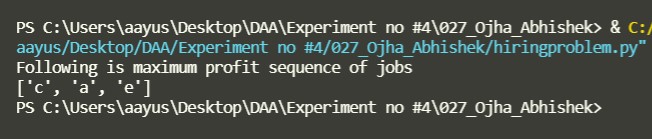




# Practical Implementation hiring Problem :-

|  |
| --- |
| *def* printJobScheduling(*arr*, *t*):    n = len(arr)    for i in range(n): for j in range(n - 1 - i): if arr[j][2] < arr[j + 1][2]: arr[j], arr[j + 1] = arr[j + 1], arr[j]    result = [False] \* t    job = ['-1'] \* t    for i in range(len(arr)): for j in range(min(t - 1, arr[i][1] - 1), -1, -1):  if result[j] is False: result[j] = True job[j] = arr[i][0] break print(job)  arr = [['a', 2, 100],  ['b', 1, 19],  ['c', 2, 27],  ['d', 1, 25],  ['e', 3, 15]]    print("Following is maximum profit sequence of jobs")    printJobScheduling(arr, 3) |

**Output:**



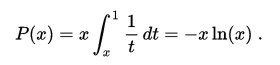
Time Complexity of the above solution is O(n2). It can be optimized using Priority Queue(max heap).

**Time complexity** : O(nlog(n))

**Space complexity** : O(n)

**Analysis :**

The optimal sample size and Probability of success for different values of n are : Optimal Sample size k = n / e Probability of success is given by :



**Conclusion:**

The Optimal Strategy doesn’t always find the best candidate but selects the almost best candidates most of the times