

NEW

PREPARE

CERTIFY

COMPETE

APPLY

Search

h190031920

All Contests > adadynamic2

adadynamic2 [Details](#)

Challenges

Fair Cut

Success Rate: 97.30%

Max Score: 40

Try Again

Grid Walking

Success Rate: 99.19%

Max Score: 55

Try Again

Knapsack

Success Rate: 99.33%

Max Score: 60

Try Again

Current Rank: 1

View your results

Contest ends in a day

Current Leaderboard

Compare Progress

Review Submissions

NEW

PREPARE

CERTIFY

COMPETE

APPLY

Search

h190031920

All Contests > adadynamic2 > Fair Cut

Fair Cut

Problem

Submissions

Leaderboard

Discussions

Problem	Language	Time	Result	Score
Fair Cut	Python 3	7 days ago	Accepted	40

View Results

Contest Calendar

Interview Prep

Blog

Scoring

Environment

FAQ

About Us

Support

Careers

Terms Of Service

Privacy Policy

Request a Feature

Language: Python 3

Open in editor

```

1 n, k = map(int, input().split())
2 if k > n // 2:
3     k = n - k
4
5 a = sorted(map(int, input().split()))
6
7 dp = [[float('inf') for i in range(n + 1)] for j in range(n + 1)]
8
9 dp[0][0] = 0
10
11
12 for i in range(0, n):
13     for j in range(0, i + 1):
14         if j > k or i - j > n - k:
15             continue
16
17         temp_li = dp[i][j] + a[i] * (i - j - (n - k - (i - j)))
18
19         temp_lu = dp[i][j] + a[i] * (j - (k - j))
20
21         if dp[i + 1][j + 1] > temp_li:
22             dp[i + 1][j + 1] = temp_li
23
24         if dp[i + 1][j] > temp_lu:
25             dp[i + 1][j] = temp_lu
26
27 print(dp[n][k])
28
29

```

Grid Walking

Problem

Submissions

Leaderboard

Discussions

Submitted 7 days ago • Score: 55.00

Status: Accepted

✓	Test Case #0	✓	Test Case #1	✓	Test Case #2
✓	Test Case #3	✓	Test Case #4	✓	Test Case #5
✓	Test Case #6	✓	Test Case #7	✓	Test Case #8
✓	Test Case #9	✓	Test Case #10	✓	Test Case #11
✓	Test Case #12				

Submitted Code

Language: Python 3

Open in editor

```
4 import re
5 import sys
6
7 def gridWalking(m, x, D):
8     MOD = 10 ** 9 + 7
9     n = len(D)
10     md = [[0 for _ in range(n + 1)] for _ in range(m + 1)]
11     for i in range(n):
12         M = [[0 for _ in range(m + 1)] for _ in range(D[i] + 1)]
13         for j in range(1, D[i] + 1):
14             M[j][0] = 1
15             for j in range(1, m + 1):
16                 for k in range(1, D[i] + 1):
17                     M[k][j] = 0
18                     if k - 1 > 0:
19                         M[k][j] = (M[k][j] + M[k - 1][j - 1]) % MOD
20                     if k + 1 <= D[i]:
21                         M[k][j] = (M[k][j] + M[k + 1][j - 1]) % MOD
22             md[0][i + 1] = 1
23             for j in range(1, m + 1):
24                 md[j][i + 1] = M[x[i]][j]
25     c = [[0 for _ in range(m + 1)] for _ in range(m + 1)]
26     for i in range(m + 1):
27         c[i][0] = 1
28         c[i][i] = 1
29     for i in range(1, m + 1):
30         for j in range(1, i):
31             c[i][j] = (c[i - 1][j - 1] + c[i - 1][j]) % MOD
32     mdt = [[0 for _ in range(n + 1)] for _ in range(m + 1)]
33     for i in range(m + 1):
34         mdt[i][1] = md[i][1]
35     for i in range(n + 1):
36         mdt[0][i] = 1
37     for i in range(2, n + 1):
38         for j in range(1, m + 1):
39             mdt[j][i] = 0
40             for k in range(j + 1):
41                 mdt[j][i] = (
42                     mdt[j][i] + ((c[j][j - k] * ((mdt[k][i - 1] * md[j - k][i]) % MOD)) % MOD)) % MOD
43     return mdt[m][n]
44
45 if __name__ == '__main__':
46     fptr = open(os.environ['OUTPUT_PATH'], 'w')
47
48     t = int(input().strip())
49
50     for t_itr in range(t):
51         first_multiple_input = input().rstrip().split()
52
53         n = int(first_multiple_input[0])
54
55         m = int(first_multiple_input[1])
56
57         x = list(map(int, input().rstrip().split()))
58
59         D = list(map(int, input().rstrip().split()))
60
61         result = gridWalking(m, x, D)
62
63         fptr.write(str(result) + '\n')
64
65     fptr.close()
```

Knapsack

Problem

Submissions

Leaderboard

Discussions

Submitted 3 minutes ago • Score: 60.00

Status: Accepted

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	✓ Test Case #11

Submitted Code

Language: Python 3

[Open in editor](#)

```
5 import random
6 import re
7 import sys
8
9 #
10 # Complete the 'unboundedKnapsack' function below.
11 #
12 # The function is expected to return an INTEGER.
13 # The function accepts following parameters:
14 # 1. INTEGER k
15 # 2. INTEGER_ARRAY arr
16 #
17
18 def maxSum(arr,k):
19     subset = [[False for i in range(k+1)] for j in range(len(arr)+1)]
20     for i in range(len(arr)+1):
21         subset[i][0] = True
22         for i in range(1,len(arr)+1):
23             for j in range(1,k+1):
24                 if(arr[i-1]>j):
25                     subset[i][j] = subset[i-1][j]
26                 else:
27                     subset[i][j] = subset[i-1][j] or subset[i][j-arr[i-1]]
28
29     j = k
30     while j>=0:
31         if subset[len(arr)][j] == True:
32             return j
33         j-=1
34
35 if __name__ == '__main__':
36     fptr = open(os.environ['OUTPUT_PATH'], 'w')
37
38     t = int(input().strip())
39
40     for i in range(t):
41         first_multiple_input = input().rstrip().split()
42
43         n = int(first_multiple_input[0])
44         k = int(first_multiple_input[1])
45
46         arr = list(map(int, input().rstrip().split()))
47
48         result = maxSum(arr, k)
49
50         fptr.write(str(result) + '\n')
51
52     fptr.close()
```