copyright: Jagadeesh Vasudevamurthy

Frie: Friends.ipynb

WRITE CODE ONLY IN THIS CELL

```
############################################################
# Solution.py
# Author: Jagadeesh Vasudevamurthy
# Copyright: Jagadeesh Vasudevamurthy 2023
###########################################################

############################################################
# All imports
###########################################################
##from Util import *

class Solution:
  def __init__(self,n:'int',a:'list of size 2'):
    self._n = n
    #a[0] = How many friends are there in n. Use 
_increment_number_of_friends to fill
    #a[1] = num_steps Use _increment_steps
    self._list = a
    ##YOU CAN HAVE ANY NUMBER OF DATA STRUCTURES HERE
    #self._u = Util()


    ## NOTHING CAN BE CHANGED BELOW
    self._alg()

  def _increment_steps(self)->'none':
    self._list[1] = self._list[1] + 1

  def _increment_number_of_friends(self)->"int":
    self._list[0] = self._list[0] + 1
    return (self._list[0])


  ##Implement your code BELOW
  ##You can have any number of private variables and functions

  def _alg(self):
    print("WRITE CODE")
    print("Write as many small functions as possible")
    #create an empty list of size n that stores the sum of factors
    factors_sum = [0] * (self._n + 1)

    #first for loop - to get the sum of factors
    for i in range(1, self._n // 2): #iterate for half size of n to
save time
        #call this function for each iteration to record number of
iterations
        self._increment_steps()
        for j in range(i * 2, self._n + 1, i):
            factors_sum[j] += i

    # second for loop - to find if there is a friend number
    for i in range(1, self._n + 1):
        #get the ith index of first sum of factors and assign to 2nd
sum of factors
        sum_of_factors = factors_sum[i]
        #only proceed if its greater than i
        if sum_of_factors > i:
            # now, check if both sum of the factors are equal
            if sum_of_factors <= self._n and
factors_sum[sum_of_factors] == i:
                #if true, call this function to record the number of
friends
                self._increment_number_of_friends()
                print(self._list[0],":",i,sum_of_factors)
```

Some useful function

Can use if required

```
############################################################
# Util.py
# Author: Jagadeesh Vasudevamurthy
# Copyright: Jagadeesh Vasudevamurthy 2020
###########################################################

############################################################
# NOTHING CAN BE CHANGED IN THIS FILE
###########################################################

############################################################
# All imports
###########################################################
import sys # For getting Python Version
import random
import math
from time import process_time
```

```
class Util():
  pass


  ############################################
  # log to the next possible integer
  #########################################
  def log_upper_bound(self, n:'int', b:'int')->'int':
    f = math.log(n,b)
    c = math.ceil(f)
    return c

  ############################################
  # log to the smallest possible integer
  #########################################
  def log_lower_bound(self, n:'int', b:'int')->'int':
    f = math.log(n,b)
    c = math.floor(f)
    return c

  ############################################
  # sqrt to the next possible integer
  #########################################
  def sqrt_upper_bound(self, n:'int')->'int':
    f = math.sqrt(n)
    c = math.ceil(f)
    return c
```

TEST BENCH

NOTHING CAN BE CHANGED BELOW

```
############################################################
# Friends.py
# Author: Jagadeesh Vasudevamurthy
# Copyright: Jagadeesh Vasudevamurthy 2023
###########################################################

############################################################
#           NOTHING CAN BE CHANGED IN THIS FILE
###########################################################

############################################################
# All imports
###########################################################
#from Util import *
```

```
#from Solution import *


class Friends():
  def __init__(self):
    self._u = Util()
    self._testBench()

  def _testBench(self):
    self._tests()
    print("ALL TESTS PASSED")

  def _test1(self,n:'int',ans:'int'):
    print("-------------", n , "------------------------")
    a = [0,0]
    t1_start = process_time()
    s = Solution(n,a) ##All action happens here
    t1_stop = process_time()
    if (a[1] == 0):
        print("How did you solve the problem in 0 steps")
        print("when ever you loop call _increment_steps() ")
        assert(False)
    if (a[0] == 0):
        print("Number of friends is 0. How?")
        print("when ever you find friend call
_increment_number_of_friends ")
        assert(False)
    d = t1_stop - t1_start;
    print(n, " has ", a[0], " friends. Took", a[1], " steps to
compute")
    print("Total CPU time in sec =",d)
    logn_base2 = self._u.log_lower_bound(n,2)
    nlogn = n * logn_base2
    w = a[1] / nlogn
    print("n = ", n)
    print("logn ", logn_base2)
    print("nlogn ",nlogn)
    print("num_steps/nlogn", w)
    if (a[0] != ans):
      print(n," Has", ans, "Friends. But you are telling",a[0])
      assert(a[0] == ans)


  def _tests(self):
    n = 10000
    a = self._test1(n,5)

    n = 20000
    a = self._test1(n,8)
    n = 100000000 ## YOU CANNOT CHANGE THIS.
    a = self._test1(n,231)


############################################################
# main
# YOU CANNOT CHANGE ANYTHING BELOW
###########################################################
def main():
  print("Testing Friends.py Starts")
  s = Friends()
  print("Testing Friends.py ENDS")

############################################################
# Pthin calls this
###########################################################
if (__name__  == '__main__'):
  main()
```

```
Testing Friends.py Starts
------------- 10000 -------------------------
WRITE CODE
Write as many small functions as possible
1 : 220 284
2 : 1184 1210
3 : 2620 2924
4 : 5020 5564
5 : 6232 6368
10000  has  5  friends. Took 4999  steps to compute
Total CPU time in sec = 0.00737799999998856
n =  10000
logn  13
nlogn  130000
num_steps/nlogn 0.038453846153846l5
------------- 20000 -------------------------
WRITE CODE
Write as many small functions as possible
1 : 220 284
2 : 1184 1210
3 : 2620 2924
4 : 5020 5564
5 : 6232 6368
6 : 10744 10856
7 : 12285 14595
8 : 17296 18416
20000  has  8  friends. Took 9999  steps to compute
Total CPU time in sec = 0.014919999999989386
n =  20000
logn  14
```

```
nlogn  280000
num_steps/nlogn 0.03571071428571428
------------- 100000000 -------------------------
WRITE CODE
Write as many small functions as possible
1 : 220 284
2 : 1184 1210
3 : 2620 2924
4 : 5020 5564
5 : 6232 6368
6 : 10744 10856
7 : 12285 14595
8 : 17296 18416
9 : 63020 76084
10 : 66928 66992
11 : 67095 71145
12 : 69615 87633
13 : 79750 88730
14 : 100485 124155
15 : 122265 139815
16 : 122368 123152
17 : 141664 153176
18 : 142310 168730
19 : 171856 176336
20 : 176272 180848
21 : 185368 203432
22 : 196724 202444
23 : 280540 365084
24 : 308620 389924
25 : 319550 430402
26 : 356408 399592
27 : 437456 455344
28 : 469028 486178
29 : 503056 514736
30 : 522405 525915
31 : 600392 669688
32 : 609928 686072
33 : 624184 691256
34 : 635624 712216
35 : 643336 652664
36 : 667964 783556
37 : 726104 796696
38 : 802725 863835
39 : 879712 901424
40 : 898216 980984
41 : 947835 1125765
42 : 998104 1043096
43 : 1077890 1099390
44 : 1154450 1189150
45 : 1156870 1292570
```

```
46 : 1175265 1438983
47 : 1185376 1286744
48 : 1280565 1340235
49 : 1328470 1483850
50 : 1358595 1486845
51 : 1392368 1464592
52 : 1466150 1747930
53 : 1468324 1749212
54 : 1511930 1598470
55 : 1669910 2062570
56 : 1798875 1870245
57 : 2082464 2090656
58 : 2236570 2429030
59 : 2652728 2941672
60 : 2723792 2874064
61 : 2728726 3077354
62 : 2739704 2928136
63 : 2802416 2947216
64 : 2803580 3716164
65 : 3276856 3721544
66 : 3606850 3892670
67 : 3786904 4300136
68 : 3805264 4006736
69 : 4238984 4314616
70 : 4246130 4488910
71 : 4259750 4445050
72 : 4482765 5120595
73 : 4532710 6135962
74 : 4604776 5162744
75 : 5123090 5504110
76 : 5147032 5843048
77 : 5232010 5799542
78 : 5357625 5684679
79 : 5385310 5812130
80 : 5459176 5495264
81 : 5726072 6369928
82 : 5730615 6088905
83 : 5864660 7489324
84 : 6329416 6371384
85 : 6377175 6680025
86 : 6955216 7418864
87 : 6993610 7158710
88 : 7275532 7471508
89 : 7288930 8221598
90 : 7489112 7674088
91 : 7577350 8493050
92 : 7677248 7684672
93 : 7800544 7916696
94 : 7850512 8052488
95 : 8262136 8369864
```

```
96 : 8619765 9627915
97 : 8666860 10638356
98 : 8754130 10893230
99 : 8826070 10043690
100 : 9071685 9498555
101 : 9199496 9592504
102 : 9206925 10791795
103 : 9339704 9892936
104 : 9363584 9437056
105 : 9478910 11049730
106 : 9491625 10950615
107 : 9660950 10025290
108 : 9773505 11791935
109 : 10254970 10273670
110 : 10533296 10949704
111 : 10572550 10854650
112 : 10596368 11199112
113 : 10634085 14084763
114 : 10992735 12070305
115 : 11173460 13212076
116 : 11252648 12101272
117 : 11498355 12024045
118 : 11545616 12247504
119 : 11693290 12361622
120 : 11905504 13337336
121 : 12397552 13136528
122 : 12707704 14236136
123 : 13671735 15877065
124 : 13813150 14310050
125 : 13921528 13985672
126 : 14311688 14718712
127 : 14426230 18087818
128 : 14443730 15882670
129 : 14654150 16817050
130 : 15002464 15334304
131 : 15363832 16517768
132 : 15938055 17308665
133 : 16137628 16150628
134 : 16871582 19325698
135 : 17041010 19150222
136 : 17257695 17578785
137 : 17754165 19985355
138 : 17844255 19895265
139 : 17908064 18017056
140 : 18056312 18166888
141 : 18194715 22240485
142 : 18655744 19154336
143 : 20014808 21457192
144 : 20022328 22823432
145 : 20308995 20955645
```

```
146 : 21448630 23030090
147 : 22227075 24644925
148 : 22249552 25325528
149 : 22508145 23111055
150 : 22608632 25775368
151 : 23358248 25233112
152 : 23389695 25132545
153 : 23628940 27428276
154 : 24472180 30395276
155 : 25596544 25640096
156 : 25966832 26529808
157 : 26090325 26138475
158 : 28118032 28128368
159 : 28608424 29603576
160 : 30724694 32174506
161 : 30830696 31652704
162 : 31536855 32148585
163 : 31818952 34860248
164 : 32205616 34352624
165 : 32642324 35095276
166 : 32685250 34538270
167 : 33501825 36136575
168 : 34256222 35997346
169 : 34364912 34380688
170 : 34765731 36939357
171 : 35115795 43266285
172 : 35361326 40117714
173 : 35373195 40105845
174 : 35390008 39259592
175 : 35472592 36415664
176 : 37363095 45663849
177 : 37784810 39944086
178 : 37848915 39202605
179 : 38400512 38938288
180 : 38637016 40678184
181 : 38663950 43362050
182 : 38783992 41654408
183 : 38807968 40912232
184 : 43096904 46715896
185 : 44139856 44916944
186 : 45263384 46137016
187 : 46237730 61319902
188 : 46271745 49125375
189 : 46521405 53011395
190 : 46555250 55880590
191 : 46991890 48471470
192 : 48639032 52967368
193 : 48641584 48852176
194 : 49215166 55349570
195 : 50997596 51737764
```

```
196 : 52695376 56208368
197 : 56055872 56598208
198 : 56512610 75866014
199 : 56924192 64562488
200 : 58580540 70507972
201 : 59497888 61953512
202 : 63560025 65003175
203 : 63717615 66011985
204 : 66595130 74824390
205 : 66854710 71946890
206 : 67729064 69439576
207 : 67738268 79732132
208 : 68891992 78437288
209 : 71015260 85458596
210 : 71241830 78057370
211 : 72958556 74733604
212 : 73032872 78469528
213 : 74055952 78166448
214 : 74386305 87354495
215 : 74769345 82824255
216 : 75171808 77237792
217 : 75226888 81265112
218 : 78088504 88110536
219 : 78447010 80960990
220 : 79324875 87133365
221 : 80422335 82977345
222 : 83135650 85603550
223 : 84591405 89590995
224 : 86158220 99188788
225 : 89477984 92143456
226 : 90437150 94372450
227 : 91996816 93259184
228 : 93837808 99899792
229 : 95629904 97580944
230 : 96304845 96747315
231 : 97041735 97945785
100000000  has  231  friends. Took 49999999  steps to compute
Total CPU time in sec = 390.525084
n =  100000000
logn  26
nlogn  2600000000
num_steps/nlogn 0.019230768846153847
ALL TESTS PASSED
Testing Friends.py ENDS
```