

## Waga szalkowa

Problem: *elkhok*: def waga(l, n, s=0, p=0)

Dany jest zestaw odważników. Jakie ciężary można odważyć z użyciem tych odważników?

Przykład:

odw = [1, 3, 5, 10, 16, 24] len = 6

def waga(l, n, p):

if n==0: return True

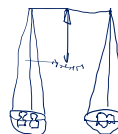
if p==len(l): return False

return waga(l, n-l[p], p+1) or waga(l, n, p+1)

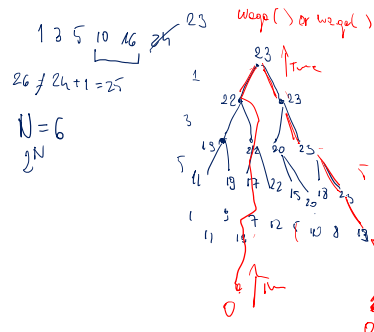
# end def

for w in range(1, 50):

print(w, waga(odw, w, 0))



0 00000  
1 00001  
2 00010  
3 00011



63 11111

1

2

1, 3, 5, 10, 16, 24

def waga(l, n, p=0, res=[]):

if n==0:

print(res)  
return True

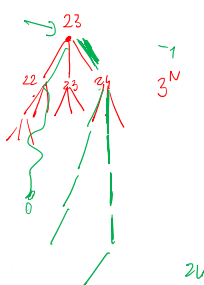
if p==len(l):

return False

waga(l, n-l[p], p+1, res+[l[p]]) or

waga(l, n, p+1, res)

waga(l, n+l[p], p+1, res+[l[p]])



3

## Przykład - pary

Problem:

Dana jest tablica/lista z liczbami naturalnymi. Należy policzyć ile jest par elementów o określonym iloczynie.

Przykład:

t = [4, 1, 5, 7, 9, 4, 5, 9, 6, 5, 3, 2, 7, 6, 1, 1, 7, 9, 9, 1]

Są 4 pary o iloczynie 24.

4

## Przykład – Licz pary

import random

def generuj(n):

return [ random.randint(1,9) for \_ in range(n) ]

# end def

def licz\_pary(l, s):

licz = 0

for i in range(len(l)-1):

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

if l[i]\*l[j]==s:

licz = licz+1

print("pary", licz)

# end def

t = generuj(20)

licz\_pary(t, 24)



5

## Licz pary w tablicy 2-wymiarowej

N = 100

T = [[randint(1,9) for \_ in range(N)] for \_ in range(N)]

for i in range(N):

for j in range(N):

for k in range(N):

for l in range(N):

if i!=k and j!=l:

if T[i][j]+T[k][l]==s:

licz += 1

licz\_pary = 0

for i in range(N-1):



T[i][j]

for j in range(N-1):

for l in range(N-1):

T[i][j]+T[k][l]==s

6

## Przykład – Licz trójki

```
def licz_trojki(l,s):
    licz = 0
    for i in range(len(l)):
        for j in range(i+1, len(l)):
            for k in range(j+1, len(l)):
                if l[i]*l[j]*l[k]==s:
                    licz = licz+1
    print("trojki",licz)
# end def

t = generuj(20)
licz_trojki(t,24)
```

*Handwritten notes:*   
 -  $i \neq j \neq k$  (with arrows pointing to i, j, k in the loops)  
 -  $i < j < k$  (with arrows pointing to i, j, k in the loops)  
 -  $i, j, k$  (with arrows pointing to i, j, k in the loops)

7

## Przykład – Licz n-ki

```
def licz_nki(l,s,n,p):
    global licznik
    if n==1:
        for i in range(p, len(l)):
            if l[i]==s: licznik=licznik+1
    else:
        for i in range(p, len(l)):
            if s%l[i]==0: licz_nki(l,s//l[i],n-1,i+1)
# end def

t = generuj(20)

licznik=0
licz_nki(t,24,4,0)
print("nki",licznik)
```

*Handwritten notes:*   
 -  $n=1$  (with arrow pointing to  $n==1$ )  
 -  $n$  (with arrow pointing to  $n$  in the function signature)  
 -  $n-1$  (with arrow pointing to  $n-1$  in the recursive call)  
 -  $i+1$  (with arrow pointing to  $i+1$  in the recursive call)  
 -  $s \div l[i]$  (with arrow pointing to  $s//l[i]$  in the recursive call)  
 -  $n=4 \rightarrow n=3$  (with arrows pointing to  $n=4$  and  $n=3$ )  
 -  $s=24$  (with arrow pointing to  $s$  in the function signature)  
 -  $24/2 = 12$  (with arrow pointing to  $s//l[i]$  in the recursive call)

8