```
# ssp.py - Polynomial time algorithm for Subset sum problem.
#
# Theory of the implemented algorithm can be found at:
#                                    http://es-andreabianchini.it/ssp_eng.pdf
#
# Copyright 2020 Andrea Bianchini
#
#  This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
#

from random import randrange
import math


def computeBase(c12):
    i=0;
    while(c12[i]==0 and i<n-1):
        i=i+1
    # i is the first one
    i=i+1
    # skipped
    while(i<n-1 and c12[i]==0):
      i=i+1
    # i is the second one
    while(i<n-1 and c12[i]==1):
      i=i+1

    if (i>n-1):
      return n-1

    return i

def findMax(topvalue, currvalue, value, start):
    lsb1=0
    lsb2=0
    lsb22=0
    lsbmax=0
    lsbmin=0
    i=0
    basemax=0
    base=0
    m=0
    firstbit=0
    k=0
    amax=0
    kk=0
    oldamax=0
    solfound=0
    global ro
    global teta
    global alfa
    global niter
    global niter1
```

```
global w
global sol
global c1
global bestsolk

if (ro<0.0):
        ro=value*2.0/(w[start]+w[0])
#       printf("ro=%.3f\n",ro)


if (teta<0.0):
        i=1
        teta=math.log(n)/math.log(10.0)
        log10N=teta
        while(log10N>i):
                teta*=math.log(n/(math.pow(10.0,log10N-(i))))/math.log(10.0)
                i=i+1

        teta/=2.0
        teta=teta+value*2.0/(n*(w[n-1]+w[0]))
        teta=((teta*10.0))/10.0
#   printf("teta=%.3f\n",teta)


niter1=niter1+1

kk=0
for i in range(start):
        kk=kk+w[i]

if (currvalue+kk<=bestsolk):
        return

alfa=alfa+1

i=start
k=0;
lsb1=-1
firstbit=-1
while(k<value and i>=0):
        niter=niter+1
        if (k+w[i]<=value):
                k=k+w[i]
                lsb1=i
                if (firstbit==-1):
                        firstbit=i
        else:
                if (lsb1>-1):
                        break
        i=i-1

lsbmin=0
lsbmax=lsb1-1
while(lsbmin+1<lsbmax):
        niter=niter+1
        m=(lsbmin+lsbmax)//2
        if (k+w[m]>value):
                lsbmax=m
        else:
                lsbmin=m

lsb2=lsbmax
if (lsb2>-1 and k+w[lsb2]<=value and  lsb2<lsb1):
        k=k+w[lsb2]
```

```
else:
        lsb2=lsbmin
        if (lsb2>-1 and k+w[lsb2]<=value and lsb2<lsb1):
                k=k+w[lsb2]
        else:
                lsb2=-1

#    printf("value-k=%I64d\n",value-k);
amax=k
oldamax=amax

basemax=firstbit+1
base=firstbit+1

while(base>1):
#       printf("base=%d LSB2=%d\n",base,lsb2)
#       getch()
        if (lsb2>=0):
                k=k-w[lsb2]
        i=base-1
        if (i>-1):
                k=k-w[i]
        i=lsb1-1
        lsb1=-1
        while(k<value and i>=0):
                niter=niter+1
                if (k+w[i]<=value):
                        k=k+w[i]
                        lsb1=i
                else:
                        if (lsb1>-1):
                                break
                i=i-1

        lsbmin=0
        lsbmax=lsb1-1
        while(lsbmin+1<lsbmax):
                niter=niter+1
                m=(lsbmin+lsbmax)//2
                if (k+w[m]>value):
                        lsbmax=m
                else:
                        lsbmin=m

        lsb2=lsbmax
        if (lsb2>=0 and k+w[lsb2]<=value and lsb2<lsb1):
                k=k+w[lsb2]
        else:
                lsb2=lsbmin
                if (lsb2>=0 and k+w[lsb2]<=value and lsb2<lsb1):
                        k=k+w[lsb2]
                else:
                        lsb2=-1

        nitems=base-1-lsb1
        if lsb2>-1:
                nitems=nitems+1
        if (k>amax and lsb2>=0 and lsb2<lsb1 and nitems<=ro):
                oldamax=amax
                amax=k
                basemax=base-1

        base=base-1
```

```
fbit=-1
sbit=-1
i=basemax-1
k=0
lsb1=-1
while(k<value and i>=0):
        niter=niter+1
        if (k+w[i]<=value):
                k=k+w[i]
                lsb1=i
                c1[i]=1
                if (fbit==-1):
                        fbit=i
                if (fbit>-1 and sbit==-1):
                        sbit=i
        else:
                if (lsb1>-1):
                        break
        i=i-1

lsbmin=0
lsbmax=lsb1-1
while(lsbmin+1<lsbmax):
        niter=niter+1
        m=(lsbmin+lsbmax)//2
        if (k+w[m]>value):
                lsbmax=m
        else:
                lsbmin=m

lsb2=lsbmax
if (lsb2>-1 and k+w[lsb2]<=value and lsb2<lsb1):
        k=k+w[lsb2]
        c1[lsb2]=1
else:
        lsb2=lsbmin
        if (lsb2>-1 and k+w[lsb2]<=value and lsb2<lsb1):
                k=k+w[lsb2]
                c1[lsb2]=1
        else:
                lsb2=-1

if (sbit<0):
        sbit=lsb2

k1=currvalue+k

if (k1>bestsolk):
        print("basemax="+str(basemax)+" sol="+str(k1))
        bestsolk=k1
        for zz in range(n):
                sol[zz]=c1[zz]

if (k==value):
        solfound=1
        alfa=alfa-1
        return

superbase=computeBase(c1)
lsb=0

amenaprimo=0;
bprimo=0;
b=-1;
```

```
    cbase = superbase;
    a=k;
    k=value-k;
    cnt=0;

    if (lsb1==-1):
            alfa=alfa-1
            return

    lsb=0
    while(c1[lsb]==0 and lsb<n):
            lsb=lsb+1
    if (lsb>=n):
            lsb=basemax+1
    lsb=lsb-1

    alfa1=1


    if (superbase<=start+1):
            while(solfound==0 and lsb>-1 and lsb<superbase-3 and teta*bprimo<=a-
amenaprimo and c1[fbit]==1 and c1[sbit]==1):
                    cnt=cnt+1
                    if (lsb>-1):
                            findMax(topvalue,topvalue-k,k,lsb)

                    for i1 in range(lsb+1):
                            c1[i1]=0

                    while(c1[lsb]==0 and lsb<superbase):
                            lsb=lsb+1
                    if (lsb>=superbase):
                            lsb=superbase-1
                    c1[lsb]=0
                    k=k+w[lsb]
                    amenaprimo=amenaprimo+w[lsb]
                    bprimo=value-(a-amenaprimo)
                    lsb-=1

    alfa=alfa-1
    return

try:
    inp = open("input.txt","rt")
    n=int(inp.readline())
    c=int(inp.readline())
    wmin=int(inp.readline())
    wmax=int(inp.readline())
    inp.close()

    w=sorted([int(wmin+randrange(wmax-wmin)) for i in range(n)])
    c1=[0 for i in range(n)]
    sol=[0 for i in range(n)]
    ro=-1.0
    teta=-1.0
    niter=0
    niter1=0
    bestsolk=0
    alfa=0

    print("n="+str(n))
    print("c="+str(c))
    print("w = ",w)
    print()
```

```
    findMax(c,0,c,n-1)
    solvalue=sum([sol[i]*w[i] for i in range(n)])
    print()
    print("Solution="+str(solvalue))
    print("Sol. Items = ",[w[i] for i in range(n) if sol[i]==1])

except Exception:
    print()
    print("ssp by Andrea Bianchini.")
    print("USAGE:")
    print("ssp")
    print("ssp reads input data from file input.txt")
    print("WHERE file input.txt is in the form:")
    print("n     # number of items")
    print("c     # capacity of the knapsack")
    print("wmin  # min item's weight")
    print("wmax  # max item's weight")
    print()
```