**Cryptography Lattice Project**

**Pramod Aravind Byakod – 113436879**

**Program:**

def first\_100\_primes():

prime\_list = []

first = 2

for i in range(100):

prime\_list.append(first)

first=first.next\_prime()

return prime\_list

prime\_list= first\_100\_primes()

n=100

id\_num=113436879\*2

init\_S=10^94\*id\_num

sum\_subset=[]

init\_M=matrix.identity(101)\*2

for i in range(n):

init\_M[i,n]=floor(10^100\*(prime\_list[i].n(prec=600).nth\_root(3)))

l\_row = [1]\*(n+1)

init\_M[n]=l\_row

init\_M[n,n]=init\_S

M=copy(init\_M)

err\_S=99/100

expr = 0

while expr!=94:

base=id\_num\*(10^expr) #id0

exp=101-len(str(base))

M\_fac=10^exp

S\_fac=10^(exp+2)

S\_dec = 0

while S\_dec!=199:

for i in range(n):

M[i,n]=init\_M[i,n]//M\_fac\*M\_fac

S=S\_fac\*(base-err\_S+S\_dec/100)

M[n,n]=S

#LLL\_algo=M.LLL()

BKZ\_algo=M.BKZ()

temp=1

for i in range(99,100,1):

if BKZ\_algo[i,n]!=0:

continue

else:

j=0

while j!=n:

if BKZ\_algo[i,j]!=1 and BKZ\_algo[i,j]!=-1:

temp=0

break

j=j+1

if temp!=0:

sub\_sum=0

vector=[]

j=0

while j!=n:

if BKZ\_algo[i,j]==-1:

vector.append(1)

sub\_sum=sub\_sum+init\_M[j,n]

else:

vector.append(0);

j=j+1

if sub\_sum>=init\_S:

sum\_subset.append(sub\_sum);

print 'Binary List:\n'+str(vector)

else:

temp=1

continue;

S\_dec = S\_dec+1

if not sum\_subset:

print str(expr)+' Result not found'

else:

print 'Exponent: '+str(expr)

print 'Sum of the subset:\n'+str(min(sum\_subset))

sum\_subset=[]

expr = expr +1

print 'Finished'

**Output:**

Binary List:

[1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Exponent: 0

Sum of the subset:

2268737580351338916369666901486990221267613974359993794545131028687475822032748270843492839074637277071

Binary List:

[1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Exponent: 1

Sum of the subset:

2268737580119321747317133738579288052915430533838902828334461606322834183582653498569850502603071580568

Binary List:

[0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Exponent: 2

Sum of the subset:

2268737580010580083155455786704004006644551687496979723004710084473244110173598517909834738353932264691

Binary List:

[0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Exponent: 3

Sum of the subset:

2268737580001471838158786556568666384932581097834915237223174067378038364857564288199076990809120613263

Binary List:

[0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Exponent: 4

Sum of the subset:

2268737580000098296655709919380025906424184486158492932967402495096043179295234573930940754763096264514

5 Result not found

6 Result not found

7 Result not found

8 Result not found

9 Result not found

10 Result not found

11 Result not found

12 Result not found

13 Result not found

14 Result not found

15 Result not found

16 Result not found

17 Result not found

18 Result not found

19 Result not found

20 Result not found

21 Result not found

22 Result not found

23 Result not found

24 Result not found

25 Result not found

26 Result not found

27 Result not found

28 Result not found

29 Result not found

30 Result not found

31 Result not found

32 Result not found

33 Result not found

34 Result not found

35 Result not found

36 Result not found

37 Result not found

38 Result not found

39 Result not found

40 Result not found

41 Result not found

42 Result not found

43 Result not found

44 Result not found

45 Result not found

46 Result not found

47 Result not found

48 Result not found

49 Result not found

50 Result not found

51 Result not found

52 Result not found

53 Result not found

54 Result not found

55 Result not found

56 Result not found

57 Result not found

58 Result not found

59 Result not found

60 Result not found

61 Result not found

62 Result not found

63 Result not found

64 Result not found

65 Result not found

66 Result not found

67 Result not found

68 Result not found

69 Result not found

70 Result not found

71 Result not found

72 Result not found

73 Result not found

74 Result not found

75 Result not found

76 Result not found

77 Result not found

78 Result not found

79 Result not found

80 Result not found

81 Result not found

82 Result not found

83 Result not found

84 Result not found

85 Result not found

86 Result not found

87 Result not found

88 Result not found

89 Result not found

90 Result not found

91 Result not found

92 Result not found

93 Result not found

Out of all the iterations, there are vectors produced for 5 iterations. Among all, the least number found in M that is greater than S is 2268737580000098296655709919380025906424184486158492932967402495096043179295234573930940754763096264514

And its binary vector list representation is

[0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

**Discussion of strategy**:

This problem can be approached using BKZ and LLL algorithms. Here I have considered BKZ over LLL. Reason being, BKZ is faster and produces better results in my case.

My ID is 113436879 and number of M is stored in a variable” init\_M”. We already know from the question that, entry of the set is 101 digits and target variable S has 103 digits. We add some modification to value of S to make first 9 digits ID. Apply the BKZ algorithm on it to find the subset sum that is in the range [2\*113436879-0.99,2\*13436879+0.99](\*10^94). Where 0.99 is the error rate. It’s been added to achieve the max error free subset. Also, the reason behind using 0.99 is that after modification of init\_m, the subset sum of m is sometimes equal to or less than the subset sum of init\_m. The worst case is that the subset has the full set of M, which is 100 and the 2 digits after the first 9 digits are 9 and 9, then although the subset sum is in the range [2\*113436879-0.99,0]\*(10^94), the corresponding subset sum of init\_m can still reach 2\*113436879\*(10^94).

To find the solution, after applying the BKZ algorithm we check if the matrix contains the vector having first 99 entries either 1 or -1 and last entry is 0, and the result is greater than what we set. So that results the result greater than our initially set value, meaning 2\*113436879\*(10^94) can be the benchmark on our way to find the more accurate solution. Then the job is to print the minimum solution found in this range.

Next, I set the digits after the first 10 digits of init\_m to be 0, and the range of target S be: [2\*113436879-0.99,2\*113436879+0.99]\*(10^94), and repeat the steps of above.

This is the basic idea behind this approach.