# LIS - Longest increasing subsequence

| Array's member | 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

**LIS = {6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98}**

$$A_{i+1} > A_i$$

**Problem:** Given sequence $X = \langle x_1, x_2, \ldots, x_n \rangle$, find the **longest subsequence** $Z = \langle z_1, z_2, \ldots, z_k \rangle$ that is $z_{i+1} > z_i$.

A subsequence is a subset of elements from the sequence with **strictly increasing** order (**not necessarily contiguous**).

# 1. Full Search חיפוש שלם

## 1.1 Step 1 : Build a array of all subsequences.

int[] plus1(int[] arr) method

**Pseudeocode:**

```
i = length of arr - 1
Loop ( i >= 0 and arr[i] equals 1 )
    loop begin
    arr[i] = 0
    i = i – 1
    loop end
if ( i >= 0)
    arr[i] = 1

return arr
```

int[][] allCombinations(int[] X)  method

**Pseudeocode:**

```
count = 2^(length of X)
list[][] – array of "count" elements
bin[] – array of 0 and 1
k = 0 (index for t[] array
Loop ( i from 0 to count step 1 )
    loop begin
    bin = plus1(bin)
    realLength = lenReal(bin)
    t[realLength] – array
    Loop (j from 0 to length of X step 1 )
        loop begin
        if ( bin[j] equals 1 )
            t[k] = X[j]
```

```
            k = k + 1
        loop end
    list[i] = t
    loop end

return list[]
```

```
int lenReal(int[] arr)   method
```

**Pseudeocode:**

```
res = 0;
Loop ( i from 0 to length of arr step 1 )
    loop begin
    if ( arr[i] equals 1 )
        res = res + 1
    loop end

return res
```

## 1.2 Step 2 : Build a array of all increasing subsequences.

```
int[][] buildIncreasingMatrix(int [][] mat)   method
```

**Pseudeocode:**

```
res[][]  ( array's length  - mat.length )
flag = true
k = 0
Loop ( i from 0 to mat.length step 1)
    loop begin
    Loop ( j from 1 to mat[i].length step 1)
        loop begin
        if (mat[i][j] < mat[i][j-1])
            flag = false
            break
        loop end
        if (flag)
```

```
            res[k] = mat[i]
            k = k + 1

        flag = true

    loop end

    res1[--k][]
    Loop (I from 0 to k step 1)
        res1[i] = res[i]

    return res1
```

## 1.3 Step 3 : Find a length of LIS.

```
int maxLength(int [][] mat)   method
```

**Pseudeocode:**

```
res = 1

Loop (i from 0 to mat.length step 1)
    loop begin
    len = mat[i].length
    if ( len > res )
        res = len
    loop end
return res
```

## 1.4 Step 4 : Build a array of all longest increasing subsequences.

```
int[][] LIS(int [][] mat, int len)   method
```

**Pseudeocode:**

```
res[][]      ( array's  length – mat.length )
k = 0
Loop ( i from 0 to mat.length step 1 )
     loop begin
     if (mat[i].length equals len )
          res[k] = mat[i]
          k = k + 1
     loop end
lis[k][]
Loop ( i from 0 to k step 1)
     lis[i] = res[i]
```

**return lis**

```
public static void main(String[] args) {
    int[] arr = {29, 6, 14, 31, 39, 78, 63, 50, 13, 64, 61, 62, 19, 64, 20, 70, 43, 84, 35, 98};
    int mat[][] = allCombinations(arr);
    int incMat[][] = buildIncreasingMatrix(mat);
    int max = maxLength(incMat);
    int lis[][] = LIS(incMat, max);
    printIntMatrix(lis);
}
```

# 2. Greedy Search חיפוש חמדני

int[] Greedy(int[] arr) method

**Pseudeocode:**

```
res[] – array ( length equals length of arr )
res[0] = arr[0]
k = 1
Loop ( i from 1  to length of arr step 1 )
    loop begin
    if ( arr[i] > res[k-1] )
        res[k] = arr[i]
        k = k + 1
    loop end
res1[] array ( length k )
Loop ( i from 0  to k step 1 )
    loop begin
```

```
    res1[i] = res[i]
    loop end

return res1
```

# 3. Dynamic Programming תכנון דינמי

**LIS dynamic programming ( length of subsequence ).**

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

29 index 0

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 index 0

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 index 1

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 31 index 2

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 31 39 index 3

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 31 39 78 index 4

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 31 39 63 index 4 39 < 63 < 78

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 14 31 39 50 index 4 39 < 50 < 63

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 31 39 50 index 1

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 31 39 50 65 index 5

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 31 39 50 61 index 5 50 < 61 < 65

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 31 39 50 61 62 index 6

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 39 50 61 62 index 2

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 39 50 61 62 64 index 7

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 50 61 62 64 index 3

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 50 61 62 64 70 index 8

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 43 61 62 64 70 index 4

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 43 61 62 64 70 84 index 9

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 35 61 62 64 70 84 index 4

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 6 | 14 | 31 | 39 | 78 | 63 | 50 | 13 | 65 | 61 | 62 | 19 | 64 | 20 | 70 | 43 | 84 | 35 | 98 |

6 13 19 20 35 61 62 64 70 84 98 index 10

6 13 19 20 35 61 62 64 70 84 98

6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98

# LIS dynamic programming

**6**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 13**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 13, 19**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 13, 19, 20**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 13, 19, 20, 35**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61, 62**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61, 62, 64**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61, 62, 64, 70**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61, 62, 64, 70, 84**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
**6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98**, 0, 0, 0, 0, 0, 0, 0, 0, 0,

**6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98**

```
int binarySearchBetweenArr(int []arr, int end, int value)
```

**Pseudeocode:**

```
low = 0
high = end
if (value < arr[0])          return  0
if (value > arr[end])        return  end+1

Loop ( low <= high )
    loop begin
    middle = (low + high)/2
    if (low == high)
            return  low
    else
        if ( arr[middle] == value)
            return middle
        if ( arr[middle] < value)
            high = middle
        else
            low = middle+1
    loop end
return -1
```

```
int binarySearchBetweenMat(int [][]arr, int end, int value)
```

**Pseudeocode:**

```
low = 0
high = end
if (value < arr[0][0])              return  0
if (value > arr[end][end])          return  end+1

Loop ( low <= high )
    loop begin
    middle = (low + high)/2
    if (low == high)
            return  low
    else
        if ( arr[middle][middle] == value)
            return middle
        if ( arr[middle][middle] < value)
                        high = middle
        else
            low = middle+1
```

```
    loop end
return -1
```

```
int LISLength(int [] arr)
        Pseudeocode:
                    size = length of arr
                    array d[size]
                    d[0] = arr[0]
                    end = 0
                    Loop ( i from 1 to size step 1 )
                        loop begin
                        index = binarySearchBetweenArr(d, end, arr[i])
                        d[index] = arr[i]
                        if (index > end)
                            end = end + 1
                        loop end
                    return  end+1
```

```
int[] LIS2(int [] arr)
        Pseudeocode:
                size = length of arr
                array mat[size][size]
                mat[0][0] = arr[0]
                end = 0
                Loop ( i from 1 to size step 1 )
                    loop begin
                    index = binarySearchBetweenMat(d, end, arr[i])
                    mat[index][index] = arr[i]
                    Loop ( j from 0 to index step 1 )
                        loop begin
                        mat[index][j] = mat[index-1][j]
                        loop end
                    mat[index][index] = arr[i]
                    if (index>end)
```

```
        end = end + 1
    loop end
array ans[end+1]
Loop ( j from 0 to index step 1 )
    loop begin
    ans[j] = mat[end][j]
return ans
```

[29, 6, 14, 31, 39, 78, 63, 50, 13, 65, 61, 62, 19, 64, 20, 70, 43, 84, 35, 98]

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 65, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 43, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 43, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 13, 19, 20, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

**[6, 14, 31, 39, 50, 61, 62, 64, 70, 84, 98]**