

# Problem Solving Intermediate

## 1. Maximizing Element With Constraints

### 1. Maximizing Element With Constraints

In this problem, the goal is to determine the maximum value of an element at a certain index in an array of integers that can be constructed under some constraints.

More specifically,  $n$  is the desired array size,  $maxSum$  is the maximum allowed sum of elements in the array, and  $k$  is the index of the element that needs its value to be maximized. The 0-indexed array has the following constraints:

1. The array consists of  $n$  positive integers.
2. The sum of all elements in the array is at most  $maxSum$ .
3. The absolute difference between any two consecutive elements in the array is at most 1.

What is the maximum value of the integer at index  $k$  in such an array?

For example, let's say  $n = 3$ ,  $maxSum = 6$ , and  $k = 1$ . So, the goal is to find the maximum value of the element at index 1 in an array of 3 positive integers, where the sum of elements is at most 6, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 2, and it can be achieved, for example, by the array [1, 2, 2]. This array has 3 elements, each of them a positive integer. The sum of the elements does not exceed 6, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index  $k = 1$ . Therefore, the answer is 2 because that is the maximum value of the integer at index  $k$ .

#### Function Description

Complete the function `maxElement` in the editor below. The function must return an integer denoting the maximum value of the element at index  $k$  given the above constraints.

`maxElement` has the following parameter(s):

```
int n: the size of the array
int maxSum: the maximum allowed sum of the elements in
above constraints
```

#### Constraints

- $1 \leq n \leq maxSum \leq 10^9$
- $1 \leq k \leq n$

#### Input Format For Custom Testing

The first line contains an integer,  $n$ , denoting the number of elements in the array.

The second line contains an integer,  $maxSum$ , denoting the maximum allowed sum of the elements in the array.

The third line contains an integer  $k$ , denoting the index of the element in the array where the value needs to be maximized.

#### Sample Case 0

##### Sample Input For Custom Testing

```
3
7
1
```

##### Sample Output

```
3
```

##### Explanation

In this case,  $n = 3$ ,  $maxSum = 7$ , and  $k = 1$ . So, the goal is to find the maximum value of an element at index 1 in an array of 3 positive integers, where the sum of elements is at most 7, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 3, and it is achieved, for example, by the array [2, 3, 2]. This array has 3 elements, each a positive integer. The sum of all elements does not exceed 7, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index  $k = 1$ . Therefore, the answer is 3 because that is the maximum value of the integer at index  $k$ .

#### Sample Case 1

Language Python 3 Environment

Autocomplete Ready

Line: 66 Col: 1

```
20
21 def maxElement(n, maxSum, k):
22     # Function to calculate the sum of an arithmetic sequence from 1 to x
23     def sequence_sum(x):
24         return (x * (x + 1)) // 2
25     left, right = 1, maxSum
26     while left < right:
27         mid = (left + right + 1) // 2
28         # Calculate the sum of elements with values less than or equal to mid,
29         # excluding the element at index k.
30         total = sequence_sum(mid - 1)
31         # If k is less than mid, add the contribution of the element at index k.
```

#### Test Results

#### Custom Input

Run Code

Run Tests

Submit

#### Compiled successfully. 5/15 test cases passed

Use print or log statements to debug why your hidden test cases are failing. Hidden test cases are used to evaluate if your code can handle different scenarios, including corner cases.

#### Test case 4

Your Output (stdout)

```
1 3
```

#### Test case 5

#### Test case 7

#### Test case 8

#### Test case 9

#### Test case 10

#### Hidden Test Case

Use print or log statements to debug why your hidden test cases are failing. Hidden test cases are used to evaluate if your code can handle different scenarios, including corner cases

Language Python 3 Environment

Autocomplete Ready

Line: 47 Col: 23

```
16 # 1. INTEGER n
17 # 2. INTEGER maxSum
18 # 3. INTEGER k
19 #
20
21 def maxElement(n, maxSum, k):
22     # Function to calculate the sum of an arithmetic sequence from 1 to x
23     def sequence_sum(x):
24         return (x * (x + 1)) // 2
25
26     left, right = 1, maxSum
27
28     while left < right:
29         mid = (left + right + 1) // 2
30
31         # Calculate the sum of elements with values less than or equal to mid,
32         # excluding the element at index k.
33         total = sequence_sum(mid - 1)
34
35         # If k is less than mid, add the contribution of the element at index k.
36         if k <= mid:
37             total += k
38
39         # If k is greater than mid, add the contribution of all elements from 1 to mid.
40         else:
41             total += sequence_sum(mid) - sequence_sum(mid - k)
42
43         # Calculate the sum of elements from k+1 to n.
44         total += (n - k)
45
46         if total <= maxSum:
47             left = mid
48         else:
49             right = mid - 1
50
51     return left
52 > if __name__ == '__main__': --
```

#### Test Results

#### Custom Input

Run Code

Run Tests

Submit

1

### Sample Output

3

#### Explanation

In this case,  $n = 3$ ,  $maxSum = 7$ , and  $k = 1$ . So, the goal is to find the maximum value of an element at index 1 in an array of 3 positive integers, where the sum of elements is at most 7, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 3, and it is achieved, for example, by the array [2, 3, 2]. This array has 3 elements, each a positive integer. The sum of all elements does not exceed 7, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index  $k = 1$ . Therefore, the answer is 3 because that is the maximum value of the integer at index  $k$ .

#### Sample Case 1

##### Sample Input For Custom Testing

4  
6  
2

### Sample Output

2

#### Explanation

In this case,  $n = 4$ ,  $maxSum = 6$ , and  $k = 2$ . So, the goal is to find the maximum value of an element at index 2 in an array of 4 positive integers, where the sum of elements is at most 6, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 2, and it is achieved, for example, by the array [1, 1, 2, 1]. This array has 4 elements, each a positive integer. The sum of all elements does not exceed 6, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index  $k = 2$ . Therefore, the answer is 2 because that is the maximum value of the integer at index  $k$ .

## 1. Maximizing Element With Constraints

In this problem, the goal is to determine the maximum value of an element at a certain index in an array of integers that can be constructed under some constraints.

More specifically,  $n$  is the desired array size,  $maxSum$  is the maximum allowed sum of elements in the array, and  $k$  is the index of the element that needs its value to be maximized. The 0-indexed array has the following constraints:

1. The array consists of  $n$  positive integers.
2. The sum of all elements in the array is at most  $maxSum$ .
3. The absolute difference between any two consecutive elements in the array is at most 1.

What is the maximum value of the integer at index  $k$  in such an array?

For example, let's say  $n = 3$ ,  $maxSum = 6$ , and  $k = 1$ . So, the goal is to find the maximum value of the element at index 1 in an array of 3 positive integers, where the sum of elements is at most 6, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 2, and it can be achieved, for example, by the array [1, 2, 2]. This array has 3 elements, each of them a positive integer. The sum of the elements does not exceed 6, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index  $k = 1$ . Therefore, the answer is 2 because that is the maximum value of the integer at index  $k$ .

#### Function Description

Complete the function `maxElement` in the editor below. The function must return an integer denoting the maximum value of the element at index  $k$  given the above constraints.

`maxElement` has the following parameter(s):

`int n`: the size of the array

`int maxSum`: the maximum allowed sum of the elements in the array

Language Python 3

Environment

Autocomplete Ready

Icons: eye, list, refresh, undo, redo, help, menu

```
16 # 1. INTEGER n
17 # 2. INTEGER maxSum
18 # 3. INTEGER k
19 #
20
21 def maxElement(n, maxSum, k):
22     # Function to calculate the sum of an arithmetic sequence from 1 to x
23     def sequence_sum(x):
24         return (x * (x + 1)) // 2
25
26     left, right = 1, maxSum
27
28     while left < right:
29         mid = (left + right + 1) // 2
30
31         # Calculate the sum of elements with values less than or equal to mid,
32         # excluding the element at index k.
33         total = sequence_sum(mid - 1)
34
35         # If k is less than mid, add the contribution of the element at index k.
36         if k <= mid:
37             total += k
38
39         # If k is greater than mid, add the contribution of all elements from 1 to mid.
40         else:
41             total += sequence_sum(mid) - sequence_sum(mid - k)
42
43         # Calculate the sum of elements from k+1 to n.
44         total += (n - k)
45
46         if total <= maxSum:
47             left = mid
48         else:
49             right = mid - 1
50
51     return left
52 > if __name__ == '__main__': --
```

Line: 52 Col: 27

### Test Results

### Custom Input

Run Code

Run Tests

Submit

Language Python 3

Environment

Autocomplete Ready

Icons: eye, list, refresh, undo, redo, help, menu

```
38
39
40     # If k is greater than mid, add the contribution of all elements from 1 to mid.
41     else:
42         total += sequence_sum(mid) - sequence_sum(mid - k)
43
44     # Calculate the sum of elements from k+1 to n.
45     total += (n - k)
46
47     if total <= maxSum:
48         left = mid
49     else:
50         right = mid - 1
51
52     return left
53 > if __name__ == '__main__': --
```

Line: 52 Col: 27

### Test Results

### Custom Input

Run Code

Run Tests

Submit

### Compiled successfully. 5/15 test cases passed

Use print or log statements to debug why your hidden test cases are failing. Hidden test cases are used to evaluate if your code can handle different scenarios, including corner cases.

✗ Test case 14

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 6

Input (stdin)

Run as Custom Input Download

1 4  
2 6  
3 2

Your Output (stdout)

1 2

Expected Output

Download

1 2

## 2. Bitwise AND

### 2. Bitwise AND

Given an array of non-negative integers, count the number of unordered pairs of array elements such that their [bitwise AND](#) is a power of 2.

For example, let's say the array is  $arr = [10, 7, 2, 8, 3]$ , and let  $'\&'$  denote the bitwise AND operator. There are 6 unordered pairs of its elements that have a bitwise AND that is a power of two:

- For indices (0,1),  $10 \& 7 = 2$ , which is a power of 2.
- For indices (0,2),  $10 \& 2 = 2$ , which is a power of 2.
- For indices (0,3),  $10 \& 8 = 8$ , which is a power of 2.
- For indices (0,4),  $10 \& 3 = 2$ , which is a power of 2.
- For indices (1,2),  $7 \& 2 = 2$ , which is a power of 2.
- For indices (2,4),  $2 \& 3 = 2$ , which is a power of 2.

Therefore, the answer is 6.

#### Function Description

Complete the function `countPairs` in the editor below.

`countPairs` has the following parameter:

`int arr[n]`: an array of integers

Returns:

`int`: the number of unordered pairs of elements of `arr` such that their bitwise AND is a power of 2

#### Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $0 \leq arr[i] < 2^{12}$

#### Input Format For Custom Testing

The first line contains an integer,  $n$ , denoting the number of elements in `arr`.

Each line  $i$  of the  $n$  subsequent lines (where  $0 \leq i < n$ ) contains an integer describing `arr[i]`.

4m 56s left

⌕

ALL

ⓘ

✓

✓

▼ Input Format For Custom Testing

The first line contains an integer,  $n$ , denoting the number of elements in `arr`.  
Each line  $i$  of the  $n$  subsequent lines (where  $0 \leq i < n$ ) contains an integer describing `arr[i]`.

▼ Sample Case 0

Sample Input For Custom Testing

STDIN	Function
4	$\Rightarrow n = 4$
1	$\Rightarrow arr = [1, 2, 1, 3]$
2	
1	
3	

Sample Output

4

Explanation

All unordered pair of elements whose bitwise AND is a power of 2 are:

- For indices (0,2),  $1 \& 1 = 1$ , which is a power of 2.
- For indices (0,3),  $1 \& 3 = 1$ , which is a power of 2.
- For indices (1,3),  $2 \& 3 = 2$ , which is a power of 2.
- For indices (2,3),  $1 \& 3 = 1$ , which is a power of 2.

Therefore, the answer is 4.

▼ Sample Case 1

Sample Input For Custom Testing

3

6

2

4

Sample Output

0

Explanation

There are no pairs of array elements such that their bitwise AND is a power of 2. Therefore, the answer is 0.

Language Python 3

Environment

Autocomplete Ready

🔍 📄 🔄 ⌂ ?

```
1 > #!/bin/python3--
10 #
11 # Complete the 'countPairs' function below.
12 #
13 # The function is expected to return a LONG_INTEGER.
14 # The function accepts INTEGER_ARRAY arr as parameter.
15 #
16 from collections import defaultdict
17 def countPairs(arr):
18     # Write your code here
19     po2 = lambda x: x > 0 and not(x & (x - 1))
20     d = defaultdict(int)
21     for x in arr:
22         d[x] += 1
23     d = list(d.items())
24     ans = 0
25     for i in range(len(d)):
26         a, a_cnt = d[i]
27         for j in range(i, len(d)):
28             b, b_cnt = d[j]
29             if po2(a & b):
30                 if a == b:
31                     ans += (a_cnt * (a_cnt - 1)) // 2
32                 else:
33                     ans += a_cnt * b_cnt
34     return ans
35
36 > if __name__ == '__main__': --
```

Test Results Custom Input

Run Code Run Tests Submit

Language Python 3

Environment

Autocomplete Ready

🔍 📄 🔄 ⌂ ?

```
20 d = defaultdict(int)
21 for x in arr:
22     d[x] += 1
23 d = list(d.items())
24 ans = 0
25 for i in range(len(d)):
26     a, a_cnt = d[i]
27     for j in range(i, len(d)):
28         b, b_cnt = d[j]
29         if po2(a & b):
30             if a == b:
31                 ans += (a_cnt * (a_cnt - 1)) // 2
32             else:
33                 ans += a_cnt * b_cnt
34 return ans
```

Test Results Custom Input

Run Code Run Tests Submit

Compiled successfully. All available test cases passed

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin)

Run as Custom Input Download

1 4

2 3

3 3

4 3

5 3

Your Output (stdout)

Download

1 0

Expected Output

Download

1 0