# Problem 1798: Maximum Number of Consecutive Values You Can Make

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

You are given an integer array

coins

of length

n

which represents the

n

coins that you own. The value of the

i

th

coin is

coins[i]

. You can

make

some value

x

if you can choose some of your

n

coins such that their values sum up to

x

.

Return the

maximum number of consecutive integer values that you

can

make

with your coins

starting

from and

including

0

.

Note that you may have multiple coins of the same value.

Example 1:

Input:

coins = [1,3]

Output:

2

Explanation:

You can make the following values: - 0: take [] - 1: take [1] You can make 2 consecutive integer values starting from 0.

Example 2:

Input:

coins = [1,1,1,4]

Output:

8

Explanation:

You can make the following values: - 0: take [] - 1: take [1] - 2: take [1,1] - 3: take [1,1,1] - 4: take [4] - 5: take [4,1] - 6: take [4,1,1] - 7: take [4,1,1,1] You can make 8 consecutive integer values starting from 0.

Example 3:

Input:

coins = [1,4,10,3,1]

Output:

20

Constraints:

coins.length == n

1 <= n <= 4 * 10

4

1 <= coins[i] <= 4 * 10

4

## Code Snippets

**C++:**

```
class Solution {
public:
int getMaximumConsecutive(vector<int>& coins) {

}
};
```

**Java:**

```
class Solution {
public int getMaximumConsecutive(int[] coins) {

}
}
```

**Python3:**

```
class Solution:
def getMaximumConsecutive(self, coins: List[int]) -> int:
```

**Python:**

```
class Solution(object):
def getMaximumConsecutive(self, coins):
```

```
"""
:type coins: List[int]
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[]} coins
* @return {number}
*/
var getMaximumConsecutive = function(coins) {

};
```

**TypeScript:**

```
function getMaximumConsecutive(coins: number[]): number {

};
```

**C#:**

```
public class Solution {
public int GetMaximumConsecutive(int[] coins) {

}
}
```

**C:**

```
int getMaximumConsecutive(int* coins, int coinsSize) {

}
```

**Go:**

```
func getMaximumConsecutive(coins []int) int {

}
```

**Kotlin:**

```
class Solution {
fun getMaximumConsecutive(coins: IntArray): Int {

}
}
```

**Swift:**

```
class Solution {
func getMaximumConsecutive(_ coins: [Int]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn get_maximum_consecutive(coins: Vec<i32>) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[]} coins
# @return {Integer}
def get_maximum_consecutive(coins)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[] $coins
* @return Integer
*/
function getMaximumConsecutive($coins) {

}
}
```

**Dart:**

```
class Solution {
int getMaximumConsecutive(List<int> coins) {

}
}
```

**Scala:**

```
object Solution {
def getMaximumConsecutive(coins: Array[Int]): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec get_maximum_consecutive(coins :: [integer]) :: integer
def get_maximum_consecutive(coins) do

end
end
```

**Erlang:**

```
-spec get_maximum_consecutive(Coins :: [integer()]) -> integer().
get_maximum_consecutive(Coins) ->
.
```

**Racket:**

```
(define/contract (get-maximum-consecutive coins)
(-> (listof exact-integer?) exact-integer?)
)
```

## Solutions

**C++ Solution:**

```
/*
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int getMaximumConsecutive(vector<int>& coins) {

}
};
```

### Java Solution:

```
/**
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int getMaximumConsecutive(int[] coins) {

}
}
```

### Python3 Solution:

```
"""
Problem: Maximum Number of Consecutive Values You Can Make
Difficulty: Medium
Tags: array, greedy, sort
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def getMaximumConsecutive(self, coins: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

```
class Solution(object):
def getMaximumConsecutive(self, coins):
"""
:type coins: List[int]
:rtype: int
"""
```

**JavaScript Solution:**

```
/**
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} coins
* @return {number}
*/
var getMaximumConsecutive = function(coins) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function getMaximumConsecutive(coins: number[]): number {

};
```

### C# Solution:

```
/*
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int GetMaximumConsecutive(int[] coins) {

}
}
```

### C Solution:

```
/*
* Problem: Maximum Number of Consecutive Values You Can Make
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

int getMaximumConsecutive(int* coins, int coinsSize) {

}
```

### Go Solution:

```
// Problem: Maximum Number of Consecutive Values You Can Make
// Difficulty: Medium
// Tags: array, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func getMaximumConsecutive(coins []int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun getMaximumConsecutive(coins: IntArray): Int {

}
}
```

### Swift Solution:

```
class Solution {
func getMaximumConsecutive(_ coins: [Int]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Maximum Number of Consecutive Values You Can Make
// Difficulty: Medium
// Tags: array, greedy, sort
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn get_maximum_consecutive(coins: Vec<i32>) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {Integer[]} coins
# @return {Integer}
def get_maximum_consecutive(coins)

end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $coins
* @return Integer
*/
function getMaximumConsecutive($coins) {

}
}
```

### Dart Solution:

```
class Solution {
int getMaximumConsecutive(List<int> coins) {

}
}
```

### Scala Solution:

```
object Solution {
def getMaximumConsecutive(coins: Array[Int]): Int = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec get_maximum_consecutive(coins :: [integer]) :: integer
def get_maximum_consecutive(coins) do

end
end
```

### Erlang Solution:

```
-spec get_maximum_consecutive(Coins :: [integer()]) -> integer().
get_maximum_consecutive(Coins) ->
.
```

### Racket Solution:

```
(define/contract (get-maximum-consecutive coins)
(-> (listof exact-integer?) exact-integer?)
)
```