518. Coin Change 2

For Practice

Medium 5057 100 Add to List Share

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the number of combinations that make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `0`.

You may assume that you have an infinite number of each kind of coin.

The answer is **guaranteed** to fit into a signed **32-bit** integer.

Example 1:

```
Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1
```

Example 2:

```
Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.
```

Example 3:

```
Input: amount = 10, coins = [10]
Output: 1
```

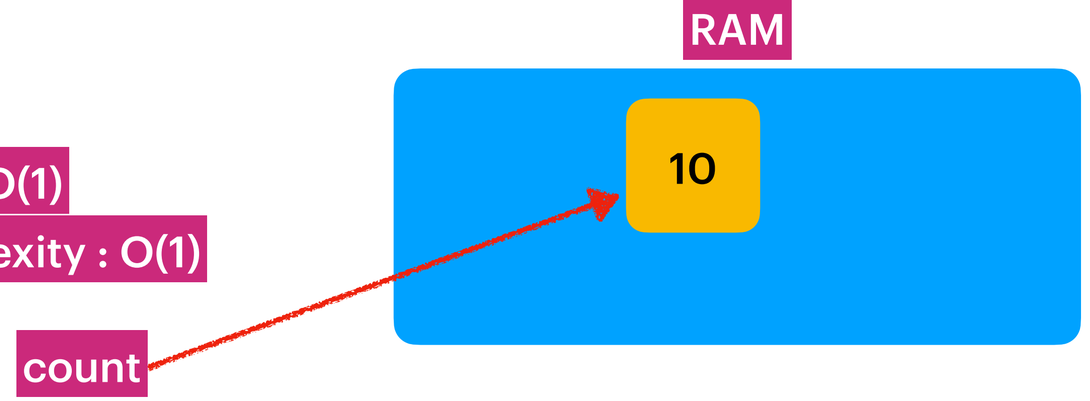
Constraints:

- `1 <= coins.length <= 300`
- `1 <= coins[i] <= 5000`
- All the values of `coins` are **unique**.
- `0 <= amount <= 5000`

int count = 10;

Time Complexity : O(1)

Space Complexity : O(1)



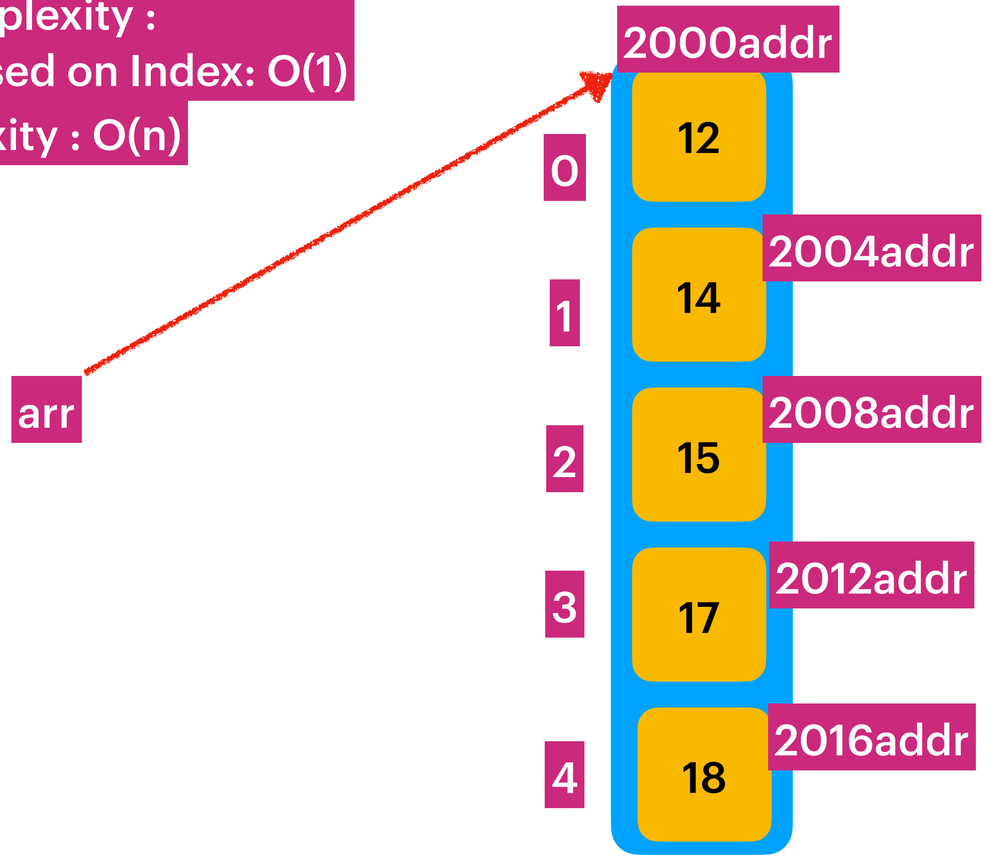
int count = 10;

Array

Time Complexity :
Access Element Based on Index: O(1)

Space Complexity : O(n)

int[] arr = {12,14,15,17,18};

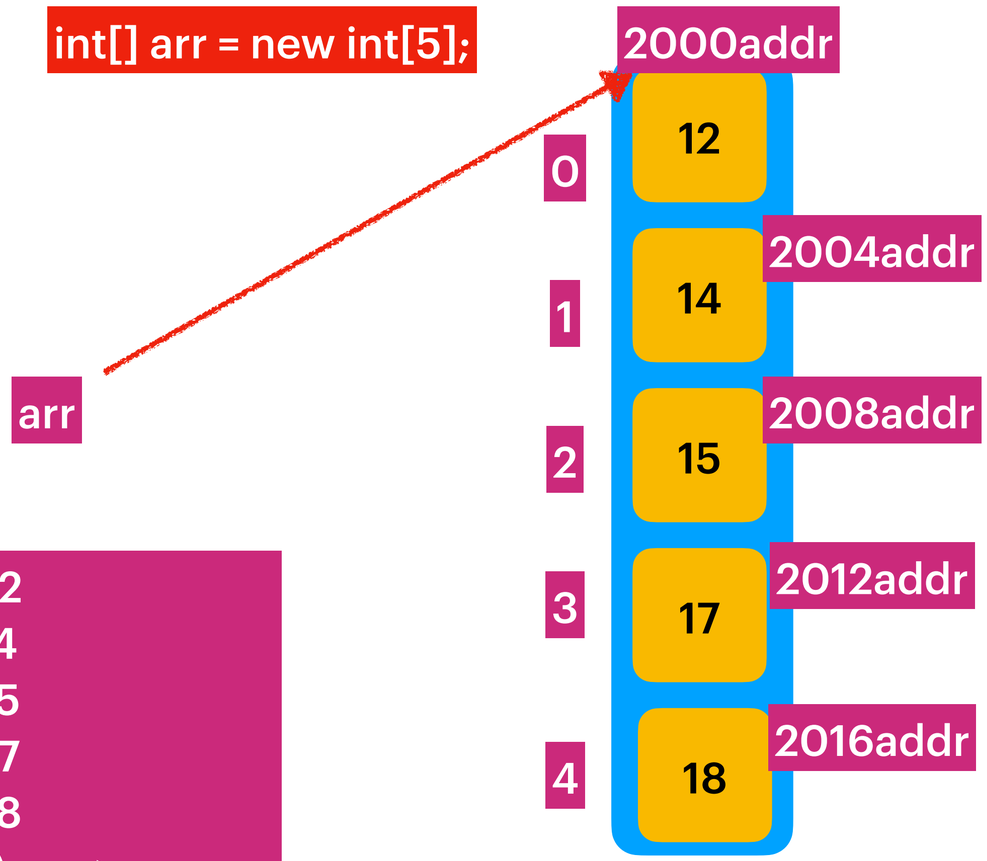


arr[0] = 2000addr + 0 * 4 = 2000addr = 12

arr[3] = 2000addr + 3 * 4 = 2012addr = 17

arr[4] = 2000addr + 4 * 4 = 2016addr = 18

Array Fixed in Size.



arr[0] = 12
arr[1] = 14
arr[2] = 15
arr[3] = 17
arr[4] = 18
arr[5] = 20 (X) ——> ArrayIndexOutOfBoundsException

ArrayList Increases the Size Dynamically.

Design ArrayList [Dynamic Array]

Space Complexity : O(n)

`int[] arr = new int[5] ;` —> Initial capacity :5

`arr.length > capacity/2 ;` —-> If the array occupied 50%

1. Double the capacity, capacity = 2 *capacity = 2 *5 = 10
2. Initialise new arry with the current capacity
   int[] newArr = new int[10];
3. Copy elements of old array into new Array.
4. Reassign the newArr object to oldArr reference
   arr = newArr;

Time Complexity : O(1)
`add(int element) ;`

Time Complexity : O(n)
`remove(int index) ;`

Time Complexity : O(1)
`get(int index) ;`

Time Complexity : O(n)
`search(int element) ;`

Time Complexity : O(n)
`add(int index, int element) ;`

Time Complexity : O(1)
`replaceOrSet(int index, int element) ;`