Dash
All
Articles
Videos
Problems
Quiz
« Prev
Next »

Buy Maximum Items with Given Sum

Given an array **arr[]** consisting of the cost of toys and an integer **K** depicting the amount of money available to purchase toys. The task is to find the maximum number of toys one can buy with the amount **K**.
Note: One can buy only 1 quantity of a particular toy.

Examples:

> ***Input:*** *arr[] = {1, 12, 5, 111, 200, 1000, 10, 9, 12, 15}, K = 50*
> ***Output:*** *6*
> *Toys with amount 1, 5, 9, 10, 12, and 12*
> *can be purchased resulting in a total amount of 49.*
> *Hence, the maximum number of toys are 6.*
>
> ***Input:*** *arr[] = {1, 12, 5, 111, 200, 1000, 10}, K = 50*
> ***Output:*** *4*

Approach: Insert all the elements of the given array in a priority_queue now one by one remove elements from this priority queue and add these costs in a variable **sum** initialised to **0**. Keep removing the elements while the new addition keep the sum smaller than **K**. In the end, the count of elements removed will be the required answer.

C++ Java

```java
// Java implementation of the approach
import java.io.*;
import java.util.*;

class GFG {

    // Function to return the count of
    // maximum toys that can be bought
    public static int maxToys(int[] arr, int k)
    {
        int n = arr.length;

        // Create a priority_queue and push
        // all the array elements in it
        PriorityQueue<Integer> pq
            = new PriorityQueue<Integer>();
        for (int i = 0; i < n; i++) {
            pq.offer(arr[i]);
        }

        // To store the count of maximum
        // toys that can be bought
        int count = 0;
        while (!pq.isEmpty() && pq.peek() <= k) {
            k = k - pq.poll();
            count++;
        }
        return count;
    }

    // Driver code
    public static void main(String[] args)
    {
        int[] arr
            = new int[] { 1, 12, 5, 111, 200, 1000, 10 };
        int k = 50;

        System.out.println(maxToys(arr, k));
```

```
        }
    }
```


Output:

4

Time Complexity: O(N*logN)
Auxiliary Space: O(N)