## 7335. Saucepans and lids

A huge disaster occurred this morning at the café where you used to have snacks during your university studies. The cleaner, Larisa Ivanovna, accidentally knocked over one of the cabinets while sweeping the floor, causing all the kitchen utensils stored inside to scatter across the floor. Fortunately, it only contained saucepans with lids. However, some of them got bent or broken, so they had to be thrown away.

Now the schoolmaster wants to calculate the losses and determine how many new saucepans and lids should be purchased. But first, it is necessary to find out how many remaining saucepans can be covered by the remaining lids.

The saucepans and lids are round. A lid can cover a saucepans only if its radius is not less than the radius of the pot.

**Input.** The first line contains integers *n*, *m* (1 ≤ *n*, *m* ≤ 1000) – the number of remaining saucepans and lids. The second line contains *n* integers *ai* (1 ≤ *ai* ≤ 1000) – the radii of the remaining saucepans. The third line contains *m* integers *bi* (1 ≤ *bi* ≤ 1000) – the radii of the remaining lids.

**Output.** Print one number – the largest number of saucepans that can be covered by the available lids.

|  |  |
| --- | --- |
| **Sample input** | **Sample output** |
| 5 5  4 8 1 2 5  7 2 4 6 5 | 4 |

## SOLUTION

**greedy**

# Algorithm analysis

Sort the radii of the lids and the radii of the saucepans in ascending order. For the smallest saucepan, find the smallest lid that can cover it. Then, for the second smallest pan, find the smallest lid that fits it, and so on. The answer will be the number of saucepans that can be covered with lids.

**Example**

Let’s find the maximum number of pans for which lids can be matched in the given example.



# Algorithm implementation

Declare arrays to store the radii of saucepans and lids.

#define MAX 1010

int pan[MAX], lid[MAX];

Read the input data.

scanf("%d %d",&n,&m);

for(i = 0; i < n; i++)

scanf("%d",&pan[i]);

for(i = 0; i < m; i++)

scanf("%d",&lid[i]);

Sort the radii of pans and lids.

sort(pan,pan+n);

sort(lid,lid+m);

Using the greedy method, search for the smallest lid each time that can cover the smallest pan.

for (i = j = 0; (i < n) && (j < m); j++)

if (pan[i] <= lid[j]) i++;

Print the number of covered pans.

printf("%d\n",i);

**Java implementation**

**import** java.util.\*;

**public** **class** Main

{

**public** **static** **void** main(String[] args)

{

Scanner con = **new** Scanner(System.***in***);

**int** n = con.nextInt();

**int** m = con.nextInt();

Integer pan[] = **new** Integer[n];

**for**(**int** i = 0; i < n; i++)

pan[i] = con.nextInt();

Integer lid[] = **new** Integer[n];

**for**(**int** i = 0; i < m; i++)

lid[i] = con.nextInt();

Arrays.*sort*(pan);

Arrays.*sort*(lid);

**int** i = 0;

**for**(**int** j = 0; (i < n) && (j < m); j++)

**if** (pan[i] <= lid[j]) i++;

System.***out***.println(i);

con.close();

}

}

**Python implementation**

Read the input data.

n, m = map(int,input().split())

pan = list(map(int,input().split()))

lid = list(map(int,input().split()))

Sort the radii of pans and lids.

pan.sort()

lid.sort()

Using the greedy method, search for the smallest lid each time that can cover the smallest pan.

i = j = 0

**while** i < n **and** j < m:

**if** pan[i] <= lid[j]: i += 1

j += 1

Print the number of covered pans.

print(i)