## 7335. Кастрюли и крышки

Огромное бедствие произошло сегодня утром в кафе, в котором Вы привыкли перекусывать во время учебы в университете. Уборщица Лариса Ивановна во время подметания пола уронила один из шкафов, в котором хранились все кухонные принадлежности. Все содержимое шкафа было разбросано по полу. К счастью, он содержал только кастрюли с крышками. Тем не менее, некоторые из них погнулись или сломались, поэтому были выброшены.

Теперь школьный учитель хочет подсчитать потери и выяснить, как много новых кастрюль и крышек следует купить. Но сначала следует выяснить, сколько оставшихся кастрюль можно накрыть оставшимися крышками.

Кастрюли и крышки круглые. Крышка может покрыть кастрюлю, если только радиус крышки не меньше радиуса кастрюли.

**Вход.** Первая строка содержит числа *n*, *m* (1 ≤ *n*, *m* ≤ 1000) – количество оставшихся кастрюль и крышек. Вторая строка содержит *n* целых чисел *ai* (1 ≤ *ai* ≤ 1000) – радиусы оставшихся кастрюль. Третья строка содержит *m* целых чисел *bi* (1 ≤ *bi* ≤ 1000) – радиусы оставшихся крышек.

**Выход.** Выведите одно число – наибольшее количество кастрюль, которое может быть покрыто имеющимися крышками.

|  |  |
| --- | --- |
| **Пример входа** | **Пример выхода** |
| 5 54 8 1 2 57 2 4 6 5 | 4 |

## РЕШЕНИЕ

**жадность**

**Анализ алгоритма**

Отсортируем по возрастанию радиусы крышек и радиусы кастрюль. Для самой маленькой кастрюли найдем наименьшую крышку, которой ее можно накрыть. Далее для второй наименьшей кастрюли найдем наименьшую ей подходящую крышку и так далее. Ответом будет количество кастрюль, которое можно покрыть крышками.

**Пример**

Рассмотрим какому наибольшему числу кастрюль можно подобрать крышки для заданного примера.



**Реализация алгоритма**

Объявим массивы, в которых будем хранить радиусы кастрюль и крышек.

#define MAX 1010

int pan[MAX], lid[MAX];

Читаем входные данные.

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]);

Сортируем радиусы кастрюль и крышек.

sort(pan,pan+n);

sort(lid,lid+m);

Используя жадный метод, ищем каждый раз наименьшую крышку, которой можно накрыть наименьшую кастрюлю.

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

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

Выводим количество накрытых кастрюль.

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

**Java реализация**

**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 реализация**

Читаем входные данные.

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

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

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

Сортируем радиусы кастрюль и крышек.

pan.sort()

lid.sort()

Используя жадный метод, ищем каждый раз наименьшую крышку, которой можно накрыть наименьшую кастрюлю.

i = j = 0

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

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

 j += 1

Выводим количество накрытых кастрюль.

print(i)