

# GREEDY ALGORITHM

Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanation:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int main() {
4     int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
5     int n = sizeof(denominations) / sizeof(denominations[0]);
6     int V;
7     scanf("%d", &V);
8     int count = 0;
9     for (int i = 0; i < n; i++) {
10         while (V >= denominations[i]) {
11             V -= denominations[i];
12             count++;
13         }
14     }
15     printf("%d\n", count);
16     return 0;
17 }
18 }
```

	Input	Expected	Got
✓	49	5	5 ✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child  $i$  has a greed factor  $g[i]$ , which is the minimum size of a cookie that the child will be content with; and each cookie  $j$  has a size  $s[j]$ . If  $s[j] \geq g[i]$ , we can assign the cookie  $j$  to the child  $i$ , and the child  $i$  will be content. Your goal is to maximize the number of your content children and output the maximum number.

**Example 1:**

**Input:**

```
3  
1 2 3  
2  
1 1
```

**Output:**

```
1
```

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

**Constraints:**

```
1 <= g.length <= 3 * 10^4  
0 <= s.length <= 3 * 10^4  
1 <= g[i], s[j] <= 2^31 - 1
```

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>  
2  
3 void bubbleSort(int arr[], int n) {  
4     for (int i = 0; i < n - 1; i++) {  
5         for (int j = 0; j < n - i - 1; j++) {  
6             if (arr[j] > arr[j + 1]) {  
7                 int temp = arr[j];  
8                 arr[j] = arr[j + 1];  
9                 arr[j + 1] = temp;  
10            }  
11        }  
12    }  
13 }  
14  
15 int main() {  
16     int n, m;  
17     scanf("%d", &n);  
18     int g[n];  
19     for (int i = 0; i < n; i++) {  
20         scanf("%d", &g[i]);  
21     }  
22     scanf("%d", &m);  
23     int s[m];  
24     for (int i = 0; i < m; i++) {  
25         scanf("%d", &s[i]);  
26     }  
27     bubbleSort(g, n);  
28     bubbleSort(s, m);  
29     int i = 0, j = 0, content = 0;  
30     while (i < n && j < m) {  
31         if (s[j] >= g[i]) {  
32             content++;  
33             i++;  
34             j++;  
35         } else {  
36             j++;  
37         }  
38     }  
39     printf("%d\n", content);  
40     return 0;  
41 }  
42  
43
```

	Input	Expected	Got
✓	2	2	2 ✓
	1 2		
	3		
	1 2 3		

Passed all tests! ✓

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person burns out  $3^i$  kilometers. If he has eaten  $i$  burgers with  $c$  calories each, then he has to run at least  $3^i * c$  kilometers to burn out those calories. For example, if he eats burgers with calorie counts [1, 3, 2] in the order: [1, 3, 2], the kilometers he needs to run are  $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 18$ . But this is not the minimum, so need to try out other orders of consumption and choose the minimum value.

Note: He can eat burger in any order and use an efficient sorting algorithm. Apply greedy algorithm to solve this problem.

#### Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is  $n$  space-separated integers

#### Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

#### Sample Input

```
3
5 10 7
```

#### Sample Output

```
76
```

#### For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     int n;
6     int dist = 0;
7     scanf("%d", &n);
8     int arr[100];
9     for (int i = 0; i < n; i++) {
10         scanf("%d", &arr[i]);
11     }
12     for (int i = 0; i < n; i++) {
13         for (int j = 0; j < n; j++) {
14             if (arr[i] > arr[j]) {
15                 int temp = arr[i];
16                 arr[i] = arr[j];
17                 arr[j] = temp;
18             }
19         }
20     }
21     for (int i = 0; i < n; i++) {
22         dist += pow(n, i) * arr[i];
23     }
24     printf("%d", dist);
25 }
```

	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓

Passed all tests! ✓

Given an array of N integer, we have to maximize the sum of  $\text{arr}[i] * i$ , where  $i$  is the index of the element ( $i = 0, 1, 2, \dots, N$ ). Write an algorithm based on Greedy technique with a Complexity  $O(n\log n)$ .

Input Format:

First line specifies the number of elements-n.

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

```
5  
2 5 3 4 0
```

Sample output:

```
40
```

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3  
4 int compare(const void* a, const void* b) {  
5     return *(int*)a - *(int*)b;  
6 }  
7 int main() {  
8     int n;  
9     scanf("%d", &n);  
10    int arr[n];  
11    for (int i = 0; i < n; i++) {  
12        scanf("%d", &arr[i]);  
13    }  
14    qsort(arr, n, sizeof(int), compare);  
15    int sum = 0;  
16    for (int i = 0; i < n; i++) {  
17        sum += arr[i] * i;  
18    }  
19    printf("%d\n", sum);  
20    return 0;  
21 }  
22 |
```

	Input	Expected	Got	
✓	5 2 5 3 4 0	40	40	✓
✓	10 2 2 4 4 3 3 5 5 5	191	191	✓
✓	2 45 3	45	45	✓

Given two arrays array\_One[] and array\_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs( 1 element from each) is minimum. That is SUM (A[i] \* B[i]) for all i is minimum.

For example:

Input	Result
3	28
1	
2	
3	
4	
5	
6	

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int asc(const void* a, const void* b) {
5     return (*(int*)a - *(int*)b);
6 }
7
8 int desc(const void* a, const void* b) {
9     return (*(int*)b - *(int*)a);
10 }
11
12 int main() {
13     int n;
14     scanf("%d", &n);
15     int A[n], B[n];
16     for (int i = 0; i < n; i++) {
17         scanf("%d", &A[i]);
18     }
19     for (int i = 0; i < n; i++) {
20         scanf("%d", &B[i]);
21     }
22     qsort(A, n, sizeof(int), asc);
23     qsort(B, n, sizeof(int), desc);
24     int sum = 0;
25     for (int i = 0; i < n; i++) {
26         sum += A[i] * B[i];
27     }
28     printf("%d\n", sum);
29     return 0;
30 }
31
```

Input	Expected	Got	
3 1 2 3 4 5 6	28	28	✓
4 7 5 1 2 1 3 4 1	22	22	✓
5 20 10 30 10 40 8 9 4 3 10	590	590	✓

Passed all tests! ✓