

# GREEDY ALGORITHM

## QUESTION 3.A

### AIM:

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

Explanaton:

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

### ALGORITHM:

#### Step 1: Start

**Step 2:** Input the integer  $v$ , the amount for which denominations are needed.

**Step 3:** Initialize an array `denominations` with values {1000, 500, 100, 50, 20, 10, 5, 2, 1}.

**Step 4:** Initialize `count` to 0 to keep track of the total number of denominations.

**Step 5:** For each denomination in `denominations`:

- Divide  $v$  by the current denomination to find how many of that denomination are needed and add the result to `count`.
- Update  $v$  to the remainder after division.

**Step 6:** Print the value of `count`.

**Step 7: Stop**

**PROGRAM :**

```
#include <stdio.h>
int main() {
    int v;
    scanf("%d", &v);
    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int count = 0;
    for (int i = 0; i < sizeof(denominations) / sizeof(denominations[0]); i++) {
        count += v / denominations[i];
        v %= denominations[i];
    }
    printf("%d\n", count);
    return 0;
}
```

**OUTPUT:**

	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

**RESULT :**

The above program is executed successfully.

## QUESTION 3.B

### AIM:

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
```

### ALGORITHM:

#### Step 1: Start

**Step 2:** Input the integer  $n$ , the number of elements in array  $g$ .

**Step 3:** Input  $n$  integers into array  $g$ .

**Step 4:** Input the integer  $m$ , the number of elements in array  $c$ .

**Step 5:** Input  $m$  integers into array  $c$ .

**Step 6:** Initialize  $co$  to 0 to count compatible pairs.

**Step 7:** For each element in  $g$ , check if there exists an element in  $c$  such that  $c[i] \leq g[j]$ :

- If a compatible element is found, increment  $co$  and stop checking further for that  $g[j]$ .

**Step 8:** Print the value of  $co$ .

**Step 9: Stop**

### PROGRAM :

```

#include <stdio.h>
int main() {
    int n, m, co=0;
    scanf("%d", &n);
    int g[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &g[i]);
    }
    scanf("%d", &m);
    int c[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &c[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            if(c[i]<=g[j])
            {
                co++;
                break;
            }
        }
    }
    printf("%d\n", co);
}

```

OUTPUT:

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

Passed all tests! ✓

RESULT :

The above program is executed successfully.

QUESTION 3.C

AIM:

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories. If he has eaten  $i$  burgers with  $c$  calories each, then he has to run at least  $3^i * c$  kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are  $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$ . But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm. Apply greedy approach to solve the problem.

#### Input Format

First Line contains the number of burgers

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

#### ALGORITHM:

##### Step 1: Start

**Step 2:** Input the integer  $n$ , the number of elements in array  $c$ .

**Step 3:** Input  $n$  integers into array  $c$ .

**Step 4:** Sort the array  $c$  in descending order.

**Step 5:** Initialize  $k$  to 0 to store the weighted sum.

**Step 6:** For each element  $c[i]$ , calculate  $c[i] * n^i$  and add it to  $k$ .

**Step 7:** Print the value of  $k$ .

##### Step 8: Stop

#### PROGRAM:

```

#include <stdio.h>
#include<math.h>
int main()
{
    int n;
    scanf("%d",&n);
    int c[n];
    for(int i=0;i<n;i++){
        scanf("%d",&c[i]);
    }
    int temp = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            if(c[i] < c[j]) {
                temp = c[i];
                c[i] = c[j];
                c[j] = temp;
            }
        }
    }
    int k=0;
    for(int i=0;i<n;i++)
    {
        k+=(pow(n,i)*c[i]);
    }
    printf("%d",k);
}

```

OUTPUT:

	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! ✓

RESULT :

The above program is executed successfully.

QUESTION 3.D

AIM:

Given an array of N integer, we have to maximize the sum of  $arr[i] * i$ , 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

## ALGORITHM:

### Step 1: Start

**Step 2:** Input the integer **n**, the number of elements in array **a**.

**Step 3:** Input **n** integers into array **a**.

**Step 4:** Sort the array **a** in ascending order.

**Step 5:** Initialize **sum** to 0 to store the weighted sum.

**Step 6:** For each element **a[i]**, multiply it by its index **i** and add it to **sum**.

**Step 7:** Print the value of **sum**.

### Step 8: Stop

## PROGRAM:

```
#include<stdio.h>

int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    int temp = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = i+1; j < n; j++)
        {
            if(a[i] > a[j]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    int sum=0;
    for(int i=0;i<n;i++)
    {
        sum+=(a[i]*i);
    }
    printf("%d",sum);
}
```

## OUTPUT:

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

Passed all tests! ✓

## QUESTION 3.E



**AIM:**

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  $\text{SUM } (A[i] * B[i])$  for all `i` is minimum.

**For example:**

Input	Result
3 1 2 3 4 5 6	28

**ALGORITHM :**

**Step 1: Start**

**Step 2:** Input the integer `n`, the number of elements in arrays `a` and `b`.

**Step 3:** Input `n` integers into array `a`.

**Step 4:** Input `n` integers into array `b`.

**Step 5:** Sort array `a` in ascending order.

**Step 6:** Sort array `b` in descending order.

**Step 7:** Initialize `min` to 0 to store the minimum weighted sum.

**Step 8:** For each index `i`, multiply `a[i]` and `b[i]` and add the result to `min`.

**Step 9:** Print the value of `min`.

**Step 10: Stop**

**PROGRAM :**

```

#include<stdio.h>

int main()
{
    int n;
    scanf("%d",&n);
    int a[n],b[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++)
    {
        scanf("%d",&b[i]);
    }
    int temp = 0;
    for (int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    for (int i= 0; i < n; i++)
    {
        for (int j=i+1;j<n;j++)
        {
            if(b[i]<b[j])
            {
                temp=b[i];
                b[i]=b[j];
                b[j]=temp;
            }
        }
    }
    int min=0;
    for(int i=0;i<n;i++)
    {
        min+=(a[i]*b[i]);
    }
    printf("%d",min);
}

```

**OUTPUT:**

	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! ✓

## RESULT:

The above program is executed successfully.