

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Greedy Algorithms](#) / [3-G-Burger Problem](#)

<b>Started on</b>	Tuesday, 27 August 2024, 2:58 PM
<b>State</b>	Finished
<b>Completed on</b>	Tuesday, 27 August 2024, 2:58 PM
<b>Time taken</b>	12 secs
<b>Marks</b>	1.00/1.00
<b>Grade</b>	<b>10.00</b> out of 10.00 ( <b>100%</b> )

## Question 1

Correct

Mark 1.00 out of 1.00

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

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

```

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

```

```
26         return 0;
27     }
28
29     return 0;
30 }
```

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 2-G-Cookies Problem

Jump to...

4-G-Array Sum max problem ▶