CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1 ✓

Finish review

Started on	Friday, 18 October 2024, 1:41 PM
State	Finished
Completed on	Friday, 18 October 2024, 1:41 PM
Time taken	10 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**
Correct
Mark 1.00 out of 1.00
Flag question

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 %)

```c
#include <stdio.h>
#include <math.h>

void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high)
{
    int pivot = arr[low];
    int i = low;
    int j = high;

    while (i < j)
    {
        while (arr[i] > pivot && i <= high - 1) i++;
        while (arr[j] <= pivot && j >= low + 1) j--;
        if (i < j)  swap(&arr[i], &arr[j]);
    }
    swap(&arr[low], &arr[j]);
    return j;
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int partitionIndex = partition(arr, low, high);
        quickSort(arr, low, partitionIndex - 1);
        quickSort(arr, partitionIndex + 1, high);
    }
}

int main()
{
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;++i)
    {
        scanf("%d",&arr[i]);
    }

    quickSort(arr,0,n-1);

    int sum=0;
    for(int i=0;i<n;++i)
    {
        sum+=pow(n,i)*arr[i];
```

	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.

Finish review

◄ 2-G-Cookies Problem

Jump to...

4-G-Array Sum max problem ►