

HackerRank
Prepare
Interview Preparation Kit
Greedy Algorithms
Minimum Absolute Difference in an Array
Exit Full Screen View

Problem
Submissions
Leaderboard

The absolute difference is the positive difference between two values a and b , is written $|a - b|$ or $|b - a|$ and they are equal. If $a = 3$ and $b = 2$, $|3 - 2| = |2 - 3| = 1$. Given an array of integers, find the minimum absolute difference between any two elements in the array.

Example. $arr = [-2, 2, 4]$

There are 3 pairs of numbers: $[-2, 2]$, $[-2, 4]$ and $[2, 4]$. The absolute differences for these pairs are $|(-2) - 2| = 4$, $|(-2) - 4| = 6$ and $|2 - 4| = 2$. The minimum absolute difference is 2.

Function Description

Complete the minimumAbsoluteDifference function in the editor below. It should return an integer that represents the minimum absolute difference between any pair of elements.

minimumAbsoluteDifference has the following parameter(s):

- `int arr[n]`: an array of integers

Returns

- `int`: the minimum absolute difference found

Input Format

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin)

```

1 3
2 3 -7 0

```

Expected Output

```

1 3

```

```
def minimumAbsoluteDifference(arr):
    # Write your code here
    arr.sort()
    return min(abs(arr[i]-arr[i+1]) for i in range(n-1))
```

The code above first sorts the array and then returns the minimum absolute difference between two elements in the array. The code uses a greedy approach by assuming that sorting the array will lead to the smallest absolute difference residing between adjacent elements.

HackerRank
Prepare
Interview Preparation Kit
Greedy Algorithms
Luck Balance
Exit Full Screen View

Problem
Submissions
Leaderboard

Lena is preparing for an important coding competition that is preceded by a number of sequential preliminary contests. Initially, her luck balance is 0. She believes in "saving luck", and wants to check her theory. Each contest is described by two integers, $L[i]$ and $T[i]$:

- $L[i]$ is the amount of luck associated with a contest. If Lena wins the contest, her luck balance will decrease by $L[i]$; if she loses it, her luck balance will increase by $L[i]$.
- $T[i]$ denotes the contest's importance rating. It's equal to 1 if the contest is important, and it's equal to 0 if it's unimportant.

If Lena loses no more than k important contests, what is the maximum amount of luck she can have after competing in all the preliminary contests? This value may be negative.

Example

$k = 2$

$L = [5, 1, 4]$

$T = [1, 1, 0]$

Contest	$L[i]$	$T[i]$
1	5	1
2	1	1
3	4	0

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Input (stdin)

```

1 6 3
2 5 1
3 2 1
4 1 1
5 8 1
6 10 0
7 5 0

```

Expected Output

```

1 29

```

```
def luckBalance(k, contests):
    # Write your code here
```

```

impcontests = sorted([x[0] for x in contests if x[1] == 1])
unimpcontests = [x[0] for x in contests if x[1] == 0]
l = max(0, len(impcontests) - k)
return -sum(impcontests[:l]) + sum(impcontests[l:]) +
sum(unimpcontests)

```

The code above first sorts the array with what are considered the important contests and then calculates the losses and gains from each important contest and the losses and gains of each unimportant contest. Lastly, the algorithm calculates the final luck balance.

HackerRank | Prepare | Interview Preparation Kit | Greedy Algorithms | Greedy Florist

Line: 58 Col: 2

Problem
A group of friends want to buy a bouquet of flowers. The florist wants to maximize his number of new customers and the money he makes. To do this, he decides he'll multiply the price of each flower by the number of that customer's previously purchased flowers plus 1. The first flower will be original price, $(0 + 1) \times \text{original price}$, the next will be $(1 + 1) \times \text{original price}$ and so on.

Given the size of the group of friends, the number of flowers they want to purchase and the original prices of the flowers, determine the minimum cost to purchase all of the flowers. The number of flowers they want equals the length of the c array.

Example
 $c = [1, 2, 3, 4]$
 $k = 3$

The length of $c = 4$, so they want to buy 4 flowers total. Each will buy one of the flowers priced $[2, 3, 4]$ at the original price. Having each purchased $x = 1$ flower, the first flower in the list, $c[0]$, will now cost $(\text{current purchase} + \text{previous purchases}) \times c[0] = (1 + 1) \times 1 = 2$. The total cost is $2 + 3 + 4 + 2 = 11$.

Submissions

Leaderboard

Upload Code as File | Test against custom input | Run Code | Submit Code

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message
Success

Input (stdin)
1 3 3
2 2 5 6

Expected Output
1 13

```

def getMinimumCost(k, c):
    return sum([(i // k + 1) * cost for i, cost in
enumerate(sorted(c, reverse=True))])

```

The code above first sorts the array, calculates each flower's cost, and then sums up the individual flower costs.

- Which problems did you attempt to solve?
 - Minimum Absolute Difference in an Array
 - Luck Balance
 - Greedy Florist
- Which was the easiest problem to solve?
 - The easiest problem to solve was the Minimum Absolute Difference in an Array problem.
- Which was the most difficult problem to solve?

- The most difficult problem to solve among the 3 I did was the Greedy Florist problem.
4. How did your understanding of the greedy paradigm evolve after solving this problem set? How "proficient" would you consider yourself in this paradigm? Did you gain any interesting insights about the paradigm that became apparent after this problem set?
- After solving these problems, my understanding of the greedy paradigm has further increased. One prominent, interesting thing I noticed in my solution to the problems was that I sorted each array before finding the optimal solution to each problem. If I rate my proficiency in this paradigm on a scale from 1 to 10, it would be a 6.