

PA#6, Q2 (median maintenance) Test cases

[Subscribe for email updates.](#)

🔖 No tags yet. [+ Add Tag](#)

Sort replies by: [Oldest first](#) [Newest first](#) [Most popular](#)

[Jon Hoffman](#) · 20 days ago 🔒

Disclosure: I have not turned in the assignment yet, so I don't know if my solution is correct. I used the following two test cases:

Testcase A:

{1 2 4 3 5 6 8 7 9}

Program output:

MEDIANS:

1 1 2 2 3 3 4 4 5

LOW HEAP:

1 2 3 4 5

HIGH HEAP:

6 7 8 9

medians sum= 25

Testcase B:

{11 3 6 9 2 8 4 10 1 12 7 5}

Program output:

MEDIANS:

11 3 6 6 6 6 6 6 6 6

7 6

LOW HEAP:

1 2 3 4 5 6

HIGH HEAP:

7 8 9 10 11 12

medians sum= 75

If somebody notices that either of these test cases is incorrect, please say something!

↑ 14 ↓ · flag

Matteo Nicoli · 20 days ago

your cases are correct.

↑ 0 ↓ · flag



Aluisio Rodrigo Fonseca de Santana · 4 days ago

I got your answers right and the assignment too. But isn't the median (when size of array is odd) calculated as the following: $(\max(\text{low_heap}) + \min(\text{high_heap})) / 2$?

to make it clear here is some prints of TestCase B:

```
low heap = [11] high heap = [] median is = 11#first element inserted
low heap = [3] high heap = [11] median is = 7.0
low heap = [3] high heap = [6, 11] median is = 6
low heap = [6, 3] high heap = [9, 11] median is = 7.5
low heap = [3, 2] high heap = [6, 11, 9] median is = 6
low heap = [6, 2, 3] high heap = [8, 11, 9] median is = 7.0
low heap = [4, 2, 3] high heap = [6, 8, 9, 11] median is = 6
low heap = [6, 4, 3, 2] high heap = [8, 10, 9, 11] median is = 7.0
low heap = [4, 2, 3, 1] high heap = [6, 8, 9, 11, 10] median is = 6
low heap = [6, 4, 3, 1, 2] high heap = [8, 10, 9, 11, 12] median is = 7.0
low heap = [6, 4, 3, 1, 2] high heap = [7, 10, 8, 11, 12, 9] median is = 7
low heap = [6, 4, 5, 1, 2, 3] high heap = [7, 10, 8, 11, 12, 9] median is = 6.5
median sums = 84.0
```

what am I missing?

↑ 0 ↓ · flag



Nima Farnoodian · 4 days ago

Hi Aluisio,

I think you are wrong with the definition of what the median is. What you compute for some elements at each time is indeed the average of the elements. When the array size is even, the median is $n/2$ th smallest value of the array, otherwise $(n+1)/2$ th smallest element. For example, when you have just [3,11] as input array, the median is 3 because $n=2$, so we should look for the first smallest element. Another example is to consider input array when elements are [11 3 6 9 2]. In this case the median is 6 because if we sort elements, we get [2 3 6 9 11]. Then the $(n+1)/2$ th smallest element is 6 where n is 5 and odd. In general, to find the Median, place the numbers in **value order** and find the middle. But in the case of using two heaps (Hlow and Hhigh, as described in Lect. 5.6), if the size of array (size of Hlow + size of Hhigh) is even, then the median is the maximum value extracted from Hlow, but if (size of Hlow + size of Hhigh) is odd, two conditions appear as follow:

If (size of Hlow = size of Hhigh - 1), then the median is the minimum value extracted from Hhigh.

If (size of Hlow - 1 = size of Hhigh), then the median is the maximum value extracted from Hlow.

Note that you should be careful about how to put numbers into these two heaps. By this I

mean your algorithm must make a balance between the sizes of these two heaps. In other word, the difference between the size of Hhigh and the size of Hlow has to be at most one. Formally $|\text{Size of Hlow} - \text{Size of Hhigh}| < 2$ where $|\cdot|$ means **Absolute Value**.
I hope it would help you, Good Luck!

↑ 0 ↓ · flag

[+ Comment](#)



Vladimir Lyutin · 20 days ago

Test cases got right. Assignment got wrong. All seems fine except the result. Maybe I don't get something. We just sum the medians (10,000 of them) and then compute modulo 1000? Has someone got the result right?

↑ 0 ↓ · flag

Miroslav Dimitrov · 19 days ago

I got the test cases right too, but assignment answer was wrong.

Because the answer is not correct and it should not be considered as spoiler, is by any chance you receive this answer before modulus operation (4470****)? (last 4 digits are omitted)

↑ 0 ↓ · flag



Vladimir Lyutin · 19 days ago

No, I got 4683****. Seems we've made different mistakes.

↑ 0 ↓ · flag

Miroslav Dimitrov · 19 days ago

Found the error in my code, all OK with the final answer!

↑ 0 ↓ · flag

Miroslav Dimitrov · 19 days ago

Hm, maybe you are calculating modulo 1000, when you should calculate **modulo 10000**?

↑ 0 ↓ · flag

Jon Hoffman · 19 days ago

@Vladimir,

That's not good! I have 4683**** too!

(I have not submitted yet)

↑ 0 ↓ · flag

Miroslav Dimitrov · 19 days ago

I confirm that the final answer before modulus is in the form: **4683******

↑ 4 ↓ · flag



Vladimir Lyutin · 19 days ago

Yes, my mistake was **mod 1000** instead of **mod 10000**. Thank You very much!

↑ 1 ↓ · flag



Benedek Kiss · 8 days ago

Don't forget to carry on only the modulo to avoid integer overflow during computation. Although it looks like the numbers were small enough here to allow such laziness. ;-)

↑ 0 ↓ · flag

[+ Comment](#)

Cindy Van Dooren · 19 days ago

Is the second test case really correct or am I missing something?

I would say that the medians look like this:

11
7
6
7.5
6
7
6
7
6
7
7
6.5

I assume we would round them down to the previous integer (as in integer division, but I'm not sure if that's what we're supposed to do). In my opinion, the fourth median should be calculated as follows:

- order the numbers: 3 6 9 11,
- since it's an even number, take the average of the middle two: $(6 + 9) / 2 = 15 / 2 = 7.5$

Am I having a totally different understanding of median?

↑ 2 ↓ · flag

Matteo Nicoli · 19 days ago

yes, you are using a different definition of the median. For the exercise is 6, not the average

between 6 and 9.

↑ 2 ↓ · flag

Cindy Van Dooren · 19 days ago

I see now, thanks!

↑ 0 ↓ · flag

[+ Comment](#)

Thom Strom · 14 days ago

Thanks, Jon, for posting the A & B test cases. Does anyone have more test-cases to practice with?

For my implementation (in Go), the A & B test cases pass, but I'm getting an incorrect pre-modulo result for the homework assignment: 2490**** (~49% of the total of all entries in the file). I'm really confused, since others appear to be getting a pre-modulo median (4683****), which is ~93% of the total entries within the file.

I've confirmed that both the high & low heaps have 5000 elements each, so I'm at least hopefully reading all of the elements in from the file properly. I also set the total type to an int64 to avoid any strange overflows, but the result did not change.

As a second test case, I took the first 100 entries from Median.txt, and my result is 238538 (56% of the total of all entries in the file).

↑ 0 ↓ · flag

[+ Comment](#)



Nima Farnoodian · 10 days ago

Hi friend,

my result is 2500**** before modulus. My algorithm works well for some test data, but according to what you wrote above, my result is not right. Could someone help me solve this problem. Note that the technique I used is that I initially sort the array and then sum all medians for each xk positioned at right place.

↑ 0 ↓ · flag



Jingwen Ouyang · 9 days ago

my suggestion is that you can add the medians as you go, including the modulation. then you won't have a big array of medians to add in the end (memory issue, and overflow issue). good luck!

↑ 0 ↓ · flag

Sudhanshu Kasewa · 9 days ago

Hi Nima, I don't think your implementation is correct. The idea is to store and sum the medians of the numbers in the stream as they come in (so, the median M_k is of the first k numbers to be seen). You seem to say that once you have loaded all the numbers, you sort them, and then take the medians, (so each median M_k is the median of the smallest k numbers)

These two approaches are different and will yield different answers.

↑ 0 ↓ · flag



Nima Farnoodian · 9 days ago 🔗

Many thanks to [Sudhanshu Kasewa](#) and [Jingwen Ouyang](#) . I found out where I made a mistake. I first did not get what exactly we are asked to do because I didn't watch lecture 5.6. When I read some texts posted here, I realized that I need to see the lecture 5.6. After that I understood what the problem is and how I can solve it using two heaps. But I still have a big problem with Q1 I coded up my algorithm in Matlab R2010a and it seems to work well for test cases. But unfortunately, It can't give me the summation result for original dataset and intervals. In other word, It takes too long to compute so that I decide to end computation process. Could someone please tell me what I can do to get this issue around . Is this probably because of Matlab?

↑ 0 ↓ · flag

Matt Larson · 9 days ago 🔗

Nima--regarding your problem with Q1, maybe you are hitting the problem where the algorithm described in the lectures just takes a long time to run for larger sizes of t and n (input size)? With a straight implementation I think you get $O(tn)$, which is 20 billion operations in our case. I hit this problem and my solution took about 80 mins to run. So in a real world implementation this would need some improvements, and there is an excellent discussion on the forums with various solutions involving sorting or buckets that can really speed things up if this is the issue that you are having:

https://class.coursera.org/algo-009/forum/thread?thread_id=290

↑ 0 ↓ · flag



Nima Farnoodian · 8 days ago 🔗





Matt Larson--Thank you, I'll check what the other fellow students did to make algorithm faster.

↑ 0 ↓ · flag

[+ Comment](#)

New post

To ensure a positive and productive discussion, please read our [forum posting policies](#) before posting.

B	<i>I</i>			 Link	<code><code></code>	 Pic	Math	Edit: Rich ▾		Preview
<div></div>										

☐☒

Add post