PA4: Median Finding

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

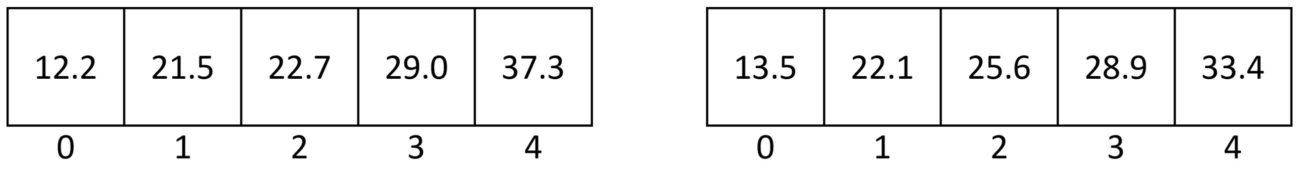
The Cola Wars are still raging, with Coca-Cola and Pepsi competing to be at the top of the global brown fizzy liquid market. While measuring rates of inflation the U.S. Department of the Treasury wants to know whether rates are higher for consumers or corporations. To answer this question the Treasury wants to know the median price that soda manufacturers pay for sugar. Since Coca-Cola and Pepsi are so competitive, they're afraid to give away the details of too many of their contracts. For this assignment we will write an algorithm to determine the median price that Coca-Cola and Pepsi pay for sugar with only a limited number of "requests" for amounts paid from each company.

Problem Statement

We define the median of a collection is values to be a value for which the number of elements less than that value matches the number of elements greater than that value. For example, the median of the list

"1,2,3,4,5" would be 3, and the median of the list "1,2,3,4,5,6" would be 3.5 (or from our definition, any value strictly between 3 and 4).

Given two sorted lists of n doubles/floats each (which will be unique within each list, but may not be between lists), you will write a divide and conquer algorithm which efficiently finds the median of both lists combined. Specifically, you will write an algorithm which can find the median of the union of the two lists using no more than $2 \cdot \lceil \log_2(n) \rceil$ total accesses into each lists. For example, if given the pair of length-4 lists below, your algorithm should return a value strictly between 22.7 and 25.6 (for example, 24.15).



Starter Code Summary The starter code provided consists of two files:

- **Company.java / company.py**: This is implements an object to represent each company and the amounts they're paying for sugar. This is also the file which contains main, so run your program from this file. There are two ways of constructing a company object. Option 1: provide a number indicating the length of the list of values, then the constructor will populate this list with random values. Option 2: provide your own list for the list of values (for python you must also provide the size of that list). Once you have the company objects, these are the important fields and methods you may want to use. You will not submit this file (the autograder has its own version that it will use).
  - `request(location)`: This is the method you will use to access the value at a particular index. The argument is the index you would like to see, the output is a float(python)/double(java). For each company object you are limited to $2 \cdot \lceil \log_2(n) \rceil$ calls to this method. If you exceed maximum allotted ping count then the method will print a statement indicating this, and then only return -1. This is the only method from this class that should be present in your final submission.
  - `size`: This attribute represents the size of the list of values. You're welcome to access this as well in the code you submit.
  - `printall`: This method prints the entire list of values for that company, it may be useful for debugging
  - `toString` or `str`: in Java and Python (respectively) these methods convert the object to a string. In the starter code invoking these will result in a string where only the values previously queried can be seen. All other values will appear as "??".
  - `slice`: This will return just a piece of the list of values. The arguments are two integers to represent a start index and an end index. The output will be the portion of the list of values between the start (inclusive) and the end (exclusive) given. This may be useful for debugging. Your final submission should not use this method, however.
- **MedianFinder.java / medianfinder.py**: *This is the file you will modify and submit.* In here there is the header of just one method: `find_median(c1, c2)`. You should implement this function to be a divide and conquer algorithm to find the median of the list of values for companies `c1` and `c2`. Your algorithm should return the value of the median. Your algorithm must also follow the divide and conquer paradigm. In particular, it must include a conquer step consisting of recursive calls on one or more subproblems.

Input You algorithm will be given two company objects as input.

Output The function you implement will return a single float/double representing the median of the values contained within the two companies.

Running Time Your algorithm must find the median using no more than $2 \cdot \lceil \log_2(n) \rceil$ requests to each company (so a maximum of $4 \cdot \lceil \log_2(n) \rceil$ requests total).

Testing No test cases are provided for this assignment. Instead, the `company` class constructor will create random collections of numbers for you. Since there is also an option to construct a company object using your own list, I recommend that you also come up with some of your own test cases to make sure you haven't missed anything!

Helpful Hints To help you to get started with thinking about this problem, keep these in mind:

1. Since each company has n items in its list, there are $2n$ items total. With an even number of items the median will be between two values.
2. The two values which "bookend" the median are not necessarily going to be in different lists
3. Convince yourself that this is true: for any list of more than 2 elements, removing the smallest and largest element will not change the median of the list.
4. The autograder's version of the company class will use different names for methods/attributes that your program should not use. This means if you're getting a runtime error you should double check you don't have any lingering calls to anything besides request and size.