CSSE 220: Hardy's Taxi Programming Problem

Carefully follow the turnin instructions at the bottom of this document. This is an Pair Programming assignment. The expectation is that you and your partner will do all of the work for this assignment in Pair Programming mode, frequently switching drivers. Be sure to add comments that explain your approach. Javadoc comments are not necessary.

This program will not take a lot of code, (probably less than 200 lines), but it will take a lot of thought, especially thought about how to do it efficiently. Efficiency isa major goal for this program. In what order should you try various sums of cubes? What information should you store? How do you avoid duplicates? How do you make sure you don't miss any solutions?

The following was adapted from http://cs.bilgi.edu.tr/pages/curiosity_corner/challenges/ramanujans_number.html

Ramanujan and Hardy's taxi

G H Hardy, the famous British mathematician, brought Ramanujan, a poor man from India who was a natural mathematical genius, to work with him at Cambridge University. They had a very important and creative mathematical partnership. The British climate was bad for Ramanujan. He got tuberculosis. As he lay dying in hospital, Hardy went to visit him. As he entered the room he said, "The number of my taxi was 1729. It seemed to me rather a dull number." To which Ramanujan replied, "No, Hardy! No, Hardy! It is a very interesting number. It is the smallest number expressible as the sum of two cubes in two different ways."

Ramanujan died at the age of 33.

Is the number in the story correct? Is 1729 the smallest number expressible as the sum of two cubes in two different ways? It is fairly easy to show it is expressible as the sum of two cubes in two different ways, but is it the **smallest** such number?

Obviously we can think of some other questions:

If 1729 is the smallest such number, what is the second smallest? the n^{th} smallest? Can we write a program to find this?

What do we mean by "different ways"?

What about a more general statement of the problem:
What is the k^{th} smallest number that can be expressed as the sum of the nth powers of m numbers in r different ways? Can we find a formula for it? (I doubt it) Can we write a program to do it? Are the results interesting?

The programming isn't as easy as it might at first seem. Some of these numbers are going to be rather big!

Chris Stephenson

Write a Java method (`Hardy.nthHardyNumber()`) whose parameter is a positive integer N that then finds and prints (to as a `HardySolution` object the Nth smallest positive integer that can be expressed as the sum of two cubes (of positive integers) in two or more different ways, along with the evidence that it can be expressed that way.

The **HardysTaxi** project in your hardy repository has a framework that you are to use to calculate and report solutions, along with a few JUnit tests. I may provide more tests later.

Examples of solutions (these are the answers for N=1 and N=4):

```
1729 = 1^3 + 12^3 = 9^3 + 10^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
```

Of course any given solution could be stored in a few different orders. To make it unique, yours must order the numbers in a solution so that $a_1 <= b_1$, $a_2 <= b_2$, and $a_1 < a_2$, as in the above examples.

Efficiency: You should see for how large a value of N your program will run in a reasonable amount of time without running out of memory (using Java's standard memory allocation). N=30 should be reasonably easy. Can you do 100? 500? 1000? 2000? 5000? At what value of N do you begin to see a noticeable slowdown in the program's running time?

Note: In-between calls to `nthHardyNumber`, you are not allowed to save the Hardy numbers that you have already discovered. You must "Start from scratch" with each call, so we get a true test of the timing for computing the nth Hardy number. I.e., re-initialize all arrays, array lists, or other collections before each call to `nthHardyNumber`.

Note: You are not allowed to write the program in a way that assumes a fixed limit on the sizes of the numbers *a* and *b* that it must check. Theoretically (if it has enough time and memory), your program should be able to calculate `nthHardyNumber` for any sized n that does not make the sum of cubes be larger than the largest `long` value.

Turnin instructions

Submit it to your Hardy SVN repository.

Grading Winter 2008-09

On Tuesday, January 20, I provided (*via* email) a new set of JUnit tests that include point values. When I test your code for grading purposes, I may change some of the numbers, but the tests will be of the same types as these, and the point values will be calculated similarly.

Extras

For fun (but not to turn in), you might think about how to tackle the more general problem posed in the box on the previous page. And perhaps even code it (but do not call that code from `main()` in the `Hardy` class.)