

CSSE 220: Hardy's Taxi Programming Problem

Carefully follow the turnin instructions at the bottom of this document. This is an individual assignment. As usual, you may talk to others, but your code should be your own. 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. 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 n th 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

Read the story of Ramanujan and Hardy in [Curiosity Corner](#)

Write a Java application (Hardy.java) that reads a positive integer N from standard input (your program should not print a prompt, or anything else except the solution in the form prescribed below) and then finds and prints (to standard output) the N th 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.

Your program must print the solution in exactly the following format (including exactly one space before and after each + and =):

$$C = a_1^3 + b_1^3 = a_2^3 + b_2^3$$

Examples of this output format (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 printed in a few different orders. To make it unique, your output should order the numbers in a solution so that $a_1 \leq b_1$, $a_2 \leq b_2$, and $a_1 < a_2$, as in the above examples.

Your program should only print this one line (use a **println** so that that line ends with an end-of-line character) and nothing else!

Hardy should be the name of your class that contains `main()`, and it should be defined in the file `Hardy.java`. If you define any auxiliary classes, they may be defined in the same Java source file or a different file.

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?

Turnin instructions:

Submit it to your personal SVN repository.

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.)

Grading Spring 2008

Each program will be run for 1 minute on my computer. The following test cases will be used, giving a total of 70 pts:

N	Points	Total score if it works for all $n \leq N$
1	20	20/70
5	15	35/70
20	15	50/70 (C)
50	10	60/70 (B)
100	4	64/70 (A-)
200	3	67/70 (A)
500	3	70/70 (A+)