Semiperfect numbers

From Wikipedia, the free encyclopedia:

In number theory, a semiperfect number or pseudoperfect number is a positive integer or a natural number n that is equal to the sum of all **or** some of its proper divisors. (A proper divisor of a number is a divisor less than the number.) A semiperfect number that is equal to the sum of all its proper divisors is a perfect number. A few interesting facts about semiperfect (which have nothing to do with this problem numbers) are:

* The first few semiperfect numbers are: 6, 12, 18, 20, 24, 28, 30, 36, 40, ...
* Every multiple of a semiperfect number is semiperfect.
* A semiperfect number that is not divisible by any smaller semiperfect number is primitive.
* The smallest odd semiperfect number is 945 (see, e.g., Friedman 1993).
* A semiperfect number is necessarily either perfect or abundant.
* An abundant number that is not semiperfect is called a weird number.

In this problem you will create two static methods.

The first method you will create is the getDivisors(num) method returns an array of all divisors of num less than num, sorted in ascending order. You may assume num > 1.

The following code shows the results of the getDivisors method.

|  |  |
| --- | --- |
| The following code | Returns |
| int[] divs = SemiPerfect.getDivisors(6); |  |
| divs.length; | 3 |
| divs[0]; | 1 |
| divs[1]; | 2 |
| divs[2]; | 3 |

The second method you will create is the isSemiPerfect(num) method, which returns true if num is a semiperfect number and returns false if num is a **NOT** semiperfect number. You may assume num > 1.

The following code shows the results of the isSemiPerfect method.

|  |  |
| --- | --- |
| The following code | Returns |
| SemiPerfect.isSemiPerfect(6)); | true |
| SemiPerfect.isSemiPerfect(96)); | true |
| SemiPerfect.isSemiPerfect(106)); | false |
| SemiPerfect.isSemiPerfect(945)); | true |