Code Jam 2019 - World Finals

Analysis: Won't sum? Must now

The fact that we never need more than 3 palindromes is not obvious, and it was proved in a long, complicated paper. Numberphile made a video that explains the basics of the paper's argument. Knowing this upper bound is useful, but is not necessary to solve the problem.

Even if we haven't read every paper on arXiv for fun, we might also reasonably guess that the number of palindromes required is unlikely to be large because there are approximately sqrt(**S**) palindromes smaller than **S**, which means that there are approximately **S** palindrome pairs, which should result in Θ(**S**) numbers smaller than **S** that should be representable as sums of palindrome pairs. The number of triples is larger by another factor of sqrt(**S**), and their sums are all smaller than 3**S**, which does not leave much space to hide a number that could not be represented by a triple of palindromes, unless many triples add up to the same number.

The first step is to check if **S** is itself a palindrome. If it is a palindrome, then we are done. In the remaining explanation, we will assume that **S** is not a palindrome. Checking if a number is a palindrome takes O(log **S**) steps, one step per digit.

Test set 1

Key insight: There are only roughly 2×10^5 palindromic terms up to 10^{10}. In the analysis below, we will call this number P.

Sum of 2 palindromes?

To check if **S** is the sum of 2 palindromes, we simply loop through all palindromes, p, less than **S** and check if **S**-p is a palindrome. This takes O(P × log **S**) steps.

Sum of 3 palindromes!

We were told that every number is the sum of 3 palindromes, so we must find such a sum! On first glance, it seems that doing the same as above for 3 palindromes is too slow since there are $O(P^2)$ pairs of palindromes to check. However, for all numbers up to 10^{10}, one of the 3 palindromic terms is always at most 404 (which is needed for 1086412253, for example). Thus, as long as we search the pairs of palindromes in a way that searches triples containing a small value first, we will be fine.

Test set 2

Unfortunately, there are as many as 2×10^{20} palindromes less than 10^{40}, so it is hopeless to even iterate through the palindromes. Something quicker will be needed.

S = X + Y?

We will assume X ≥ Y. The first step is to fix the numbers of digits in X and Y. Note that the number of digits in X must be either (1) the number of digits in **S** or (2) one less than the number of digits in **S** (otherwise, X + Y < **S**).

Let's assume that X and Y have a different number of digits (we will deal with the same-number-of-digits case below). The trickiest part about this problem is dealing with the pesky carries that occur while adding. For example, if we want to find 2 palindromes that add up to 2718281828 with lengths 10 and 8, then the first digit of X can be either 1 or 2 (depending whether there is a carry on the most significant digit). Because we don't know, let's try both! First, let's assume that we do not need a carry (so the first digit of X is 2). This also means that we can determine the least significant digit of Y by considering the sum modulo 10 (which also fills in the most significant digit of Y since it is a palindrome).

```
  ..........        2........2        2........2
+   ........  ->  +   ........  ->  +   6......6
============      ============      ============
  2718281828        2718281828        2718281828
```

At this point, to fill in the remaining unknown digits, we need a 9 digit palindrome and a 7 digit palindrome that sum to 2718281828 - (2000000002 + 60000006) = 658281820. This is a subproblem and we may recurse (with leading zeroes now allowed). The other option is that there is a carry on the most significant digit. In this case, the first digit of X must be 1.

When the numbers of digits in X and Y are different, there is always at least one unknown digit that we can determine. However, if X and Y have the same number of digits, this is not necessarily true. However, we can combine two simple observations to determine viable options for the first (and last) digits of both numbers. First, we consider the sums modulo 10. This narrows the possible values we may choose. Second, the number of digits in **S** compared to the number of digits in X (and Y) tells us if we want the sum of the most significant digits to be at least 10 (causing a carry). For example, if **S** has 5 digits, while X and Y only have 4 digits, then we must provide a carry in order for this sum to work. Similarly, if the number of digits in **S** and X are equal, then we cannot have a carry, so the sum must be at most 9. Even after these constraints, we still have some options (for example, if we need a sum of 8, we can use 0+8, 1+7, ... , 7+1, 8+0). Note that it doesn't matter* which of these options we take, since the only way they interact with other digits is with the carry.

* - There is one case where it matters: we must avoid creating a leading zero in a number.

Let D be the number of digits in **S**. There are O(D) possible lengths for X and Y. For each pair of lengths, we must decide if there is a carry on each digit on the left-hand side of **S**. Depending on implementation, this is O($2^{D/2}$ × D) operations. This leads to a total of O($2^{D/2}$ × D^2) operations. Note that instead of just computing one digit at a time, we can determine the whole "overhang" values. In the example above, we can determine that the first two digits are either 26 or 27:

```
  ..........        27......72        27......72
+   ........  ->  +   ........  ->  +   65....56
============      ============      ============
  2718281828        2718281828        2718281828
```

This means that we only need to make D/overhang choices for the carries instead of D/2 choices. Thus, when the overhang is large, those computations do not contribute heavily to the complexity. This reduces the total complexity to O($2^{D/2}$ × D) operations.

Sum of 3 palindromes

In the paper listed above, the authors describe an algorithm to write any number as the sum of 3 palindromes is described. This was obviously not needed to solve this problem, but it does require an algorithm to find the sum of three palindromes.

To solve this problem, we will use the fact that there are O(**S** sqrt(**S**)) triples of palindromes less than **S**, which means that each number can be written with an average of O(sqrt(**S**)) different triples. This allows us to randomly select palindromes as one of our three values, then use the algorithm above to check whether the remaining value can be wrritten as the sum of 2 palindromes.

It is possible that there is a number that can only be represented in one different way as a sum of three palindromes, but we were unable to find any case that was not the sum of triples in many different ways. In fact, we were unable to find an input number such that one of the three numbers was more than 10801 (though, we expect that many such cases exist).