Q1: What if you were to cache the calculation, for example in the file system. What would an example implementation of the cache look like? By cache I mean, given an array input, skip the calculation of the output if you have already calculated the output at least once already.

A: My implementation of cache is using a hash table storing each element in the array and the array contains all the factors of this element. In my hash table implementation, I store the previous calculation results. When I find a specific factor A, and A has its own factors[a1,a2,…] which I had calculated before. Then I will skip the calculation and add all the factors of A without any other calculations. I also sort the input array in ascending order first, as a result, when I search for the factors of current element, I only need to search the part of array on the left of current element. This would improve the search algorithm from O(N) to O(K) and avoid some unnecessary calculations.

Q2: What is the performance of your caching implementation? Is there any way to make it more performant.

A: Because I am using a hash table implementation, so the search operation would give a O(1) time complexity. In my program, I iterate the array and for each i in the array I search from the beginning to its left, however, when I have calculated one element, I skip the calculation and return the former calculation result. So this would make the algorithm more performant. The more elements in the array are factors of other elements in the array, the more performant this algorithm will be. In the ideal case, it can be approximately O(N) time complexity. For the general conditions, the time complexity for this algorithm is O(N\*K). For searching, it would be O(1).

Because I am using a hash table to store the mapping message, the space complexity is O(N) in my program.

Q3: What if you wanted to reverse the functionality. What if you wanted to output each integer and all the other integers in the array that is the first integer is a factor of I.E:

Would this change your caching algorithm? With caching, the output should be the same except the calculation are not performed.

A: My caching method will not be changed. I think the only difference is that I need to sort the array in descending order, and still search for the left part of the array. And when I do computation, I check if i is multiples of current element instead of i is the factor of current element. The overall solution is the same. Specific code can be seen at factor\_cache\_reverse.rb file.