In Molecular Genetics, there is a notion of an Open Reading Frame (ORF). An ORF is a portion of DNA that is used as the blueprint for a protein. All ORFs start with a particular sequence, and end with a particular sequence.

In this task, we wish to find all sections of a genome which start with a given sequence of characters, and end with a (possibly) different given sequence of characters.

To solve this problem, you will need to create a class `OrfFinder`. The constructor for this class takes a string `genome` as a parameter. Additionally, this class will need a method `find(start, end)`.

1.1 Input

`genome` is a single non-empty string consisting only of uppercase [A-D]. `genome` is passed as an arguement to the `__init__` method of `OrfFinder` (i.e. it gets used when creating an instance of the class).

`start` and `end` are each a single non-empty string consisting of only uppercase [A-D].

1.2 Output

`find` returns a list of strings. This list contains all the substrings of `genome` which have `start` as a prefix and `end` as a suffix. There is no particular requirement for the order of these strings. `start` and `end` must not overlap in the substring (see the last two cases of the example below).

1.3 Example

```
genome1 = OrfFinder("AAABBBCCC")
genome1.find("AAA","BB")
>>> ["AAABB","AAABBB"]
genome1.find("BB","A")
>>>[]
genome1.find("AA","BC")
>>>["AABBBC","AAABBBC"]
genome1.find("A","B")
>>> ["AAAB","AAABB","AAABBB","AAB","AABB","AABBB","AB","ABB","ABBB"]
genome1.find("AA","A")
>>> ["AAA"]
#note that "AA" is not valid, since start and end would need to overlap
genome1.find("AAAB","BBB")
>>> []
# note that "AAABBB" is not valid, since start and end would need to overlap
```

1.4 Complexity Requirements

- The `__init__` method of `OrfFinder` must run in time complexity $O(N^2)$, where N is the length of `genome`.
- Let T be the length of the string `start`, U be the length of the string `end`, and V be the number of characters in the output list (for a correctly generated output list according to the instructions in 1.2), then `find` must run in time complexity $(T + U + V)$.

As an example of what the complexity for `find` means, consider a string consisting of $\frac{N}{2}$ "B"s followed by $\frac{N}{2}$ "A"s. If we call `find("A","B")`, the output is empty, so V is $O(1)$. On the other hand, if we call `find("B", "A")` then V is $O(N^2)$.