Not Another Committee!

If there's one thing that universities love to do, it's to create committees. Need to design a new degree? Have a committee figure it out! Dealing with a problem? A committee will sort it out! However, given the large number of committees, the university has realized it needs software to manage the composition of these committees.

More specifically, a committee is composed of several professors, each of which can belong to one or more fields (e.g., a professor could do research in both Computer Science and Mathematics). Furthermore, professors can either be tenured or not. In this problem, we will determine if a **bad** committee has been formed. A committee is *bad* if both of these conditions are true:

- There are only one or two unique fields represented in the committee.
- There are not enough tenured professors in the committee. If the committee has N professors, we need at least $(N//2) + 1$ of them to be tenured. The operator $(//)$ means floor division (i.e., dividing and rounding down to the nearest integer.)

Here are some sample professors:

Name	Tenured (T/F)	Areas
bob	T	systems theory
sally	T	theory
ally	T	ai
rob	F	systems
ben	F	systems
jenny	F	systems ai

Here are some sample committees:

Name	Members	Bad? (T/F)
c1	bob sally ally	F
c2	bob rob jenny stephanie	T
c7	sally ally	F

We have supplied a class for representing professors (see below for the implementation of the `Professor` class). Your job is to complete the `Committee` class.

You are required to implement:

- the `Committee` constructor

- `addMember` - takes a `Professor` as an argument and add the professor, if they are not already on the committee. Returns `true` is the professor was successfully added to the committee and `false` otherwise. You may assume the professors have unique names.
- `numMembers` - returns the number of committee members
- `numTenured` - returns the number of tenured professor on the committee
- `getUniqueFields` - returns a set of the fields represented by the professors on the committee
- `isBadCommittee` - returns `true` if the committee is bad as defined above and `false` otherwise.

You are are welcome to write additional helper methods and to include additional attributes.

Here is the code for the `Professor` class. For this practice problem, you can find it in the file named `professor.py`. For the actual exam, you would be expected to copy it into a file named `professor.py`.

```
'''
Distribution file for the committees problem containing the Professor class.

Your solution should be placed in committees.py, not this file.
'''

from typing import List, Set

class Professor:
    '''
    Class encapsulating information about a professor.

    Public attributes:
        -   name (str): The name of the professor.
        -   tenured (bool): Whether the professor is tenured.
        -   fields (Set[str]): The fields the professor works in.
    '''
    def __init__(self, name: str, tenured: bool, fields: List[str]) -> None:
        '''
        Constructor for Professor class.

        Parameters:
            -   name (str): The name of the professor.
            -   tenured (bool): Whether the professor is tenured.
            -   fields (List[str]): The fields the professor works in.

        Returns:
            None
        '''
```

```
        self.name = name
        self.tenured = tenured
        self.fields = set(fields)
```

Here is the skeleton code for the `Committee` class. For this practice problem, you can find it in the file named `committee.py`. For the actual exam, you would be expected to copy it into a file named `committee.py`.

```
'''
Distribution file for the committees problem containing the Committee class.

Please implement your solution in this file by filling in the TODOs.
'''

from typing import List, Set
from professor import Professor

class Committee:
    '''
    Class encapsulating information about a committee.

    Public attributes:
        -   name (str): The name of the committee.
        -   members (List[Professor]): The professors on the committee.

    Public methods:
        -   add_member: Add a professor to the committee.
        -   is_bad_committee: Check if the committee is invalid according to
            the criteria in the problem statement.
    '''
    def __init__(self, name: str) -> None:
        '''
        Constructor for Committee class.

        Parameters:
            -   name (str): The name of the committee.

        Returns:
            None
        '''
        # TODO: Implement this method
        pass

    def add_member(self, professor: Professor) -> bool:
        '''
        Add a professor to the committee if the professor is not already on the
        committee.
```

```
    Parameters:
        -   professor (Professor): The professor to add.

    Returns:
        True if the professor was added, False otherwise.
    '''
    # TODO: Implement this method
    pass

def __num_member(self) -> int:
    '''
    Return the number of members on this committee.

    Parameters:
        None

    Returns:
        The number of members on this committee.
    '''
    # TODO: Implement this method
    pass

def __num_tenured(self) -> int:
    '''
    Return the number of tenured professors on this committee.

    Parameters:
        None

    Returns:
        The number of tenured professors on this committee.
    '''
    # TODO: Implement this method
    pass

def __get_unique_fields(self) -> Set[str]:
    '''
    Return the unique fields of the professors on this committee.

    Parameters:
        None

    Returns:
        The unique fields of the professors on this committee.
    '''
```

```
    # TODO: Implement this method
    pass

def is_bad_committee(self) -> bool:
    '''
    Check if this committee is bad according to the criteria in the
    problem statement.

    Parameters:
        None

    Returns:
        True if this committee is invalid, False otherwise.
    '''
    # TODO: Implement this method
    pass
```