Output Contest Matches

During the NBA playoffs, we always arrange the rather strong team to play with the rather weak team, like make the rank 1 team play with the rank n_{th} team, which is a good strategy to make the contest more interesting. Now, you're given **n** teams, you need to output their **final** contest matches in the form of a string.

The **n** teams are given in the form of positive integers from 1 to n, which represents their initial rank. (Rank 1 is the strongest team and Rank n is the weakest team.) We'll use parentheses('(', ')') and commas(',') to represent the contest team pairing - parentheses('(' , ')') for pairing and commas(',') for partition. During the pairing process in each round, you always need to follow the strategy of making the rather strong one pair with the rather weak one.

Example 1:

```
Input: 2
Output: (1,2)
Explanation:
Initially, we have the team 1 and the team 2, placed like: 1,2.
Then we pair the team (1,2) together with '(', ')' and ',', which is the final answ
er.
```

Example 2:

```
Input: 4
Output: ((1,4),(2,3))
Explanation:
In the first round, we pair the team 1 and 4, the team 2 and 3 together, as we need
 to make the strong team and weak team together.
And we got (1,4),(2,3).
In the second round, the winners of (1,4) and (2,3) need to play again to generate
the final winner, so you need to add the paratheses outside them.
And we got the final answer ((1,4),(2,3)).
```

Example 3:

```
Input: 8
Output: (((1,8),(4,5)),((2,7),(3,6)))
Explanation:
First round: (1,8),(2,7),(3,6),(4,5)
Second round: ((1,8),(4,5)),((2,7),(3,6))
Third round: (((1,8),(4,5)),((2,7),(3,6)))
Since the third round will generate the final winner, you need to output the answer
 (((1,8),(4,5)),((2,7),(3,6))).
```

Note:

1. The **n** is in range [2, 2^{12}].
2. We ensure that the input **n** can be converted into the form 2^k, where k is a positive integer.

Solution 1

```
public class Solution {
    public String findContestMatch(int n) {
        List<String> matches = new ArrayList<>();
        for(int i = 1; i <= n; i++) matches.add(String.valueOf(i));

        while(matches.size() != 1){
            List<String> newRound = new ArrayList<>();
            for(int i = 0; i < matches.size()/2; i++)
                newRound.add("(" + matches.get(i) + "," + matches.get(matches.siz
e() - i - 1) + ")");
            matches = newRound;
        }
        return matches.get(0);
    }
}
```

written by fallcreek original link here

Solution 2

Pretty straight forward one, just keep going until `n == 1`.

```
class Solution(object):
    def findContestMatch(self, n):
        """
        :type n: int
        :rtype: str
        """
        res = map(str, range(1, n+1))
        while n > 1:
            res = ["(" + res[i] + "," + res[n-1-i] + ")" for i in range(n >> 1)]
            n >>= 1
        return res[0]
```

written by realisking original link here

Solution 3

C++

```cpp
class Solution {
public:
    string findContestMatch(int n) {
        vector<string> m(n);
        for (int i = 0; i < n; i++) {
            m[i] = to_string(i + 1);
        }

        while (n > 1) {
            for (int i = 0; i < n / 2; i++) {
                m[i] = "(" + m[i] + "," + m[n - 1 - i] + ")";
            }
            n /= 2;
        }

        return m[0];
    }
};
```

Java

```java
public class Solution {
    public String findContestMatch(int n) {
        String[] m = new String[n];
        for (int i = 0; i < n; i++) {
            m[i] = String.valueOf(i + 1);
        }

        while (n > 1) {
            for (int i = 0; i < n / 2; i++) {
                m[i] = "(" + m[i] + "," + m[n - 1 - i] + ")";
            }
            n /= 2;
        }

        return m[0];
    }
}
```

written by alexander original link here