Factor Combinations

Numbers can be regarded as product of its factors. For example,

```
8 = 2 x 2 x 2;
  = 2 x 4.
```

Write a function that takes an integer *n* and return all possible combinations of its factors.

Note:

1. Each combination's factors must be sorted ascending, for example: The factors of 2 and 6 is `[2, 6]`, not `[6, 2]`.
2. You may assume that *n* is always positive.
3. Factors should be greater than 1 and less than *n*.

Examples:
input: `1`
output:

```
[]
```

input: `37`
output:

```
[]
```

input: `12`
output:

```
[
  [2, 6],
  [2, 2, 3],
  [3, 4]
]
```

input: `32`
output:

```
[
  [2, 16],
  [2, 2, 8],
  [2, 2, 2, 4],
  [2, 2, 2, 2, 2],
  [2, 4, 4],
  [4, 8]
]
```

Solution 1

```
public List<List<Integer>> getFactors(int n) {
    List<List<Integer>> result = new ArrayList<List<Integer>>();
    helper(result, new ArrayList<Integer>(), n, 2);
    return result;
}

public void helper(List<List<Integer>> result, List<Integer> item, int n, int star
t){
    if (n <= 1) {
        if (item.size() > 1) {
            result.add(new ArrayList<Integer>(item));
        }
        return;
    }

    for (int i = start; i <= n; ++i) {
        if (n % i == 0) {
            item.add(i);
            helper(result, item, n/i, i);
            item.remove(item.size()-1);
        }
    }
}
```

written by yinfeng.zhang.9 original link here

Solution 2

```
class Solution {
    public:
        void getResult(vector<vector<int>> &result,vector<int> &row,int n){
            int i=row.empty()?2:row.back();
            for(;i<=n/i;++i){
                if(n%i==0){
                    row.push_back(i);
                    row.push_back(n/i);
                    result.push_back(row);
                    row.pop_back();
                    getResult(result,row,n/i);
                    row.pop_back();
                }
            }
        }

        vector<vector<int>> getFactors(int n) {
            vector<vector<int>> result;
            vector<int>row;
            getResult(result,row,n);
            return result;
        }
    };
```

written by jiannan original link here

Solution 3

```
public List<List<Integer>> getFactors(int n) {
    List<List<Integer>> result = new ArrayList<List<Integer>>();
    if (n <= 3) return result;
    helper(n, -1, result, new ArrayList<Integer>());
    return result;
}

public void helper(int n, int lower, List<List<Integer>> result, List<Integer> cur
) {
    if (lower != -1) {
        cur.add(n);
        result.add(new ArrayList<Integer>(cur));
        cur.remove(cur.size() - 1);
    }
    int upper = (int) Math.sqrt(n);
    for (int i = Math.max(2, lower); i <= upper; ++i) {
        if (n % i == 0) {
            cur.add(i);
            helper(n / i, i, result, cur);
            cur.remove(cur.size() - 1);
        }
    }
}
```

written by hahadaxiong original link here

From Leetcoder.