Beautiful Arrangement II

Given two integers `n` and `k`, you need to construct a list which contains `n` different positive integers ranging from `1` to `n` and obeys the following requirement:
Suppose this list is [a_1, a_2, a_3, ... , a_n], then the list [$|a_1 - a_2|$, $|a_2 - a_3|$, $|a_3 - a_4|$, ... , $|a_{n-1} - a_n|$] has exactly `k` distinct integers.

If there are multiple answers, print any of them.

Example 1:

```
Input: n = 3, k = 1
Output: [1, 2, 3]
Explanation: The [1, 2, 3] has three different positive integers ranging from 1 to 3,
 and the [1, 1] has exactly 1 distinct integer: 1.
```

Example 2:

```
Input: n = 3, k = 2
Output: [1, 3, 2]
Explanation: The [1, 3, 2] has three different positive integers ranging from 1 to 3,
 and the [2, 1] has exactly 2 distinct integers: 1 and 2.
```

Note:

1. The `n` and `k` are in the range 1 4.

Solution 1

if you have `n` number, the maximum `k` can be `n - 1`;
if `n` is 9, max `k` is 8.
This can be done by picking numbers interleavingly from head and tail,

```
// start from i = 1, j = n;
// i++, j--, i++, j--, i++, j--

1   2   3   4   5
  9   8   7   6
out: 1 9 2 8 3 7 6 4 5
dif:  8 7 6 5 4 3 2 1
```

Above is a case where `k` is exactly `n - 1`
When k is less than that, simply lay out the rest `(i, j)` in incremental order(all diff is 1). Say if k is 5:

```
     i++ j-- i++ j--  i++ i++ i++ ...
out: 1   9   2   8    3   4   5   6   7
dif:   8   7   6   5    1   1   1   1
```

C++

```
class Solution {
public:
    vector<int> constructArray(int n, int k) {
        vector<int> res;
        for (int i = 1, j = n; i <= j; ) {
            if (k > 1) {
                res.push_back(k-- % 2 ? i++ : j--);
            }
            else {
                res.push_back(k % 2 ? i++ : j--);
            }
        }
        return res;
    }

};
```

C++ Compact

```
class Solution {
public:
    vector<int> constructArray(int n, int k) {
        vector<int> res;
        for (int i = 1, j = n; i <= j; )
            res.push_back(k > 1 ? (k-- % 2 ? i++ : j--) : (k % 2 ? i++ : j--));
        return res;
    }
};
```

Java

```java
class Solution {
    public int[] constructArray(int n, int k) {
        int[] res = new int[n];
        for (int i = 0, l = 1, r = n; l <= r; i++)
            res[i] = k > 1 ? (k-- % 2 != 0 ? l++ : r--) : (k % 2 != 0? l++ : r--);
        return res;
    }
}
```

written by alexander original link here

Solution 2

The requirement of k distinct distance can be achieved from 1, 2, ..., k+1 (<= n), by the following strategy:

```
1, k+1, 2, k, 3, k-1 ...;
The distance of this sequence is k, k-1, k-2, ..., 2, 1
```

Then append the remaining numbers to the list.

```
class Solution {
public:
    vector<int> constructArray(int n, int k) {
        int l = 1, r = k+1;
        vector<int> ans;
        while (l <= r) {
            ans.push_back(l++);
            if (l <= r) ans.push_back(r--);
        }
        for (int i = k+2; i <= n; i++)
            ans.push_back(i);
        return ans;
    }
};
```

written by zestypanda original link here

Solution 3

When `k = n-1`, a valid construction is `[1, n, 2, n-1, 3, n-2, ....]`. One way to see this is, we need to have a difference of `n-1`, which means we need `1` and `n` adjacent; then, we need a difference of `n-2`, etc.

This leads to the following idea: we will put `[1, 2, ...., n-k-1]` first, and then we have `N = k+1` adjacent numbers left, of which we want `k` different differences. This is just the answer above translated by `n-k-1`: we'll put `[n-k, n, n-k+1, n-1, ....]` after.

```
def constructArray(self, n, k):
    ans = range(1, n - k)
    for d in xrange(k+1):
        if d % 2 == 0:
            ans.append(n-k + d/2)
        else:
            ans.append(n - d/2)

    return ans
```

written by awice original link here