Number climber

For every positive integer N, there exists a unique sequence starting with 1 and ending with N and such that every number in the sequence is either the double of the preceeding number or the double plus 1.

For example, given N = 13, the sequence is [1, 3, 6, 13], because . . . :

3 = 2\*1 +1 6 = 2\*3 13 = 2\*6 +1

Write a function that returns this sequence given a number N. Try generating the elements of the resulting list in ascending order, i.e., without resorting to a list reversal or prependig the elements to a list.

using System;

using System.Collections.Generic;

using System.Linq;

public class Kata

{

public static int[] Climb(int n)

{

if(n == 1)

return new int[] {1};

else if(n == 2)

return new int[] {1,2};

else if(n == 3)

return new int[] {1,3};

else if(n % 2 == 0)

{

int[] recursive = Climb(n/2);

List<int> recursiveList = recursive.OfType<int>().ToList();

recursiveList.Add(n);

int[] finalArray = recursiveList.ToArray();

return finalArray;

}

else

{

int[] recursive = Climb((n-1)/2);

List<int> recursiveList = recursive.OfType<int>().ToList();

recursiveList.Add(n);

int[] finalArray = recursiveList.ToArray();

return finalArray;

}

}

}