Fill two instances of all numbers from 1 to n in a specific way

Given a number n, create an array of size 2n such that the array contains 2 instances of every number from 1 to n, and the number of elements between two instances of a number i is equal to i. If such a configuration is not possible, then print the same.

Examples:

```
Input: n = 3
Output: res[] = {3, 1, 2, 1, 3, 2}

Input: n = 2
Output: Not Possible

Input: n = 4
Output: res[] = {4, 1, 3, 1, 2, 4, 3, 2}
```

We strongly recommend to minimize the browser and try this yourself first.

One solution is to Backtracking. The idea is simple, we place two instances of n at a place, then recur for n-1. If recurrence is successful, we return true, else we backtrack and try placing n at different location. Following is C implementation of the idea.

```
// A backtracking based C Program to fill two instances of all numbers
// from 1 to n in a specific way
#include <stdio.h>
#include <stdbool.h>

// A recursive utility function to fill two instances of numbers from
// 1 to n in res[0..2n-1].  'curr' is current value of n.
bool fillUtil(int res[], int curr, int n)
{
     // If current number becomes 0, then all numbers are filled
     if (curr == 0) return true;

     // Try placing two instances of 'curr' at all possible locations
     // till solution is found
     int i;
     for (i=0; i<2*n-curr-1; i++)
     {
        // Two 'curr' should be placed at 'curr+1' distance
        if (res[i] == 0 && res[i + curr + 1] == 0)
        {
           // Plave two instances of 'curr'
           res[i] = res[i + curr + 1] = curr;

           // Recur to check if the above placement leads to a solution
           if (fillUtil(res, curr-1, n))
               return true;

           // If solution is not possible, then backtrack
           res[i] = res[i + curr + 1] = 0;
        }
     }
     return false;
}

// This function prints the result for input number 'n' using fillUtil()
void fill(int n)
{
    // Create an array of size 2n and initialize all elements in it as 0
    int res[2*n], i;
    for (i=0; i<2*n; i++)
       res[i] = 0;

    // If solution is possible, then print it.
    if (fillUtil(res, n, n))
    {
        for (i=0; i<2*n; i++)
           printf("%d ", res[i]);
    }
    else
        puts("Not Possible");
}

// Driver program
int main()
{
  fill(7);
  return 0;
}
```

Run on IDE

Output:

```
7 3 6 2 5 3 2 4 7 6 5 1 4 1
```

The above solution may not be the best possible solution. There seems to be a pattern in the output. I an Looking for a better solution from other geeks.

This article is contributed by **Asif**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

22 Comments Category: Backtracking Tags: Backtracking

Related Posts:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Print all possible paths from top left to bottom right of a mXn matrix
- Tug of War
- Backtracking | Set 7 (Sudoku)
- Backtracking | Set 5 (m Coloring Problem)
- Backtracking | Set 4 (Subset Sum)
- Backtracking | Set 3 (N Queen Problem)
- Backtracking | Set 2 (Rat in a Maze)

(Login to Rate and Mark)

3.5 Average Difficulty : **3.5/5.0**
Based on **2** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share 13 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

22 Comments GeeksforGeeks 1 Login

Recommend 1 Share Sort by Newest

Join the discussion…

Siddharth Gupta • 5 months ago

Similar solution using an extra used array for maintaining state for backtracking.

http://code.geeksforgeeks.org/...

• Reply • Share ›

Billionaire • 5 months ago

for (i=0; i<2*n-curr-1; i++)

can be replaced by:

for (i=0; i+cur+1<2*n; i++)

for simplixity

• Reply • Share ›

Dman • 6 months ago

Simple Backtracking approach

https://ideone.com/aj7BDq

2 • Reply • Share ›

Abhishek Yadav • 7 months ago

Above program will give compile time error . for below Array declaration..
We cannot pass variable in array to declare it dynamic.

```
void fill(int n)
{
// Create an array of size 2n and initialize all elements in it as 0
int res[2*n] // incorrect
...
}
```

correct with below..
int *res=new int[2*n];

• Reply • Share ›

Klaus ➦ Abhishek Yadav • 6 months ago

It is possible in C99.

1 • Reply • Share ›

Bavani Sankar ➦ Klaus • 2 months ago

Yeah, now let's all laugh at Abhishek Yadav.

HAHAHAHAHAH!!!

• Reply • Share ›

Ilia • 8 months ago

I've assumed that the n = 1, 2, 5, 6, 9, 10... are invalid, so it runs much faster up into the early thirties. Having a proof of that would be nice though.

https://ideone.com/lzStdq

• Reply • Share ›

let_us_c • a year ago

```
#include<stdio.h>

static int a[10],r[20],h[10];

int check(int,int);

int rec(int);

int n;

int main()

{

int i,j,x;

scanf("%d",&n);

for(i=n;i>0;i--)
```

see more

• Reply • Share ›

Shahid let_us_c • 3 months ago

Can anyone help me to find the solution of Conference Track Management

You
are planning a big programming conference and have received many
proposals which have passed the initial screen process but you're having
trouble fitting them into the time constraints of the day -- there are
so many possibilities! So you write a program to do it for you.
The conference has multiple tracks each of which has a morning and afternoon session.
Each session contains multiple talks.

Afternoon sessions begin at 1pm and must finish in time for the networking event.
The networking event can start no earlier than 4:00 and no later than 5:00.
No talk title has numbers in it.
All talk lengths are either in minutes (not hours) or lightning (5 minutes).
Presenters will be very punctual; there needs to be no gap between sessions.

Note

see more

• Reply • Share ›

Sudhindra A • a year ago

There can be multiple answers for the given question. But you have chosen the first possible option. This would lead to a pattern, where it will display a possibility in which the given number occupies the first position.

You can either display all the possible outputs(or ignore the pattern as its expected).

By the way, what is the complexity here?

• Reply • Share ›

jitesh Sudhindra A • a year ago

To print all solution go to this link

https://ideone.com/Z86JTL

• Reply • Share ›

Varun Jain • a year ago

I am seeing a pattern as below
solution for 1 and 2 is not possible
solution for 3 and 4 is possible
5 and 6 is not possible
7 and 8 are possible
9 and 10 is not possible
11 and 12 are possible and so on..

1 • Reply • Share ›

Harinath Srinivas Varun Jain • 7 months ago

Yeah bro.. for 4n and (4n-1), n=1,2,...
this problem has a soln..

• Reply • Share ›

Pr Varun Jain • 10 months ago

Nope, it does not follow this patter. for 5 the answer is: 5 2 3 4 2 3 5 1 4 1

and for 6 the answer is:
6 2 3 5 2 3 4 6 1 5 1 4
• Reply • Share ›

Harinath Srinivas Pr • 7 months ago
For ur soln for 6 also bw 3 and 3 there are only 2 numbers..
• Reply • Share ›

sathishcr Pr • 9 months ago
in ur given sol for 5 ...check between 3 there are noly two numbers...
• Reply • Share ›

Pr sathishcr • 9 months ago
Correct. I take that back!
• Reply • Share ›

Guest Varun Jain • a year ago
There can be multiple answers for the given question. But you have chosen the first possible option. This would lead to a pattern, where it will display a possibility in which the given number occupies the first position.

You can either display all the possible outputs(or ignore the pattern as its expected)
• Reply • Share ›

Fazer Varun Jain • a year ago
Can we prove this pattern? Coz it takes too much time for n > 13 for the ones not possible. And if this pattern is true it will improve the solution a lot.
2 • Reply • Share ›

Orxan Fazer • a year ago
Varun's pattern seems correct. It seems like if the number is divisible by 4, then there exists a sequence (except 0). If n is odd, then one could check whether n+1 is divisible by 4 and continue accordingly.
• Reply • Share ›

relue Orxan • a year ago
There isn't a bijection between a valid n and an arrangement. If you take n = 3, you can have more than one valid configuration. One optimization would be to prune based on the possibilities for using some numeral. If n = 3, you have 6 slots. But starting with the first slot, you can use any numeral: 1, 2, 3. The same applies for the second slot but not the third slot. You can't use a 3 in the third slot

either as a starting point or end point since three numerals must exist between the two '3' numerals.

^ | ⌄ • Reply • Share ›

Dave ➦ relue • 10 months ago

```
public class Program1 {

static int arr[];

public static void main(String[] args) {

int number = 4;

arr = new int[number*2];

boolean isSolved = solveProblem(arr, number);

if(isSolved)

{

for(int i = 0; i< arr.length; ++i)

{

System.out.print(arr[i]+" ");
```

see more

^ | ⌄ • Reply • Share ›