# Check if given Sudoku board configuration is valid or not

Difficulty Level : Medium • Last Updated : 18 Jun, 2022

Given a Sudoku Board configuration, check whether it is valid or not.

**Examples:**

```
Input:
    [5 3 - - 7 - - - -]
    [6 - - 1 9 5 - - -]
    [- 9 8 - - - - 6 -]
    [8 - - - 6 - - - 3]
    [4 - - 8 - 3 - - 1]
    [7 - - - 2 - - - 6]
    [- 6 - - - - 2 8 -]
    [- - - 4 1 9 - - 5]
    [- - - - 8 - - 7 9]

Output: True
```

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

- The Sudoku board could be partially filled, where empty cells are filled with the character '.'.
- An empty Sudoku board is also valid.
- A valid Sudoku board (partially filled) is not necessarily solvable. Only the filled cells need to be validated.

Below is the implementation of the above approach:

---

## C++

```
// C++ Program to check whether given sudoku
// board is valid or not
#include <bits/stdc++.h>
using namespace std;

// Checks whether there is any duplicate
// in current row or not
bool notInRow(char arr[][9], int row)
{
    // Set to store characters seen so far.
    set<char> st;

    for (int i = 0; i < 9; i++) {

        // If already encountered before, return false
        if (st.find(arr[row][i]) != st.end())
            return false;

        // If it is not an empty cell, insert value
        // at the current cell in the set
        if (arr[row][i] != '.')
            st.insert(arr[row][i]);
    }
    return true;
}

// Checks whether there is any duplicate
```

```
    for (int i = 0; i < 9; i++) {

        // If already encountered before, return false
        if (st.find(arr[i][col]) != st.end())
            return false;

        // If it is not an empty cell,
        // insert value at the current cell in the set
        if (arr[i][col] != '.')
            st.insert(arr[i][col]);
    }
    return true;
}

// Checks whether there is any duplicate
// in current 3x3 box or not.
bool notInBox(char arr[][9], int startRow, int startCol)
{
    set<char> st;

    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            char curr = arr[row + startRow][col + startCol];

            // If already encountered before, return false
            if (st.find(curr) != st.end())
                return false;

            // If it is not an empty cell,
            // insert value at current cell in set
            if (curr != '.')
                st.insert(curr);
        }
    }
    return true;
}

// Checks whether current row and current column and
// current 3x3 box is valid or not
bool isValid(char arr[][9], int row, int col)
```

```
bool isValidConfig(char arr[][9], int n)
{
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {

            // If current row or current column or
            // current 3x3 box is not valid, return false
            if (!isValid(arr, i, j))
                return false;
        }
    }
    return true;
}

// Drivers code
int main()
{
    char board[9][9] = { { '5', '3', '.', '.', '7', '.', '.', '.', '.' },
                         { '6', '.', '.', '1', '9', '5', '.', '.', '.' },
                         { '.', '9', '8', '.', '.', '.', '.', '6', '.' },
                         { '8', '.', '.', '.', '6', '.', '.', '.', '3' },
                         { '4', '.', '.', '8', '.', '3', '.', '.', '1' },
                         { '7', '.', '.', '.', '2', '.', '.', '.', '6' },
                         { '.', '6', '.', '.', '.', '.', '2', '8', '.' },
                         { '.', '.', '.', '4', '1', '9', '.', '.', '5' },
                         { '.', '.', '.', '.', '8', '.', '.', '7', '9' } }

    cout << (isValidConfig(board, 9) ? "YES\n" : "NO\n");
    return 0;
}
```

## Java

```
// Java Program to check whether given sudoku
// board is valid or not
import java.io.*;
import java.util.*;

class GFG{
```

```
        // Set to store characters seen so far.
        HashSet<Character> st = new HashSet<>();

        for(int i = 0; i < 9; i++)
        {

            // If already encountered before,
            // return false
            if (st.contains(arr[row][i]))
                return false;

            // If it is not an empty cell, insert value
            // at the current cell in the set
            if (arr[row][i] != '.')
                st.add(arr[row][i]);
        }
        return true;
    }

    // Checks whether there is any duplicate
    // in current column or not.
    public static boolean notInCol(char arr[][], int col)
    {
        HashSet<Character> st = new HashSet<>();

        for(int i = 0; i < 9; i++)
        {

            // If already encountered before,
            // return false
            if (st.contains(arr[i][col]))
                return false;

            // If it is not an empty cell,
            // insert value at the current
            // cell in the set
            if (arr[i][col] != '.')
                st.add(arr[i][col]);
        }
        return true;
```

```
public static boolean notInBox(char arr[][],
                               int startRow,
                               int startCol)
{
    HashSet<Character> st = new HashSet<>();

    for(int row = 0; row < 3; row++)
    {
        for(int col = 0; col < 3; col++)
        {
            char curr = arr[row + startRow][col + startCol];

            // If already encountered before, return
            // false
            if (st.contains(curr))
                return false;

            // If it is not an empty cell,
            // insert value at current cell in set
            if (curr != '.')
                st.add(curr);
        }
    }
    return true;
}

// Checks whether current row and current column and
// current 3x3 box is valid or not
public static boolean isValid(char arr[][], int row,
                              int col)
{
    return notInRow(arr, row) && notInCol(arr, col) &&
           notInBox(arr, row - row % 3, col - col % 3);
}

public static boolean isValidConfig(char arr[][], int n)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
```

```
            if (!isValid(arr, i, j))
                return false;
        }
    }
    return true;
}

// Driver code
public static void main(String[] args)
{
    char[][] board = new char[][] {
        { '5', '3', '.', '.', '7', '.', '.', '.', '.' },
        { '6', '.', '.', '1', '9', '5', '.', '.', '.' },
        { '.', '9', '8', '.', '.', '.', '.', '6', '.' },
        { '8', '.', '.', '.', '6', '.', '.', '.', '3' },
        { '4', '.', '.', '8', '.', '3', '.', '.', '1' },
        { '7', '.', '.', '.', '2', '.', '.', '.', '6' },
        { '.', '6', '.', '.', '.', '.', '2', '8', '.' },
        { '.', '.', '.', '4', '1', '9', '.', '.', '5' },
        { '.', '.', '.', '.', '8', '.', '.', '7', '9' }
    };

    System.out.println((isValidConfig(board, 9) ?
                        "YES" : "NO"));
}
}

// This code is contributed by Rohit OBeroi
```

## Python3

```
# Python3 program to check whether
# given sudoku board is valid or not

# Checks whether there is any
# duplicate in current row or not
def notInRow(arr, row):

    # Set to store characters seen so far.
    st = set()
```

```
            # return false
            if arr[row][i] in st:
                return False

            # If it is not an empty cell, insert value
            # at the current cell in the set
            if arr[row][i] != '.':
                st.add(arr[row][i])

        return True

    # Checks whether there is any
    # duplicate in current column or not.
    def notInCol(arr, col):

        st = set()

        for i in range(0, 9):

            # If already encountered before,
            # return false
            if arr[i][col] in st:
                return False

            # If it is not an empty cell, insert
            # value at the current cell in the set
            if arr[i][col] != '.':
                st.add(arr[i][col])

        return True

    # Checks whether there is any duplicate
    # in current 3x3 box or not.
    def notInBox(arr, startRow, startCol):

        st = set()

        for row in range(0, 3):
            for col in range(0, 3):
                curr = arr[row + startRow][col + startCol]
```

```
            # If it is not an empty cell,
            # insert value at current cell in set
            if curr != '.':
                st.add(curr)

    return True

# Checks whether current row and current
# column and current 3x3 box is valid or not
def isValid(arr, row, col):

    return (notInRow(arr, row) and notInCol(arr, col) and
            notInBox(arr, row - row % 3, col - col % 3))

def isValidConfig(arr, n):

    for i in range(0, n):
        for j in range(0, n):

            # If current row or current column or
            # current 3x3 box is not valid, return false
            if not isValid(arr, i, j):
                return False

    return True

# Drivers code
if __name__ == "__main__":

    board = [[ '5', '3', '.', '.', '7', '.', '.', '.', '.' ],
             [ '6', '.', '.', '1', '9', '5', '.', '.', '.' ],
             [ '.', '9', '8', '.', '.', '.', '.', '6', '.' ],
             [ '8', '.', '.', '.', '6', '.', '.', '.', '3' ],
             [ '4', '.', '.', '8', '.', '3', '.', '.', '1' ],
             [ '7', '.', '.', '.', '2', '.', '.', '.', '6' ],
             [ '.', '6', '.', '.', '.', '.', '2', '8', '.' ],
             [ '.', '.', '.', '4', '1', '9', '.', '.', '5' ],
             [ '.', '.', '.', '.', '8', '.', '.', '7', '9' ]]

    if isValidConfig(board, 9):
```

## C#

```
// C# Program to check whether given sudoku
// board is valid or not
using System;
using System.Collections.Generic;
class GFG {

    // Checks whether there is any duplicate
    // in current row or not
    public static bool notInRow(char[, ] arr, int row)
    {

        // Set to store characters seen so far.
        HashSet<char> st = new HashSet<char>();

        for (int i = 0; i < 9; i++) {

            // If already encountered before,
            // return false
            if (st.Contains(arr[row, i]))
                return false;

            // If it is not an empty cell, insert value
            // at the current cell in the set
            if (arr[row, i] != '.')
                st.Add(arr[row, i]);
        }
        return true;
    }

    // Checks whether there is any duplicate
    // in current column or not.
    public static bool notInCol(char[, ] arr, int col)
    {
        HashSet<char> st = new HashSet<char>();

        for (int i = 0; i < 9; i++) {
```

```
            // If it is not an empty cell,
            // insert value at the current
            // cell in the set
            if (arr[i, col] != '.')
                st.Add(arr[i, col]);
        }
        return true;
    }

    // Checks whether there is any duplicate
    // in current 3x3 box or not.
    public static bool notInBox(char[, ] arr, int startRow,
                                int startCol)
    {
        HashSet<char> st = new HashSet<char>();

        for (int row = 0; row < 3; row++) {
            for (int col = 0; col < 3; col++) {
                char curr
                    = arr[row + startRow, col + startCol];

                // If already encountered before, return
                // false
                if (st.Contains(curr))
                    return false;

                // If it is not an empty cell,
                // insert value at current cell in set
                if (curr != '.')
                    st.Add(curr);
            }
        }
        return true;
    }

    // Checks whether current row and current column and
    // current 3x3 box is valid or not
    public static bool isValid(char[, ] arr, int row,
                               int col)
    {
```

```
    public static bool isValidConfig(char[, ] arr, int n)
    {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {

                // If current row or current column or
                // current 3x3 box is not valid, return
                // false
                if (!isValid(arr, i, j))
                    return false;
            }
        }
        return true;
    }

    // Driver code
    public static void Main(string[] args)
    {
        char[, ] board = new char[, ] {
            { '5', '3', '.', '.', '7', '.', '.', '.', '.' },
            { '6', '.', '.', '1', '9', '5', '.', '.', '.' },
            { '.', '9', '8', '.', '.', '.', '.', '6', '.' },
            { '8', '.', '.', '.', '6', '.', '.', '.', '3' },
            { '4', '.', '.', '8', '.', '3', '.', '.', '1' },
            { '7', '.', '.', '.', '2', '.', '.', '.', '6' },
            { '.', '6', '.', '.', '.', '.', '2', '8', '.' },
            { '.', '.', '.', '4', '1', '9', '.', '.', '5' },
            { '.', '.', '.', '.', '8', '.', '.', '7', '9' }
        };

        Console.WriteLine(
            (isValidConfig(board, 9) ? "YES" : "NO"));
    }
}

// This code is contributed by ukasp.
```

## Javascript

```
<script>
```

```
// in current row or not
    function notInRow(arr,row)
    {
        // Set to store characters seen so far.
    let st = new Set();

    for(let i = 0; i < 9; i++)
    {

        // If already encountered before,
        // return false
        if (st.has(arr[row][i]))
            return false;

        // If it is not an empty cell, insert value
        // at the current cell in the set
        if (arr[row][i] != '.')
            st.add(arr[row][i]);
    }
    return true;
    }

    // Checks whether there is any duplicate
// in current column or not.
    function  notInCol(arr,col)
    {
        let st = new Set();

    for(let i = 0; i < 9; i++)
    {

        // If already encountered before,
        // return false
        if (st.has(arr[i][col]))
            return false;

        // If it is not an empty cell,
        // insert value at the current
        // cell in the set
        if (arr[i][col] != '.')
            st.add(arr[i][col]);
```

```
    // Checks whether there is any duplicate
// in current 3x3 box or not.
    function notInBox(arr,startRow,startCol)
    {
        let st = new Set();

    for(let row = 0; row < 3; row++)
    {
        for(let col = 0; col < 3; col++)
        {
            let curr = arr[row + startRow][col + startCol];

            // If already encountered before, return
            // false
            if (st.has(curr))
                return false;

            // If it is not an empty cell,
            // insert value at current cell in set
            if (curr != '.')
                st.add(curr);
        }
    }
    return true;
    }


    // Checks whether current row and current column and
// current 3x3 box is valid or not
    function  isValid(arr,row,col)
    {
        return notInRow(arr, row) && notInCol(arr, col) &&
           notInBox(arr, row - row % 3, col - col % 3);
    }


    function isValidConfig(arr,n)
    {
        for(let i = 0; i < n; i++)
    {
        for(let j = 0; j < n; j++)
```

```
            // false
            if (!isValid(arr, i, j))
                return false;
        }
    }
    return true;
    }

    // Driver code
    let board = [[ '5', '3', '.', '.', '7', '.', '.', '.', '.' ],
             [ '6', '.', '.', '1', '9', '5', '.', '.', '.' ],
             [ '.', '9', '8', '.', '.', '.', '.', '6', '.' ],
             [ '8', '.', '.', '.', '6', '.', '.', '.', '3' ],
             [ '4', '.', '.', '8', '.', '3', '.', '.', '1' ],
             [ '7', '.', '.', '.', '2', '.', '.', '.', '6' ],
             [ '.', '6', '.', '.', '.', '.', '2', '8', '.' ],
             [ '.', '.', '.', '4', '1', '9', '.', '.', '5' ],
             [ '.', '.', '.', '.', '8', '.', '.', '7', '9' ]];

    document.write((isValidConfig(board, 9) ?
                    "YES" : "NO"));


    // This code is contributed by rag2127
</script>
```

**Output:**

YES

**Time Complexity :** O(n*n) ,where n is the number of rows sudoku board.

**Space Complexity :** O(1)

Like 26

Previous

Check if given Sudoku solution is valid or not

Next

Hamiltonian Cycle | Backtracking-6

## RECOMMENDED ARTICLES



01 **Check if given Sudoku solution is valid or not**
21, Oct 20

02 **Validity of a given Tic-Tac-Toe board configuration**
08, Sep 15

03 **Solve Sudoku on the basis of the given irregular regions**
23, Jul 21

04 **Check if the given chessboard is valid or not**
03, Oct 18

05 **Sudoku | Backtracking-7**
14, Jul 12

06 **Program for Sudoku Generator**
10, Apr 17

07 **Check if a king can move a valid move or not when N nights are there in a modified chessboard**
29, Oct 18

08 **Minimum queens required to cover all the squares of a chess board**
23, Sep 18

## Article Contributed By :

Aashish Chauhan
@Aashish Chauhan

## Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

**Improved By :** rituraj_jain, RohitOberoi, ukasp, rag2127, adityapatil12

**Article Tags :** cpp-set, STL, Matrix

**Practice Tags :** Matrix, STL

Improve Article | Report Issue

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

Load Comments

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

## Company

About Us

Careers

In Media

Contact Us

Privacy Policy

Copyright Policy

## Learn

Algorithms

Data Structures

SDE Cheat Sheet

Machine learning

CS Subjects

Video Tutorials

Courses

## News

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

## Languages

Python

Java

CPP

Golang

C#

SQL

Kotlin

## Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

## Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship