Questions Asked Coding (DS Algo) - Shashank

Wednesday, April 6, 2022 6:13 PM

SAP

Maximum cost path in an Undirected Graph such that no edge is visited twice in a row

- Difficulty Level : Hard
- Last Updated : 24 Nov, 2021

From <https://www.geeksforgeeks.org/maximum-cost-path-in-an-undirected-graph-such-that-no-edge-is-visited-twice-in-a-row/>

Longest Common Substring in an Array of Strings

- Difficulty Level : Easy
- Last Updated : 21 Oct, 2020

From <https://www.geeksforgeeks.org/longest-common-substring-array-strings/>

CISCO

1. Given an unsorted array, give the contiguous subarray of length L with the largest Sum.

2. Array pattern whether

PAYTM

Online Test:

Output of the following:

SQL Query
Write a SQL query to show all even marks and list them with Student names all CAPTIAL and ascending order

```
// let s = "00000001111111";
// let l = 0, r = s.length - 1, ans = -1;
// while (l < r) {
//     var mid = Math.floor((l + r) / 2);
//     if (s[mid] == '1') {
//         ans = mid;
//         r = mid - 1;
//     }
//     else {
//         l = mid + 1;
//     }
// }
// console.log(ans)


// let i;
// for (i = 0; i < 3; i++) {
//     const log = () => {
//         console.log(i);
//     }
//     setTimeout(log, 1000)
// }

// function foo() {
//     let a = b = 0;
//     a++;
//     return a;
// }
// foo();
// console.log(typeof a);
// console.log(typeof b);

// var outerFunction = function () {
//     if (true) {
//         var x = 5;
//         console.log(y);
//     }
//     var nestedFunction = function () {
//         if (true) {
//             var y = 7;
//             console.log(x);
//         }
//         if (true) {
//             console.log(y);
//         }
//     }
//     return nestedFunction;
// }
// var myFunction = outerFunction();
// console.log(myFunction());

// let count = 0;
// (function immediate() {
//     if (count === 0) {
//         let count = 1;
```

FUNDING SOCIETIES

Tokenizing a string in C++

From <https://www.geeksforgeeks.org/tokenizing-a-string-cpp/>

ROUND 2

1 - Given a list of words and a string, see which word amongst the words is hidden in the string. EX
words [cat, bat , rat] and string = tcasdasd
Here we see cat is hidden in the string "tca"sdasd

2 - Given a grid, find where the string exists and return all the co-ordinates. You are allowed to move just left and bottom

Ex - [a, c , c   Find cat,
      x, a, t
      M, n , t]

We can see cat on

ROUND 3
System Design and Database design for Ecommerce
SQL Queries

CLOOTRACK

Replace all occurrences of a substring in original string by replacement_string.
Example:
Original_string ="johnmarkmmarkpeter", target_substring = "mark",replacement_string ="fictional" , result = "johnficitonalmfictionalpeter"

GOLDMAN SACHS

Hackerrank Test

https://www.geeksforgeeks.org/distributing-m-items-circle-size-n-starting-k-th-position/

https://www.geeksforgeeks.org/form-minimum-number-from-given-sequence/
Ans https://www.techiedelight.com/decode-the-given-sequence-construct-minimum-number-without-repeated-digits/

F2F Rround 1
https://leetcode.com/discuss/interview-question/394477/Goldman-Sachs-or-Phone-or-Highest-Average-Score-and-Power-of-10
Only the first question from here

```
'''
Given an integer array of size n. Elements of the array is >= 0. Starting from
arr[startInd], follow each element to the index it points to. Find a cycle and
return its length. No cycle is found -> -1.
int lengthOfCycle(int[] arr, int startIndex) {
    // todo
}
Examples:
---------
lengthOfCycle([1, 0], 0); // 2
lengthOfCycle([1, 2, 0], 0); // 3
lengthOfCycle([1, 2, 3, 1], 0); // 3
'''
class Solution:
    def lengthOfCycle(self, arr, startIndex):
        n = len(arr)
        if startIndex < 0 or startIndex >= n: return -1
        slow, fast = arr[startIndex], arr[startIndex]
        if not (slow < n and fast < n and arr[fast] < n): return -1
        while slow < n and fast < n and arr[fast] < n:
            slow = arr[slow]
            fast = arr[fast]
            fast = arr[fast]
            if slow == fast: break
        if slow != fast: return -1
        count = 0
        while True:
            slow = arr[slow]
            fast = arr[arr[fast]]
            count += 1
            if slow == fast: break
        return count
s = Solution()
assert s.lengthOfCycle([1, 2, 0], 0) == 3
assert s.lengthOfCycle([1, 0], 0) == 2
```

```
// console.log(myFunction());

// let count = 0;
// (function immediate() {
//     if (count === 0) {
//         let count = 1;
//         console.log(count);
//         count += 1;
//     }
//     console.log(count);
// })();

//Time complexity of
// let temp = 0;
// for (let i = 0; i < n; i++) {
//     for (let j = 0; j < i * i; j++) {
//         for (let k = i; ~k; k -= 1) {
//             temp++;
//         }
//     }
// }
// console.log(temp)
```

ARCECIUM

https://www.geeksforgeeks.org/length-of-longest-subarray-with-positive-product/

1. Hydrate Nodes - Tree/Graph Based
2. Maximum Length of Subset with Positive Product (Standard Problem)
3. Weird Stock - Ad Hoc Maths

From <https://leetcode.com/discuss/interview-experience/1070858/Arcesium-2021-%3A-SDE-Hiring-(6-months-experience)/853383>
https://leetcode.com/discuss/interview-question/678862/Arcesium-or-OA-2020-or-Positive-Product-and-Weird-Stock

CLASSPLUS

Round 1

1. Basic questions on Parallel vs concurrent. Process vs thread. Promise vs callback
How will you Assign priority to nodejs tasks in a event loop

2. How will you create Many to many relationship in Database

3.Design a Stack that does Push Pop and Getminimum in O(1) time (Can use extra space)
https://towardsdev.com/how-to-solve-min-stack-optimally-with-javascript-eb81a7ae3bb5

QUICKSELL Round 1

Find intersection Point in the below Matrix (i.e elements common to all rows): Ans : [1,5]

```
    intersection(
        [1, 5, 9],
        [1, 3, 5],
        [1, 5, 8],
        [100, 150, 300, 500, 1, 5]
    )
```

https://blog.rishabhjairath.in/posts/async-waterfall-js (Same as Below)
1. Takes 2 arguments (An array of asynchronous functions & a final callback function)
2. Pass the result of one asynchronous function to the next using doneCallback
3. The values passed in the doneCallback of one function should be available as parameters to the next function
4. Pass the result of last function to the final callback function
5. If an error occurs during any of the function's execution, directly jump to the final callback function, with the error parameter

> Note : Write the code in vanilla JS, without using async await external libraries like Promise, etc.
>

```jsx
function waterfall (arrayOfFunctions, finalCallback) {
        // Your code here
}

const arrayOfFunctions = [
  function(doneCallback) {
    setTimeout(function() {
      console.log("FIRST");
      doneCallback(null, "b");
    }, 100);
  },
  function(param, doneCallback) {
    setTimeout(function() {
      console.log("SECOND", param);
      doneCallback(null, "c", "d");
    }, 50);
  },
  function(param1, param2, doneCallback) {
```

```
            if slow == fast: break
        return count
s = Solution()
assert s.lengthOfCycle([1, 2, 0], 0) == 3
assert s.lengthOfCycle([1, 0], 0) == 2
assert s.lengthOfCycle([1, 2, 3, 1], 0) == 3
assert s.lengthOfCycle([1, 2, 3, 4], 0) == -1
assert s.lengthOfCycle([1, 2, 3, 4], -1) == -1
assert s.lengthOfCycle([1, 2, 3, 4], 4) == -1
assert s.lengthOfCycle([2, 3, 4, 0], 0) == -1
assert s.lengthOfCycle([2, 3, 0], 0) == 2
```

BZAAR

https://www.geeksforgeeks.org/find-element-array-sum-left-array-equal-sum-right-array/
Ans : Method 4

https://gist.github.com/abu-hasib/89875a0cce90b3b8bdc26ae5e08c6767

https://gist.github.com/adityak74/724a12724ae85c664f58bda98643d06e

Round 2

```
// [3,5,-4,8,11,1,-1,6]
// 10
// 11,-1


const nums = [3,5,-4,8,11,1,-1,6];
const target = 10;

let map = new Map();

find = () => {
   for(let num of nums){
      let betterHalf = target - num;
      if(map.get(betterHalf)){
         return [num, betterHalf];
      }
      map.set(num, true);
   }
   return [];
}

console.log(find())


//Return if brackets are balanced or not

// {([])}
// {}()
// {[}]

// function checkParan(s){

//     let other = {
//        '}' : '{',
//        ']' : '[',
//        ')' : '('
//     }

//     const stack = [];

//     for(let char of s){
//        if((stack[stack.length-1] || null) === other[char]){
//           stack.pop();
//        }
//        else{
//           stack.push(char)
//        }
//     }

//     return stack.length === 0
// }


// console.log(checkParan('{([])}'));
// console.log(checkParan('{}()'));
// console.log(checkParan('{[}]'));
// console.log(checkParan('}])'));
// console.log(checkParan('{{('));
```

PAYTM Round 1

BFS
https://www.geeksforgeeks.org/maximum-time-required-for-all-patients-to-get-infected/

DFS

```
    console.log("SECOND", param);
    doneCallback(null, "c", "d");
  }, 50);
 },
 function(param1, param2, doneCallback) {
  setTimeout(function() {
    console.log("THIRD", param1, param2);
    doneCallback(null, "e");
  }, 10);
 }
];

const finalCallback = function(err, result) {
 console.log("err", err);
 console.log("result", result);
};

waterfall(arrayOfFunctions, finalCallback);
```

--Expected Output--

```
FIRST
SECOND b
THIRD c d
err null
result e
```

```
```

FINAL ROUND

Use database Postgres

Simple forum backend

- Consists of 2 things - posts and comments
- A post can have multiple comments
- A comment can have multiple comments
- All posts and comments will be anonymous. So, no user authentication needed.
- Posts have a character limit of 1000 characters
- Comments have a character limit of 200 characters
- So, the structure can be shown as:
  - post_1
    - comment_1
      - comment_1.1
      - comment_1.2
    - comment_2
      - comment_2.1
        - comment_2.1.1
        - comment_2.1.2
      - comment_2.2
    - comment_3
- For this task, we'll refer
  - comment_1, comment_2, comment_3 as "level 1" comments
  - comment_1.1, comment_1.2, comment_2.1 as "level 2" comments
  - comment_2.1.1, comment_2.1.2 as "level 3" comments
  - and so on

  ### APIs needed

  - **Add post**
    - This API will accept title & description. Both parameters are required
    - It will return the post object as shown in **[Get posts]** API
  - **Add comment**
    - Comment can be added to both post or comment.
    - This API will accept comment text and post ID or comment ID. Both parameters are required
    - It will return the comment object as shown in **[Get comments]** API
  - **Get posts**
    - This will return an array of post objects ordered from latest to oldest
    - Each post object should contain
      - post's unique ID
      - title
      - description
      - post's creation date
      - number of level 1 comments on the post
    - This API must be paginated based on post's creation date
  - **Get comments**
    - This API will be used to fetch comments to both posts and comments.
    - It clan either receive post ID or comment ID
      - If post ID is passed in the API, it should return
        - level 1 comments for that post ID, and
        - level 2 comments on those level 1 comments
      - If level 1 comment ID is passed in the API, it should return
        - level 2 comments for that comment ID, and
        - level 3 comments on those level 2 comments
      - and so on
    - It will return an array of comment objects ordered from latest to oldest
    - Each comment object should contain
      - comment's unique ID
      - comment text
      - comment's creation date
      - child comments
    - (The comments should be structured in the database in such a way that pagination is possible in the future)

BFS
https://www.geeksforgeeks.org/maximum-time-required-for-all-patients-to-get-infected/

DFS
https://www.geeksforgeeks.org/unique-paths-in-a-grid-with-obstacles/

What is shallow copy and stuff, output questions

SPINNY - InterviewVector

1. Design an algorithm or data structure for storing common time between users loggin in and loggin out .

2. Given a number , say 123542, how many changes are needed to be made in the second half of the string so that it is anagram of the first half of string

UPTYCS

Round 1:

1> Let arr = [[[[1]], [5]], 7, [3], [[[4]]], [[[[6]]]]]
Return the value above in an array after sorting [1,3,4,5,6,7]

```
2. Given a Number X, Return the value K which is the nearest to X if we do 3^K (3
raised to power K)
```

```
3> Emulate promise.All() with callback. Basically given 3 URLs call them and
emulate promise.All()
```

```
4> Call, Bind and Apply in Javascript
```
https://blog.bitsrc.io/understanding-call-bind-and-apply-methods-in-javascript-33dbf3217be

WALMART

R1

```
// Right view of the tree
//                1
//              2  3
//          4  5     6 7
//                 8     9
//              10

class Node {
    constructor(val) {
        this.left = null;
        this.right = null;
        this.val = val;
    }
}
var maxSoFar = 1;
function printRightHelper(node, level) {
    if (!node) return;
    if (maxSoFar < level) {
        console.log(node.val + " --> ");
        maxSoFar = level;
    }
    printRightHelper(node.right, level + 1);
    printRightHelper(node.left, level + 1);
}

function printRight(root) {
    printRightHelper(root, 1)
}
```

//Program to check if Subtring

```
// function isSubring(str1, str2) {
//     let s1len = str1.length;
//     let s2len = str2.length;
//     for (let i = 0; i <= s2len - s1len; i++) {
//         let j;
//         for (j = 0; j < s1len; j++) {
//             if (str2[i + j] !== str1[j]) {
//                 break;
//             }
//         }
//         if (j === s1len) {
//             return true;
//         }
//     }
//     return false;
// }
// console.log(isSubring('cd', 'abcd'))
// console.log(isSubring('cdx', 'abcd'))
// console.log(isSubring('Xyzd', 'Fdsafafsdfsdabcd'))
```

ADOBE

House Robber 1 and House Robber 2

```
/** Write a production-grade api which accepts a fruit name as a query param,
 * and then searches for the fruit in a file of given path,
 * in a database table/collection and a rest api.
The api should be called like -
/word-count?word=orange

You can assume the file fruits.txt is in the same directory as
the node js server running, which has one fruit defined per line.
But the file is pretty huge having millions of lines.

apple
orange
banana
Mango


The rest api url is something like
https://jsonkeeper.com/b/VLCX
Which again returns data in the format -
[
    {
      fruitName: 'apple',
      quantity: 2
    },
    {
      fruitName: orange,
      quantity: 17
    }
]

 Let's assume a mongodb collection of millions of fruit entries across
 warehouses daily where documents are like -
 {
  fruitName: 'apple',
  pricePerQty: 5
  totalPrice: 25,
  addedOn: 1628240462 // timestamp
 },
  {
  fruitName: 'orange',
  pricePerQty: 10,
  totalPrice: 100,
  addedOn: 1628249462 // timestamp
 },
 {
  fruitName: 'orange',
  pricePerQty: 12,
  totalPrice: 144,
  addedOn: 1628269462 // timestamp
 }


The api should be called like -
/word-count?word=orange


The api returns a status 200 with the count if found atleast 1 quantity.
Or returns 404 if no fruit is found.
*/
const express = require('express')
const fs = require('fs')
const request = require('request')
const app = express()
const port = 3000
/* Define your api below */
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
});
//API to get word count. Use as /word-count?word=orange
app.get('/word-count', function(req, res){
    let count = 0;
    let word = req.query.word;
    const filename = 'fruits.txt';
    const readStream = fs.createReadStream(filename);
  try{
    //Reading from file
    readStream.on('data', function(chunk) {
      let line = chunk.toString('utf8');
      if(line === word){
        count++;
      }
    });
    readStream.on('err', function(){
      throw new err;
    }
    //Reading from API
    request('https://jsonkeeper.com/b/VLCX', function(err, res, data){
        let json = JSON.parse(data);
        for(let fruit of json){
          if(fruit.fruitName === word){
               count += fruit.quantity;
               break;
          }
        }
    });

    //Reading from DB
```

```
/////////ROUND 2

// Friend //valid

// friend // valid

// FRIEND // valid

// FRiend // invalid

// friEND //invalid

// Valid String

// 1. Can start single capital letter followed by smaller case

// 2. all can be smaller case

// 3. All can be upper case


function checkValid(str) {
   let arr = [...str];
   let countUpper = countLower = 0;

   for (let char of arr) {
      if (char === char.toUpperCase()) {
         countUpper++;
      }
      else {
         countLower++;
      }
   }

   //Case 1 and 2
   if (countUpper === str.length || countLower === str.length) return 'valid';

   //Case 3
   if (countUpper === 1 && arr[0] === arr[0].toUpperCase()) return 'valid';

   return 'invalid';

};

console.log(checkValid('Friend'))
```

Q2
++=
Remove null and empty values

Input:

```
const obj = {
   Name: "John Snow",
   Details: {
      age: 39,
      address_line: "",
      city: "Bangalore",
      country: {
         zipcode: "",
         name: 'India',
         dob: undefined
      }
   }
}

//Delete empty and null - Even for nested levels

function cleanup(obj){
   for(var prop in obj){
      if(!obj[prop] || obj[prop] === "" || obj[prop].length === 0){
         delete obj[prop];
      }
      else if(typeof obj[prop] === 'object'){
         cleanup(obj[prop]);
      }
   }
   return obj;
}


console.log(cleanup(obj));


Output: {
          Name: "John Snow",
  Details: {
    age: 39,
    city: "Bangalore",
    country:{
            name: "India"
    }
  }
}
```

```
            }
          }
        });

        //Reading from DB
        var sqlReq = new sql.Request();
        sqlReq.query(`SELECT Count(fruitName) FROM fruit_table WHERE fruitName=
    ${word}`, function (err, record){
          if(err){
            console.log("Failure in reading database");
            throw err;
          }
          count += record;
        });
        // The api returns a status 200 with the count if found atleast 1 quantity.
        // Or returns 404 if no fruit is found.
        count > 0 ? res.status(200).send(count) : res.status(404).send(0);
      }
      catch(err){
        console.log("Error in getting the count "+err);
        res.send(err);
      }
    }
```

URBAN COMPANY
https://www.geeksforgeeks.org/program-to-generate-all-possible-valid-ip-addresses-from-given-string-set-2/

NIYO

```
for (var i = 0; i < 3; i++) {
    setTimeout(function log() {
    console.log(i);
  }, 1000);
}


// An integer array nums and two integers k and t, return true if there are two
distinct indices i and j in the array. nums[i] - nums[j] <= t and abs(i - j) <=
k.
nums = [1,2,3,1], k = 3, t = 0
nums = [1,5,9,1,5,9], k = 2, t = 3
nums = [1,0,1,1], k = 1, t = 2

function findCombo(nums, k, t) {
    for (let i = 0; i < nums.length; i++) {
        for (let j = i + 1; j < nums.length; j++) {
            if (Math.abs(nums[i] - nums[j]) <= t && Math.abs(i - j) <= k) return
true;
        }
    }
    return false;
}
console.log(findCombo([1, 2, 3, 1], 3, 0))
console.log(findCombo([1, 5, 9, 1, 5, 9], 2, 3))
console.log(findCombo([1, 0, 1, 1], 1, 2))



// // //Distribute candies equally
// //     Root 0 - 1 (Two steps)
// // L 1         right 0 1 (one step)
// // if excess give eithr child or parent - Steps => MyTotal - 1 Steps => 2
// //                                                 2 - 1 Steps => 1
// //                                                                 3
class Node {
    constructor(val) {
        this.left = null;
        this.right = null;
        this.val = val;
    }
}
var count = 0;
function distribute(root) {
    if (!root) return 0;
    let leftSteps = distribute(root.left);
    let rightSteps = distribute(root.right);
    count += Math.abs(leftSteps) + Math.abs(rightSteps);
    return root.val + leftSteps + rightSteps - 1;
}


let root = new Node(3);
let left = new Node(0);
let right = new Node(0);
root.left = left;
root.right = right;

distribute(root);
console.log(count);
```

```
            name: "India"
        }
      }
    }
```

Q3

++

```
// Assume you use {}

// Input: 1
// Output: {}


// Input: 2
// Output: {}{} , {{}}

// Input: 3

// Output:

// {}{}{},

// {{}}{},

// {}{{}},

// {{{}}},

// {{}{}}
// }

function paranCombos(n, open, closed, str) {

    //Base Case
    if (open === n && closed === n) {
        result.push(str);
        return;
    }

    //When to add OPEN
    if (open < n) {
        paranCombos(n, open + 1, closed, str + '{');
    }

    //When to add CLOSE
    if (closed < open) {
        paranCombos(n, open, closed + 1, str + '}');
    }
}



let n = 3;
let result = [];
paranCombos(n, 0, 0, '');
console.log(result);
{{{}}},

{{}{}}
}
```

LETS TRANSPORT

R1

```
// 2) Sort array with 0,1 and 2s.( Need to do in 1 traversal )
// Example
// Input : {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1};
// Output : {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2};
const arr = [0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1, 2, 0, 2];
let left = 0;
let right = arr.length - 1;
let count = 0;
let ptr = left;
while (ptr <= right) {
    if (arr[ptr] === 0) {
        [arr[ptr], arr[left]] = [arr[left], arr[ptr]];
        left++;
        ptr++;
        count++;
    }
    else if (arr[ptr] === 1) {
        ptr++;
        count++;
    }
    else if (arr[ptr] === 2) {
        [arr[ptr], arr[right]] = [arr[right], arr[ptr]];
        right--;
    }
}
console.log(count, arr.length)
console.log(arr)



// 1) There are n houses build in a line, each of which contains some value in
```

```
it. A thief is going to steal the maximal value of these houses, but he can’t
steal in two adjacent houses because the owner of the stolen houses will tell his
two neighbors left and right side. What is the maximum stolen value?
// Examples:
// Input: hval[] = {6, 7, 1, 3, 8, 2, 4}
// Output: 19
// Explanation: The thief will steal 6, 1, 8 and 4 from the house.
// Input: hval[] = {5, 3, 4, 11, 2}
// Output: 16
// Explanation: Thief will steal 5 and 11
const hval = [5, 3, 4, 11, 2];
const memo = [];
function findMax(hval, i) {
    if (i < 0) {
        return 0;
    }
    if (i == 0) {
        return hval[0];
    }
    if (memo[i]) {
        return memo[i];
    }
    //Pick current
    let select = hval[i] + findMax(hval, i - 2);
    //Ignore Current
    let ignore = findMax(hval, i - 1);
    return memo[i] = Math.max(select, ignore);
}
console.log(findMax(hval, hval.length - 1, []))
```

wa

```
//SHASHANK SHEKHAR
//  2        3      4      5      6      7      8      9
// "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"
// string num = "6349"

let numberCombo = function(digits){
  let input = '';
  for(let char of digits){
    if(char !== '1' && char !== '0'){
        input += char;
    }
  }

  digits = input;
  if(!digits.length) return [];
  let teleMap = {
    '2' : ['a','b','c'],
    '3' : ['d','e','f'],
    '4' : ['g','h','i'],
    '5' : ['j', 'k', 'l'],
    '6' :
    '7' :
    '8' :
    '9' :
  }
  let result = [];
  function permuteAll(temp_str, i){
    if(i === digits.length){
      result.push(temp_str);
      return;
    }

    for(let val of teleMap[digits[i]]){
        permuteAll(temp_str+val, i+1);
    }
  }
  permuteAll('',0);
  return result;
}

//Q2
// class TreeNode
// {
//     int data;
//
//     TreeNode left;
//     TreeNode right;
//
//     TreeNode prev;
//     TreeNode next;
// }
//
//           1
//        /     \
//      2         3
//        \     /   \
//         4   5     6
//
// 1 -> 2 -> 3 -> 4 -> 5 -> 6
// 1 <- 2 <- 3 <- 4 <- 5 <- 6

let result = [];
function levelOrder(){
    var q = [];
    q.push(root);
    while(q.length !== 0){
        let tmpNode = q.shift();
        result.push(tmpNode);
        if(tmpNode.left !== null){
          q.push(tmpNode.left);
        }
        if(tmpNode.right !== null){
          q.push(tmpNode.right)
        }
    }
}
let prvs = null;
for(let i = 0; i < result.length; i++){
    node.prev = prvs;
    prvs = node;
    node.next = result[i+1] || null;
}


// 2 player game - Turn based
// A grid of size M X N
// Moves in 4 directions
// Spawn gems randomly
// Collect gems on moving on them
// Aim : Collect higher gems than opponent
// Has multiple rounds
// Time limit
//spawn -
// i/p: position(x,y), distance
//o/p: rand(position[])
// Board
const board = [][];
const visited = [][];
//Return one of the possible co-ordinates
function spawnRandom(x, y, distance){
  const possible = [];
  function traverse(x, y, distRemaining){

  if(x < 0 || y < 0 || x > board.length || y > board[0].length || visited[x][y])
return;

    visited[x][y] = 1;

    if(distRemaining === 0){
        possible.push([x,y]);
        return;
    }
    traverse(x + 1, y, distRemaining - 1);
    traverse(x , y + 1, distRemaining - 1);
    traverse(x - 1, y, distRemaining - 1);
    traverse(x , y - 1, distRemaining - 1);
}
traverse(x, y, distance);
return possible[Math.floor(Math.random()*possible.length)];
}


// Game
// Cell
// Move
// Player
public class Game{
  private Player[] player
  private Board board
  private Player currentPlayer
  private Round round
  public playerMove(Player player, init_x, init_y, final_x, final_y){

  }
}
```