Leetcode

34. Search for a Range

var searchRange = function(nums, target) {

   var start = findFirst(target);

   if(start==nums.length||nums[start]!=target) return [-1,-1];

   var end = findFirst(target+1)-1 > 0 ? findFirst(target+1)-1 : 0;

   return [start, end];

   function findFirst(target) {

       var l = 0;

       var r = nums.length;

       while(l<r) {

           var mid = l+Math.floor((r-l)/2);

           if(nums[mid]>=target) {

               r = mid;

           }

           else {

               l = mid+1;

           }

       }

       return l;

   }

};

36. Valid Sudoku

var isValidSudoku = function(board) {

   for(let i=0; i<board.length; i++) {

       for(let j=0; j<board[0].length; j++) {

           if(board[i][j]=='.') continue;

           if(!isValid(i,j)) return false;

       }

   }

   return true;

   function isValid(x,y) {

       var c = board[x][y];

       for(let i=0; i<9; i++) {

           //check row & col

           if(board[x][i]==c && i!=y) return false;

           if(board[i][y]==c && i!=x) return false;

           //check square

           var row = Math.floor(x/3)\*3+Math.floor(i/3);

           var col = Math.floor(y/3)\*3+Math.floor(i%3);

           if(board[row][col]==c && row!=x && col!=y) return false;

       }

       return true;

   }

};

37. Sudoku Solver

var solveSudoku = function(board) {

   if(board==null || board.length==0) {

       return;

   }

   play(board);

   function isValid(x,y,c) {

       for(let i=0; i<9; i++) {

           if(board[i][y]==c) return false;

           if(board[x][i]==c) return false;

           var row = 3\*Math.floor(x/3)+Math.floor(i/3);

           var col = 3\*Math.floor(y/3)+Math.floor(i%3);

           if(board[row][col]==c) return false;

       }

       return true;

   }

   function play(board) {

       for(let i=0; i<board.length; i++) {

           for(let j=0; j<board[0].length; j++) {

               if(board[i][j]=='.') {

                   for(let c='1'; c<='9'; c++) {

                       if(isValid(i,j,c)) {

                           board[i][j] = c;

                           if(play(board)) {

                               return true;

                           }

                           else {

                               board[i][j] = '.';

                           }

                       }

                   }

                   return false;

               }

           }

       }

       return true;

   }

};

39. Combination Sum

var combinationSum = function(candidates, target) {

   var res = [];

   if(candidates==null||candidates.length==0) {

       return res;

   }

   candidates.sort(function(a,b) {

       return a-b;

   });

   var list = [];

   helper(res, list, 0, target);

   return res;

   function helper(res, list, start, target) {

       if(target==0) {

           var copy = [].concat(list);

           res.push(copy);

       }

       else if(target>=candidates[start]) {

           for(let i=start; i<candidates.length; i++) {

               list.push(candidates[i]);

               helper(res, list, i, target-candidates[i]);

               list.pop();

           }

       }

   }

};

40. Combination Sum II

var combinationSum2 = function(candidates, target) {

   var res = [];

   if(candidates==null||candidates.length==0) {

       return res;

   }

   candidates.sort(function(a,b) {

       return a-b;

   });

   var list = [];

   helper(res, list, 0, target);

   return res;

   function helper(res, list, start, target) {

       if(target==0) {

           var copy = [].concat(list);

           res.push(copy);

       }

       for(let i=start; i<candidates.length && candidates[i]<=target; i++) {

           if(i>start && candidates[i-1]==candidates[i]) continue;

           list.push(candidates[i]);

           helper(res, list, i+1, target-candidates[i]);

           list.pop();

       }

   }

};

44. Wildcard Matching

var isMatch = function(s, p) {

   var i = 0, j = 0;

   var startIdx = -1;

   var match = 0;

   while(i<s.length) {

       // find one to one match

       if(j<p.length && (p[j]=='?' || p[j]==s[i])) {

           i++;

           j++;

       }

       //find '\*'

       else if(j<p.length && p[j]=='\*') {

           startIdx = j;

           match = i;

           j++;

       }

       //last p pointer is \*, update string pointer

       else if(startIdx!==-1) {

           j = startIdx+1;

           match++;

           i = match;

       }

       else {

           return false;

       }

   }

   //check remaining in p

   while(j<p.length && p[j]=='\*') {

       j++;

   }

   return j==p.length;

};

51. N-Queens

/\*\*

\* @param {number} n

\* @return {string[][]}

\*/

var solveNQueens = function(n) {

   var board = new Array(n);

   for(let i=0; i<n; i++) {

       board[i] = new Array(n).fill('.');

   }

   var res = [];

   play(board,0);

   return res;

   function play(board,col) {

       if(col==n) {

           pushRes(board.slice());

           return;

       }

       for(let i=0; i<n; i++) {

           if(isValid(board,i,col)) {

               board[i][col] = 'Q';

               play(board, col+1);

               board[i][col] = '.';

           }

       }

   }

   function pushRes(arr) {

       var strings = [];

       for(let i=0; i<n; i++) {

           var str = arr[i].join('');

           strings.push(str);

       }

       res.push(strings);

   }

   function isValid(board,x,y) {

       for(let i=0; i<n; i++) {

           for(let j=0; j<n; j++) {

               if((x+j==i+y) && board[i][j]=='Q') return false;

               if((i+j==x+y) && board[i][j]=='Q') return false;

               if((x==i) && board[i][j]=='Q') return false;

           }

       }

       return true;

   }

};

52. N-Queens II

var totalNQueens = function(n) {

   if(n==0) return 0;

   var row = new Array(n).fill(0);

   var res = 0;

   play(row, 0);

   return res;

   function play(row, index) {

       if(index==n) {

           res++;

           return;

       }

       for(let i=0; i<n; i++) {

           if(isValid(index,i)) {

               row[index]=i;

               play(row, index+1);

           }

       }

   }

   function isValid(x,y) {

       for(let i=0; i<x; i++) {

           if(row[i]==y) return false;

           if(Math.abs(row[i]-y)==Math.abs(i-x)) return false;

       }

       return true;

   }

};

62. Unique Paths

var uniquePaths = function(m, n) {

   var dp = new Array(m).fill(0);

   for(let i=0; i<m; i++) {

       dp[i] = [];

       dp[i][0] = 1;

   }

   for(let i=0; i<n; i++) {

       dp[0][i] = 1;

   }

   for(let i=1; i<m; i++) {

       for(let j=1; j<n; j++) {

           dp[i][j] = dp[i-1][j]+dp[i][j-1];

       }

   }

   return dp[m-1][n-1];

};

63. Unique Paths II

var uniquePathsWithObstacles = function(obstacleGrid) {

   if(obstacleGrid==null||obstacleGrid.length==0||obstacleGrid[0].length==0) return 1;

   var m = obstacleGrid.length;

   var n = obstacleGrid[0].length;

   var dp = [];

   for(let i=0; i<m; i++) {

       dp[i] = new Array(n).fill(0);

   }

   for(let i=0; i<m; i++) {

       if(obstacleGrid[i][0]==1) {

           break;

       }

       dp[i][0] = 1;

   }

   for(let i=0; i<n; i++) {

       if(obstacleGrid[0][i]==1) {

           break;

       }

       dp[0][i] = 1;

   }

   for(let i=1; i<m; i++) {

       for(let j=1; j<n; j++) {

           if(obstacleGrid[i][j]==1) {

               dp[i][j] = 0;

           }

           else {

               dp[i][j] = dp[i-1][j]+dp[i][j-1];

           }

       }

   }

   return dp[m-1][n-1];

};

68. Text Justification

var fullJustify = function(words, maxWidth) {

   var res = [];

   var i = 0;

   while(i<words.length) {

       var count = words[i].length;

       var j = i+1;

       //find the last word in line

       while(j<words.length) {

           if(words[j].length+count+1>maxWidth) {

               break;

           }

           count = words[j].length+count+1;

           j++;

       }

       var line = "";

       var diff = j-1-i;

       //if it's the last line or only one word -> left justify

       if(j==words.length||diff==0) {

           for(let x=i; x<j; x++) {

               line += words[x]+" ";

           }

           line = line.slice(0,-1);

           for(let x=line.length; x<maxWidth; x++) {

               line += " ";

           }

       }

       //middle justify

       else {

           var space = Math.floor((maxWidth-count)/diff);

           var rest = (maxWidth-count)%diff;

           for(let x=i; x<j; x++) {

               line += words[x];

               if(x<j-1) {

                   var morespace = x-i<rest ? 1:0;

                   for(let y=0; y<=(space+morespace); y++) {

                       line += " ";

                   }

               }

           }

       }

       res.push(line);

       i = j;

   }

   return res;

};

72. Edit Distance

var minDistance = function(word1, word2) {

   if(word1==word2) return 0;

   if(word1.length==0 || word2.length==0) return Math.abs(word1.length-word2.length);

   var dp = [];

   var n1 = word1.length;

   var n2 = word2.length;

   for(let i=0; i<=n1; i++) {

       dp[i] = new Array(n2).fill(0);

       dp[i][0] = i;

   }

   for(let i=0; i<=n2; i++) {

       dp[0][i] = i;

   }

   for(let i=1; i<=n1; i++) {

       for(let j=1; j<=n2; j++) {

           if(word1[i-1]==word2[j-1]) {

               dp[i][j] = dp[i-1][j-1];

           }

           else {

               dp[i][j] = Math.min(dp[i-1][j-1],dp[i-1][j],dp[i][j-1])+1;

           }

       }

   }

   return dp[n1][n2];

};

73. Set Matrix Zeroes

var setZeroes = function(matrix) {

   if(matrix==null||matrix.length==0||matrix[0].length==0) {

       return;

   }

   var firstRowZero = false;

   var firstColZero = false;

   //check if zero in the first row

   for(let i=0; i<matrix[0].length; i++) {

       if(matrix[0][i]==0) {

           firstRowZero = true;

           break;

       }

   }

   //check if zero in first col

   for(let i=0; i<matrix.length; i++) {

       if(matrix[i][0]==0) {

           firstColZero = true;

           break;

       }

   }

   //check zeros and mark if in the first row&col

   for(let i=1;  i<matrix.length; i++) {

       for(let j=1; j<matrix[0].length; j++) {

           if(matrix[i][j]==0) {

               matrix[i][0] = 0;

               matrix[0][j] = 0;

           }

       }

   }

   //set zeros

   for(let i=1;  i<matrix.length; i++) {

       for(let j=1; j<matrix[0].length; j++) {

           if(matrix[i][0]==0 || matrix[0][j]==0) {

               matrix[i][j] = 0;

           }

       }

   }

   if(firstRowZero) {

       for(let i=0; i<matrix[0].length; i++) {

           matrix[0][i] = 0;

       }

   }

   if(firstColZero) {

       for(let i=0; i<matrix.length; i++) {

           matrix[i][0] = 0;

       }

   }

};

76. Minimum Window Substring

var minWindow = function(s, t) {

   if(s==null||s.length==0||t==null||t.length==0) return "";

   var i=0, j=0;

   var tmap = {};

   var smap = {};

   for(let k=0; k<t.length; k++){

       if(tmap[t[k]]==undefined) {

           tmap[t[k]] = 0;

       }

       tmap[t[k]]++;

   }

   var found = 0;

   var length = Number.MAX\_VALUE;

   var res="";

   while(j<s.length){

       if(found<t.length){

           if(tmap[s[j]]>0){

               if(smap[s[j]]==undefined) {

                   smap[s[j]] = 0;

               }

               smap[s[j]]++;

               if(smap[s[j]]<=tmap[s[j]]){ // find a character

                   found++;

               }

           }

           j++;

       }

       while(found==t.length){ //find a window

           if(j-i<length){ //check min length

               length=j-i;

   res=s.substring(i,j);

           }

           if(tmap[s[i]]>0){

               smap[s[i]]--; //shrink window size from begining

               if(smap[s[i]]<tmap[s[i]]){

                   found--;

               }

           }

           i++;

       }

   }

   return res;

};

96. Unique Binary Search Trees

var numTrees = function(n) {

   var dp = Array(n+1).fill(0);

   dp[0] = dp[1] = 1;

   for(let i=2; i<=n; i++) {

       for(let j=1; j<=i; j++) {

           dp[i] += dp[j-1]\*dp[i-j];

       }

   }

   return dp[n];

};

127. Word Ladder

var ladderLength = function(beginWord, endWord, wordList) {

   if(wordList.indexOf(endWord)<0) {

       return 0;

   }

   var queue = [];

   queue.push(beginWord);

   var level = 1;

   while(queue.length>0) {

       var size = queue.length;

       for(let i=0; i<size; i++) {

           var cur = queue.shift();

           if(cur==endWord) {

               return level;

           }

           var nexts = findNext(cur,wordList);

           for(let next of nexts) {

               queue.push(next);

           }

       }

       level++;

   }

   return 0;

   function findNext(word, list) {

       var abc = "abcdefghijklmnopqrstuvwxyz";

       var res = [];

       for(let i=0; i<26; i++) {

           for(let j=0; j<word.length; j++) {

               var newword = word.slice(0,j)+abc[i]+word.slice(j+1);

               if(list.indexOf(newword)>=0) {

                   remove(list,newword);

                   res.push(newword);

               }

           }

       }

       // console.log(res);

       return res;

   }

   //move value to first, and shift array

   function remove(arr, value) {

       if(arr.indexOf(value)==0) {

           arr.shift();

       }

       else {

           var tmp = arr[0];

           var idx = arr.indexOf(value);

           arr[idx] = tmp;

           arr.shift();

       }

   }

};

140. Word Break II

var wordBreak = function(s, wordDict) {

//     var hash = {};

//     return helper(s);

//     function helper(str) {

//         if(hash[str]!==undefined) {

//             return hash[str]; //prune branches

//         }

//         var res = [];

//         if(str.length===0) {

//             res.push("");

//             return res;

//         }

//         for(var word of wordDict) {

//             if(str.startsWith(word)) {

//                 var substr = str.substring(word.length);

//                 var sublist = helper(substr);

//                 for(var l of sublist) {

//                     var list = word;

//                     list += l.length===0?"":" ";

//                     list += l;

//                     res.push(list);

//                 }

//             }

//         }

//         hash[str] = res;

//         return res;

//     }

   var hash = new Map();

   return helper(s);

   function helper(str) {

       if(hash.has(str)) {

           return hash.get(str);

       }

       var res = [];

       if(str.length==0) {

           res.push("");

           return res;

       }

       for(var word of wordDict) {

           if(str.startsWith(word)) {

               var substr = str.substring(word.length);

               var sublist = helper(substr);

               for(let l of sublist) {

                   var list = word;

                   list += l.length == 0 ? "":" ";

                   list += l;

                   res.push(list);

               }

           }

       }

       hash.set(str, res);

       return res;

   }

};

146. LRU Cache

/\*\*

\* @param {number} capacity

\*/

var LRUCache = function(capacity) {

   this.capacity = capacity;

   this.count = 0;

   this.hash = new Map();

   this.head = new Node(0,0); // fake head as boundary

   this.tail = new Node(0,0); // fake tail as boundary

   this.head.next = this.tail;

   this.tail.pre = this.head;

};

function Node(key, val) {

   this.key = key;

   this.val = val;

   this.pre = null;

   this.next = null;

}

LRUCache.prototype.deleteNode = function(node) {

   node.pre.next = node.next;

   node.next.pre = node.pre;

}

LRUCache.prototype.addHead = function(node) {

   node.next = this.head.next;

   node.next.pre = node;

   node.pre = this.head;

   this.head.next = node;

}

/\*\*

\* @param {number} key

\* @return {number}

\*/

LRUCache.prototype.get = function(key) {

   var res = -1;

   if(this.hash.has(key)) {

       var node = this.hash.get(key);

       res = node.val;

       this.deleteNode(node);

       this.addHead(node);

   }

   return res;

};

/\*\*

\* @param {number} key

\* @param {number} value

\* @return {void}

\*/

LRUCache.prototype.put = function(key, value) {

   if(this.hash.has(key)) {

       var node = this.hash.get(key);

       node.val = value;

       this.deleteNode(node);

       this.addHead(node);

   }

   else {

       var node = new Node(key, value);

       this.hash.set(key, node);

       this.addHead(node);

       this.count++;

       if(this.count>this.capacity) {

           this.hash.delete(this.tail.pre.key);

           this.deleteNode(this.tail.pre);

           this.count--;

       }

   }

};

/\*\*

\* Your LRUCache object will be instantiated and called as such:

\* var obj = Object.create(LRUCache).createNew(capacity)

\* var param\_1 = obj.get(key)

\* obj.put(key,value)

\*/

/\*\* version 2 \*\*/

/\*\*

\* @param {number} capacity

\*/

var LRUCache = function(capacity) {

   this.capacity = capacity;

   this.queue = []; //key queue

   this.map = new Map(); //key - val

};

function deleteNode(arr, key) {

   var idx = arr.indexOf(key);

   arr.splice(idx,1);

}

function addHead(arr, key) {

   arr.unshift(key);

}

/\*\*

\* @param {number} key

\* @return {number}

\*/

LRUCache.prototype.get = function(key) {

   if(this.map.has(key)) {

       deleteNode(this.queue,key);

       addHead(this.queue,key);

       if(this.queue.length>this.capacity) {

           var removekey = this.queue.pop();

           this.map.delete(removekey);

       }

       return this.map.get(key);

   }

   else {

       return -1;

   }

};

/\*\*

\* @param {number} key

\* @param {number} value

\* @return {void}

\*/

LRUCache.prototype.put = function(key, value) {

   if(this.map.has(key)) {

       deleteNode(this.queue,key);

       addHead(this.queue,key);

       this.map.set(key, value);

   }

   else {

       addHead(this.queue,key);

       this.map.set(key, value);

   }

   if(this.queue.length>this.capacity) {

       var removekey = this.queue.pop();

       this.map.delete(removekey);

   }

};

151. Reverse Words in a String

var reverseWords = function(str) {

   var arr = str.split(' ');

   arr = arr.filter(function(a) {

      return  a!=='';

   });

   return arr.reverse().join(' ').trim();

};

155. Min Stack

/\*\*

\* initialize your data structure here.

\*/

var MinStack = function() {

   this.stack = [];

   this.min = Number.MAX\_VALUE;

};

/\*\*

\* @param {number} x

\* @return {void}

\*/

MinStack.prototype.push = function(x) {

   if(x<=this.min) {

       this.stack.push(this.min);

       this.min = x;

   }

   this.stack.push(x);

};

/\*\*

\* @return {void}

\*/

MinStack.prototype.pop = function() {

   if(this.stack.pop()==this.min) {

       this.min = this.stack.pop();

   }

};

/\*\*

\* @return {number}

\*/

MinStack.prototype.top = function() {

   return this.stack[this.stack.length-1];

};

/\*\*

\* @return {number}

\*/

MinStack.prototype.getMin = function() {

   return this.min;

};

/\*\*

\* Your MinStack object will be instantiated and called as such:

\* var obj = Object.create(MinStack).createNew()

\* obj.push(x)

\* obj.pop()

\* var param\_3 = obj.top()

\* var param\_4 = obj.getMin()

\*/

161. One Edit Distance

var isOneEditDistance = function(s,t) {

 if(s==null||t==null||Math.abs(s.length-t.length)>1) return false;

 if(s==t) return false;

 var l1 = s.length;

 var l2 = t.length;

 for(let i=0; i<Math.min(l1,l2); i++) {

   if(s[i]!=t[i]) {

     if(l1==l2) return s.substring(i+1)==t.substring(i+1);

     else if(l1>l2) return s.substring(i+1)==t.substring(i);

     else return s.substring(i)==t.substring(i+1);

   }

 }

 return Math.abs(l1-l2)==1;

}

var s="123456abcdef";

var t="12345s6abcdef";

var res = isOneEditDistance(s,t);

console.log(res);

204. Count Primes

var countPrimes = function(n) {

   var count = 0;

   var notP = Array(n).fill(false);

   for(let i=2; i<n; i++) {

       if(notP[i]==false) {

           count++;

           for(let j=2; i\*j<n; j++) {

               notP[i\*j] = true;

           }

       }

   }

   return count;

};

206. Reverse Linked List

var reverseList = function(head) {

   //iteratively

   // if(head==null) return null;

   // var pre = null;

   // while(head!=null) {

   //     var next = head.next;

   //     head.next = pre;

   //     pre = head;

   //     head = next;

   // }

   // return pre;

   //recursively

   if(head==null||head.next==null) return head;

   var next = head.next;

   var cur = reverseList(next);

   next.next = head;

   head.next = null;

   return cur;

};

221. Maximal Square

var maximalSquare = function(matrix) {

   if(matrix==null||matrix.length==0||matrix[0].length==0) {

       return 0;

   }

   const m = matrix.length;

   const n = matrix[0].length;

   var res = 0;

   var dp = [];

   dp[0] = Array(n+1).fill(0);

   for(let i=1; i<=m; i++) {

       dp[i] = [];

       dp[i][0] = 0;

       for(let j=1; j<=n; j++) {

           if(matrix[i-1][j-1] == '1') {

               dp[i][j] = Math.min(dp[i-1][j], dp[i-1][j-1], dp[i][j-1])+1;

               res = Math.max(res, dp[i][j]);

           }else {

               dp[i][j] = 0;

           }

       }

   }

   return res\*res;

};

239. Sliding Window Maximum

var maxSlidingWindow = function(nums, k) {

   var res = [];

   if(nums==null||nums.length==0) {

       return res;

   }

   var queue = []; //contains index

   for(let i=0; i<nums.length; i++) {

       // remove numbers out of range k

       while(queue.length>0 && queue[0]<i+1-k) {

           queue.shift();

       }

       // remove smaller numbers in k range as they are useless

       while(queue.length>0 && nums[queue[queue.length-1]]<nums[i]) {

           queue.pop();

       }

       queue.push(i);

       if(i>=k-1) { //biggest keep in front

           res.push(nums[queue[0]]);

       }

   }

   return res;

};

253. Meeting Rooms II

function Interval(start, end) {

 this.start = start;

 this.end = end;

}

var minMeetingRooms = function(intervals) {

 var start = [];

 var end = [];

 for(let interval of intervals) {

   start.push(interval.start);

   end.push(interval.end);

 }

 start.sort(increase);

 end.sort(increase);

 var rooms = 0;

 var endIdx = 0;

 for(let i=0; i<start.length; i++) {

   if(start[i]<end[endIdx]) {

     rooms++;

   }else {

     endIdx++;

   }

 }

 return rooms;

 function increase(a,b) {

   return a-b;

 }

}

var in1 = new Interval(10,12);

var in2 = new Interval(12,14);

var in3 = new Interval(10,16);

var in4 = new Interval(15,16);

var in5 = new Interval(15,17);

var intervals = [in1,in2,in3,in4,in5];

var res = meetingRoomII(intervals);

console.log(res);

269. Alien Dictionary

var alienOrder = function(words) {

   //initail check

   if(words==null||words.length==0) {

       return "";

   }

   var map = new Map();

   var depth = new Map();

   for(let word of words) {

       for(let c of word) {

           if(!map.has(c)) {

               map.set(c,[]);

               depth.set(c,0);

           }

       }

   }

   // calculate the depth of each character

   for(let i=0; i<words.length-1; i++) {

       var cur = words[i];

       var next = words[i+1];

       var len = Math.min(cur.length, next.length);

       for(let j=0; j<len; j++) {

           //find not same character, the first one comes first

           if(cur[j]!==next[j]) {

               var nexts = map.get(cur[j]);

               nexts.push(next[j]);

               depth.set(next[j], depth.get(next[j])+1);

               break;

           }

       }

   }

   var queue = [];

   //get the first character

   for(let key of depth.keys()) {

       if(depth.get(key)==0) {

           queue.push(key);

       }

   }

   var res = "";

   while(queue.length>0) {

       var cur = queue.pop();

       res += cur;

       for(let next of map.get(cur)) {

           depth.set(next, depth.get(next)-1);

           if(depth.get(next)==0) {

               queue.push(next);

           }

       }

   }

   return res.length == map.size ? res:"";

};

var words = ["wrt","wrf","er","ett","rftt"];

var res = alienOrder(words);

console.log(res);

270. Closest Binary Search Tree Value

function TreeNode(val) {

   this.val = val;

   this.left = this.right = null;

}

//recursive

var closestBST = function(root, target) {

 var kid = target < root.val ? root.left:root.right;

 if(kid==null) return root.val;

 var closest = closestBST(kid, target);

 return Math.abs(closest-target) < Math.abs(root.val-target) ? closest:root.val;

}

//iterative

var closestBST2 = function(root, target) {

 var closest = root.val;

 while(root!==null) {

   closest = Math.abs(closest-target) < Math.abs(root.val-target) ? closest:root.val;

   root = target < root.val ? root.left: root.right;

 }

 return closest;

}

282. Expression Add Operators

var addOperators = function(num, target) {

   var res = [];

   if(num==null||num.length==0) {

       return res;

   }

   helper(res, "", 0, 0, 0);

   return res;

   function helper(res, list, start, val, mul) {

       if(start==num.length) {

           if(val==target) {

               res.push(list);

           }

           return;

       }

       for(let i=start; i<num.length; i++) {

           if(i!=start && num[start]=='0') break;

           var cur = parseInt(num.substring(start, i+1));

           if(start==0) {

               helper(res, list+cur, i+1, cur, cur);

           }

           else {

               helper(res, list+"+"+cur, i+1, val+cur, cur);

               helper(res, list+"-"+cur, i+1, val-cur, -cur);

               helper(res, list+"\*"+cur, i+1, val-mul+cur\*mul, cur\*mul);

           }

       }

   }

};

289. Game of Life

var gameOfLife = function(board) {

   var dir = [[-1,-1],[-1,0],[-1,1],[0,-1],[0,1],[1,-1],[1,0],[1,1]];

   for(let i=0; i<board.length; i++) {

       for(let j=0; j<board[0].length; j++) {

           var life = 0;

           for(let d of dir) {

               if(d[0]+i<0||d[0]+i>=board.length||d[1]+j<0||d[1]+j>board[0].length) {

                   continue;

               }

               if(board[d[0]+i][d[1]+j]==1 || board[d[0]+i][d[1]+j]==2) {

                   life++;

               }

           }

           if(board[i][j]==0 && life==3) { // reproduction

               board[i][j] = 3;

           }

           if(board[i][j]==1 && (life<2 || life>3)) { // die later

               board[i][j] = 2;

           }

       }

   }

   for(let i=0; i<board.length; i++) {

       for(let j=0; j<board[0].length; j++) {

           board[i][j] = board[i][j]%2;

       }

   }

};

// version 2

var gameOfLife = function(board) {

   if(board==null||board.length==0||board[0].length==0) {

       return;

   }

   var dir = [[-1,-1],[-1,0],[-1,1],[0,-1],[0,1],[1,-1],[1,0],[1,1]];

   for(let i=0; i<board.length; i++) {

       for(let j=0; j<board[0].length; j++) {

           getNext(i,j);

       }

   }

   for(let i=0; i<board.length; i++) {

       for(let j=0; j<board[i].length; j++) {

           board[i][j] %= 2;

       }

   }

   function getNext(x,y) {

       var count = 0;

       for(let d of dir) {

           if(x+d[0]<0 || x+d[0]>=board.length || y+d[1]<0 || y+d[1]>=board[0].length) {

               continue;

           }

           if(board[x+d[0]][y+d[1]]==1||board[x+d[0]][y+d[1]]==2) {

               count++;

           }

       }

       if(board[x][y]==0&&count==3) {

           board[x][y]=3;

       }

       if(board[x][y]==1&&(count<2 || count>3)) {

           board[x][y]=2;

       }

   }

};

312. Burst Balloons

var maxCoins = function(nums) {

   if (nums == null || nums.length == 0) return 0;

   var n = nums.length;

   var dp = [];

   for(let i=0; i<n; i++) {

       dp[i] = [];

       for(let j=0; j<n; j++) {

           dp[i].push(0);

       }

   }

   // var dp = new Array(n).fill(new Array(n).fill(0)); // fill the same obj, it's wrong

   for (var len = 1; len <= n; len++) {

       for (var start = 0; start <= n - len; start++) {

           var end = start + len - 1;

           for (var i = start; i <= end; i++) {

               var coins = nums[i] \* getValue(nums, start - 1) \* getValue(nums, end + 1);

               coins += i != start ? dp[start][i - 1] : 0; // If not first one, we can add subrange on its left.

               coins += i != end ? dp[i + 1][end] : 0; // If not last one, we can add subrange on its right

               dp[start][end] = Math.max(dp[start][end], coins);

           }

       }

   }

   return dp[0][n - 1];

   function getValue(nums, i) { // Deal with num[-1] and num[num.length]

       if (i < 0 || i >= nums.length) {

           return 1;

       }

       return nums[i];

   }

};

314. Binary Tree Vertical Order Traversal

function TreeNode(val) {

    this.val = val;

    this.left = this.right = null;

}

var verticalTraversal = function(root) {

 var res = [];

 if(root==null) {

   return res;

 }

 var map = new Map();

 var q = [];

 var cols = [];

 q.push(root);

 cols.push(0);

 var min = max = 0;

 while(q.length>0) {

   var node = q.shift();

   var col = cols.shift();

   if(!map.has(col)) {

     map.set(col,[]);

   }

   map.get(col).push(node.val);

   if(node.left!=null) {

     q.push(node.left);

     cols.push(col-1);

     min = Math.min(min, col-1);

   }

   if(node.right!=null) {

     q.push(node.right);

     cols.push(col+1);

     max = Math.max(max, col+1);

   }

 }

 for(let i=min; i<=max; i++) {

   res.push(map.get(i));

 }

 return res;

};

var root = new TreeNode(0);

var n1 = new TreeNode(1);

var n2 = new TreeNode(2);

var n3 = new TreeNode(3);

var n4 = new TreeNode(4);

var n5 = new TreeNode(5);

var n6 = new TreeNode(6);

root.left = n1;

root.right = n2;

n1.left = n3;

n2.left = n4;

n2.right = n5;

n4.right = n6;

var res = verticalTraversal(root);

console.log(res);

377.Combination Sum IV

var combinationSum4 = function(nums, target) {

   if(nums===null || nums.length===0) {

       return 0;

   }

   var hash = {};

   return combination(target);

   function combination(t) {

       if(t===0) {

           return 1;

       }

       if(t<0) {

           return 0;

       }

       var count = 0;

       if(hash[t]!==undefined) {

           return hash[t];

       }

       for(var n of nums) {

           count += combination(t-n);

       }

       hash[t] = count;

       return count;

   }

};

//dp

var combinationSum4 = function(nums, target) {

   var dp = Array(target+1).fill(0);

   dp[0] = 1;

   for(let i=1; i<target+1; i++) {

       for(let j=0; j<nums.length; j++) {

           if(i-nums[j]>=0) {

               dp[i] += dp[i-nums[j]];

           }

       }

   }

   return dp[target];

};

384. Shuffle an Array

var Solution = function(nums) {

   this.nums = nums;

   this.random = Math.random();

};

/\*\*

\* Resets the array to its original configuration and return it.

\* @return {number[]}

\*/

Solution.prototype.reset = function() {

   return this.nums;

};

/\*\*

\* Returns a random shuffling of the array.

\* @return {number[]}

\*/

Solution.prototype.shuffle = function() {

   var copy = this.nums.slice();

   for(let i=0; i<this.nums.length; i++) {

       var random = Math.floor(Math.random()\*(i+1));

       copy[i] = copy[random];

       copy[random] = this.nums[i];

   }

   return copy;

};

/\*\*

\* Your Solution object will be instantiated and called as such:

\* var obj = Object.create(Solution).createNew(nums)

\* var param\_1 = obj.reset()

\* var param\_2 = obj.shuffle()

\*/

401. Binary Watch

var readBinaryWatch = function(num) {

   var res = [];

   var hour = [1,2,4,8];

   var min = [1,2,4,8,16,32];

   for(let h=0; h<12; h++) {

       for(let min=0; min<60; min++) {

           if(countBit(h)+countBit(min)==num) {

               var m = min>=10 ? min:("0"+min);

               res.push(h+":"+m);

           }

       }

   }

   return res;

   function countBit(n) {

       var count = 0;

       while(n!==0) {

           if(n&1==1) count++;

           n = n>>1;

       }

       return count;

   }

};

402. Remove K Digits

var removeKdigits = function(num, k) {

   if(num==null||num.length==0||num.length==k) {

       return '0';

   }

   var stack = [];

   var i = 0;

   while(i<num.length) { // make res increasing order

       while(k>0 && stack.length>0 && stack[stack.length-1]>num[i]) {

           stack.pop();

           k--;

       }

       stack.push(num[i]);

       i++;

   }

   while(k>0) { // corner case like 1111...

       stack.pop();

       k--;

   }

   while(stack[0]=='0'&&stack.length>1) {

       stack.shift();

   }

   return stack.join('');

};

403. Frog Jump

var canCross = function(stones) {

   var map = new Map();

   map.set(0, new Set());

   map.get(0).add(1);

   for(let i=1; i<stones.length; i++) {

       map.set(stones[i], new Set());

   }

   for(let i=0; i<stones.length; i++) {

       var stone = stones[i];

       for(var step of map.get(stone)) {

           var reach = stone + step;

           if(reach==stones[stones.length-1]) return true;

           var set = map.get(reach);

           if(set!==undefined) {

               set.add(step);

               if(step>1) set.add(step-1);

               set.add(step+1);

           }

       }

   }

   return false;

};

/\*\*varsion 2\*\*/

var canCross = function(stones) {

   if(stones[1]!==1) return false;

   var map = new Map();

   map.set(0,new Set());

   map.get(0).add(1);

   for(let i=0; i<stones.length-1; i++) {

       var steps = map.get(stones[i]);

       if(!steps) return false; // the next one is not reached before

       for(let step of steps) {

           var reach = stones[i] + step;

           if(reach == stones[stones.length-1]) return true;

           if(!map.has(reach)) {

               map.set(reach,new Set());

           }

           var nextsteps = map.get(reach);

           if(step>1) nextsteps.add(step-1);

           nextsteps.add(step);

           nextsteps.add(step+1);

       }

   }

   return false;

};

438. Find All Anagrams in a String

var findAnagrams = function(s, p) {

   var abc = 'abcdefghijklmnopqrstuvwxyz';

   var chars = Array(26).fill(0);

   var result = [];

   if (s == null || p == null || s.length < p.length)

       return result;

   for (let c of p) {

       var idx = abc.indexOf(c);

       chars[idx]++;

   }

   var start = 0, end = 0, count = p.length;

   // Go over the string

   while (end < s.length) {

       // If the char at start appeared in p, we increase count

       if (end - start == p.length && chars[abc.indexOf(s[start++])]++ >= 0)

           count++;

       // If the char at end appeared in p (since it's not -1 after decreasing), we decrease count

       if (--chars[abc.indexOf(s[end++])] >= 0)

           count--;

       if (count == 0)

           result.push(start);

   }

   return result;

};

439. Ternary Expression Parser

var parseTernary = function(expression) {

 if(expression==null||expression.length==0) return "";

 var stack = [];

 for(let i=expression.length-1; i>=0; i--) {

   var c = expression[i];

   if(stack.length>0 && stack[stack.length-1]=='?') {

     stack.pop();

     var n1 = stack.pop();

     stack.pop();

     var n2 = stack.pop();

     if(c=='T') {

       stack.push(n1);

     }

     else {

       stack.push(n2);

     }

   }

   else {

     stack.push(c);

   }

 }

 return stack[0];

}

494. Target Sum

var findTargetSumWays = function(nums, S) {

//  TLE

//     var res = 0;

//     if(nums==null||nums.length==0) return res;

//     helper(0,0);

//     return res;

//     function helper(start,val) {

//         if(start==nums.length) {

//             if(val==S) res++;

//        return;

//         }

//         else {

//             helper(start+1, val+nums[start]);

//             helper(start+1, val-nums[start]);

//         }

//     }

   var res = 0;

   for(let i of nums) res+=i;

   if(S>res||S<-res) return 0;

   var dp = Array(res\*2+1).fill(0);

   dp[res] = 1;

   for(let i=0; i<nums.length; i++) {

       var next = Array(res\*2+1).fill(0);

       for(let j=0; j<2\*res+1; j++) {

           if(dp[j]!=0) {

               next[j+nums[i]] += dp[j];

               next[j-nums[i]] += dp[j];

           }

       }

       dp = next;

   }

   return dp[res+S];

};

527. Word Abbreviation

var wordsAbbreviation = function(words) {

 if(words==null||words.length==0) return words;

 var res = [];

 var prefix = [];

 for(let i=0; i<words.length; i++) {

   res[i] = makeAbbr(words[i],1);

   prefix[i] = 1;

 }

 for(let i=0; i<words.length; ) {

   var set = [];

   for(let j=i+1; j<words.length; j++) {

     if(res[i]==res[j]) {

       set.push(j);

     }

   }

   if(set.length==0) {

     i++;

   }else {

     set.push(i);

     for(let idx of set) {

       res[idx] = makeAbbr(words[idx],++prefix[idx]);

     }

   }

 }

 return res;

 function makeAbbr(s, prelen) {

   if(prelen>=s.length-2) return s;

   var res = s.substring(0,prelen);

   res += s.length-1-prelen;

   res += s[s.length-1];

   return res;

 }

}

var words = ["like", "god", "internal", "me", "internet", "interval", "intension", "face", "intrusion"];

var res = wordsAbbreviation(words);

635. Design Log Storage System

var LogSystem = function() {

   this.unit = ["Year","Month","Day","Hour","Minute","Second"];

   this.indices = [4,7,10,13,16,19];

   this.timestamps = new Map();

};

/\*\*

\* @param {number} id

\* @param {string} timestamp

\* @return {void}

\*/

LogSystem.prototype.put = function(id, timestamp) {

   this.timestamps.set(id, timestamp);

};

/\*\*

\* @param {string} s

\* @param {string} e

\* @param {string} gra

\* @return {number[]}

\*/

LogSystem.prototype.retrieve = function(s, e, gra) {

   var res = [];

   var idx = this.indices[this.unit.indexOf(gra)];

   s = s.substring(0, idx);

   e = e.substring(0, idx);

   for(let key of this.timestamps.keys()) {

       var val = this.timestamps.get(key).substring(0, idx);

       if(val>=s && val<=e) {

           res.push(key);

       }

   }

   return res;

};

/\*\*

\* Your LogSystem object will be instantiated and called as such:

\* var obj = Object.create(LogSystem).createNew()

\* obj.put(id,timestamp)

\* var param\_2 = obj.retrieve(s,e,gra)

\*/

Others

progress bar

Build a progress bar, then make three progress bars run in series. Then make five progress bars but two of them start staggered and when one completes start the next progress bar in series.

<https://codepen.io/smeng/pen/VWgaem>

var ele = document.getElementById('laoding-bar1');

var width = 0;

var id = 1;

var timeId = setInterval(load,10);

//load three loading bars in series

function load() {

 ele = document.getElementById('laoding-bar'+id);

 width = ele.style.width.split("%")[0];

 if(width>=100) {

   clearInterval(timeId);

   id++;

   if(id>3) {

     return;

   }

   timeId = setInterval(load,10);

 }

 else {

   width++;

   ele.style.width = width+"%";

 }

}

//load two loading bars in the same time

var ele4 = document.getElementById('laoding-bar4');

var ele5 = document.getElementById('laoding-bar5');

var width2 = 0;

var timeId2 = setInterval(load2,10);

function load2() {

 if(width2>=100) {

   clearInterval(timeId2);

 }

 else {

   width2++;

   ele4.style.width = width2+"%";

   ele5.style.width = width2+"%";

 }

}

字符去重排列

1。 如何给一个字符串去重后按字母表顺序排列输出。 2. 按反字母表顺序排列。

链接: <https://instant.1point3acres.com/thread/275831>

var str = "asdqwrewfegaedvzx";

function removeDup(str) {

var arr = str.split('');

 arr.sort();

 let j=0;

 for(let i=1; i<arr.length; i++) {

   if(arr[i]==arr[j]) {

      continue;

   }

   arr[++j] = arr[i];

 }

 var res = arr.join('').substring(0,j+1);

 return res;

}

function removeDupAndReverse(str) {

var arr = str.split('');

 arr.sort();

 let j=0;

 for(let i=1; i<arr.length; i++) {

  if(arr[i]==arr[j]) {

     continue;

   }

   arr[++j] = arr[i];

 }

 var res = arr.reverse().join('').substring(str.length-j-1);

 return res;

}

//if only abc no number

function removeDup2(str) {

var hash = [];

 var abc = 'abcdefghijklmnopqrstuvwxyz';

 for(let i=0; i<str.length; i++) {

  var idx = abc.indexOf(str[i]);

   hash[idx] = true;

 }

 var res = "";

 for(var i=0; i<26; i++) {

  if(hash[i]) {

   res += abc[i];

   }

 }

 return res;

}

var res = removeDup(str);

var res2 = removeDupAndReverse(str);

var res3 = removeDup2(str);

console.log(res);

console.log(res2);

console.log(res3);

第一题

给一个矩阵 ，对角线打印数字 比如

1   2   3    4   5

6   7   8    9  10

11 12 13 14  15

输出

1

26

37 11

48 12

59 13

1014

function printMatrix(nums) {

       var res = [];

       if (nums == null || nums.length == 0)

           return res;

       let rows = nums.length + nums[0].length - 1;

       for (let i = 0; i < rows; i++)

           res.push([]);

       for (let i = 0; i < nums.length; i++) {

           for (let j = 0; j < nums[0].length; j++) {

               res[ii + j].push(nums[i][j]);

           }

       }

       return res;

}

.1point3acres缃�

第二题

两个string一长T 一短S

find anagram substring of S in T in linear time

var findAnagrams = function(s, p) {

   var abc = 'abcdefghijklmnopqrstuvwxyz';

   var chars = Array(26).fill(0);

   var result = [];

   if (s == null || p == null || s.length < p.length)

       return result;

   for (let c of p) {

       var idx = abc.indexOf(c);

       chars[idx]++;

   }

   var start = 0, end = 0, count = p.length;

   // Go over the string

   while (end < s.length) {

       // If the char at start appeared in p, we increase count

       if (end - start == p.length && chars[abc.indexOf(s[start++])]++ >= 0)

           count++;

       // If the char at end appeared in p (since it's not -1 after decreasing), we decrease count

       if (--chars[abc.indexOf(s[end++])] >= 0)

           count--;

       if (count == 0)

           result.push(start);

   }

   return result;

};

Pairs of Amicable Numbers

int sumOfDiv(int x)

{

   // 1 is a proper divisor

   int sum = 1;

   for (int i=2; i<=sqrt(x); i++)

   {

       if (x%i == 0)

       {

           sum += i;

           // To handle perfect squares

           if (x/i != i)

               sum += x/i;

       }

   }

   return sum;

}

// Check if pair is amicable

bool isAmicable(int a,int b)

{

   return(sumOfDiv(a) == b && sumOfDiv(b) == a);

}

// This function prints pair of amicable pairs

// present in the input array

int countPairs(int arr[],int n)

{

   int count = 0;

   // Iterate through each pair, and find

   // if it an amicable pair

   for (int i=0; i<n; i++)

       for (int j=i+1; j<n; j++)

           if (isAmicable(arr[i], arr[j]))

               count++;

   return count;

}

bfs, dfs, topological sort

1。meeting room的题目， 给一串meeting room 时间，问是否能够schedule， 【1，2】， 【3，4】，【5，6】 2. followup 给一串meeting room 时间，需要多少个meeting room

链接: <https://instant.1point3acres.com/thread/273369>

boolean computable (int[] nums, int target) 栗子 ({1,2,3}, 6)=true 1+2+3=6 or 1\*2\*3=6 ({1,2, 2, 19}, 80)=true (1+19)\*(2+2)=80 ({1,2, 3}, 100)=false 可用+ \* () Array里的数可重复，但只能用一次，顺序随便 类似题 LC282， LC494 请问这题哪儿有？除了top-down暴力dfs (O(target!))还可以怎样? follow up 可用 - /

链接: <https://instant.1point3acres.com/thread/260315>

1. 跟里克扣68类似，不过改了细节，要跟面试官交流确定 2. 跟里克扣79类似，不过也是改了些细节 3. 跟里克扣239类似的题目 中午吃饭聊天， 加一轮design。 design主要是要多交流，确定需求等等，就把它当成实际工作的一个设计任务

链接: <https://instant.1point3acres.com/thread/272085>

设计并实现一下功能, Name CheckBox James X Aron Bob X ... void Toggle(String userName) List getActivatedUsers() boolean isChecked(String userName)

链接: <https://instant.1point3acres.com/thread/272058>

lintcode原题：

construct a binary tree from preorder and inorder traversal

在线测试本题：http://t.cn/RADxG4J

round 1

1)shuffle an array. 证明等概率

round 2

实现一个bloomfilter的数据结构，接口如下

class Bloomfilter {

private Key[] table;

private List hashFunc;

public boolean contains(Key k) {}

public void put(Key k) {}

public void remove(Key k) {}

}

该数据结构类似于HashSet，put函数存key进去，contains查询是否存在，remove删除对应的key。它提供了一组hash函数列表hashFunc。

question:

1)该数据结构有个特点，contains函数如果return false，那key一定不存在Bloomfilter内，但如果return true，则可能存在，可能不存在，为什么？

2)实现这个数据结构的三个函数

3)这个数据结构有什么缺点？怎么Scale？写出代码

round 3

1)随便讲一个知识点，assume面试官没有任何相关背景，五分钟讲清楚。

2)Make friends: 有n个人，每个人要交且只交k个朋友。输入n和k，输出一种交朋友的情况。

round 4

面试官是4个组的director，问了很多behavioral question，如说一个以前的mistake，说一个最喜欢的做过的项目。

lintcode原题word break，输出一种分割即可。 (在线测试本题：http://t.cn/RADxMwk)

follow up: 英语中，几个字母组成单词的概率远小于不是单词的概率，怎么优化？

问题是让设计一个Address Class， 然后给定几个随机的Address， 写出一个function 看这个Address是不是在各自附近。。。。还是挺简单的一道题，花了不少时间沟通设计的要求，其实把要求都问好了，不难写。 电面完当天直接接到电话通知Onsite， 两周后面的。 4轮technical ＋ 1顿饭。  第一轮： 给定一个2D char array和一个String input, 看这个input在Array里面是不是存在。每个character必须挨着前一个character，并且不能重复使用。 花了点时间才了解面试官的全部要求， DFS ＋ Back Tracking, 秒 第二轮： 让设计一个 Pokemon Go......................    你懂的，没什么coding，时间全部在画Diagram上 午饭 第三轮：国人小哥， 让写一个function把compact json print成pretty version。 题很简单，但是吃完午饭比较困，脑子基本一团酱， 而且基本功不扎实真的是我的错。。。写出来有bug,改的很崩溃，小哥说算了， 别改了，知道你会写，别写了。。。。。。。

最后一轮： 也是亚洲小哥，估计是急着回家，让设计 ＋ Implement一个Cache， 不管什么Eviction Policy. 写了一个LRU with TTL. 但是时间真的是不够，全是bug.

链接: <https://instant.1point3acres.com/thread/271412>

第一轮：求图的最短距离，不过难点在于把给的2维grid转化成图，再求图里的最短距离。因为转换没写，所以有点卡，坑此坑次地把bfs写了。转换的代码没写完，感觉第一轮就gg了。 中饭是一位nice的国人女同胞带去的。然后各种聊公司的事情。然后发现，第一轮我竟然把面试官跟shadow搞反了。感觉真要gg了。 第二轮：各种聊bq，然后ood，设计board game，有monster有英雄，要求设计游戏里的各种类。基本就是用户控制英雄走上下左右，碰到怪物就死掉，走到终点就赢。cc189有类似的题。code写完了。 第三轮：infrastructure debug，说如果发现production有问题了怎么做。然后来了一条string processing的问题。给一条string要求shuffle，然后输出的string里不能有两个连续的字母。例如：input aabbb， return babab， input aabbbb return null。用hashmap数了频率，然后还得按值排序，各种复杂，只写好了code，没测试。把源码发给面试官等候发落。 第四轮：一个在公司since beginning的engineer，system design，team specific的问题。就是bitmoji怎么通过输入text来filter返回什么。然后还得讨论request从手机发出去以后到server到database的过程。

链接: <https://instant.1point3acres.com/thread/265529>

System Design

System design 1) 怎么分类垃圾邮件 2) 实现snap的一个feature 本质上是OOD 这一轮聊得很开心， 感觉面试官很赞同我的

链接: <https://instant.1point3acres.com/thread/204965>

来源: 一亩三分地

// snapchatter  
// toggle(string username)   
// getSelectedLlist()   
// t: a getsl: a  
// t: a b c getSl: a b c  
// t: a b a c getSl: b c  
Design题我给的设计，用了doublelinkedlist，确保了toggle时的时间复 杂度为O(1)

Design: 然后问如果有一个web based的email app，打开的latency很高，应该怎么办，如何test哪个部分，如果Improve. Waral 如果有准备过what happened after type an URL，因为没有太大问题，无非就是按照那个过程，DNS, proxy server, cache(这是重点)

然后一上来就出了一道twitter系统的设计，幸亏我之前准备过system design，而且在其他面经上也看到别人提到过这个，然后给他讲了讲各个方面的设计。 比如消息推送的机制，push和poll。还有怎么样用hybrid方法。 然后他笑了，说对对对，我们现在snapchat就是这么干的。这个题好像答得还挺好的。聊的也不错。 然后我以为就结束了，但是还有10分钟的时间。他就说我们来coding吧，请听题，“树上一只猴，地下七只猴。。。”，串了串了。。。然后让我coding一个单机的web 服务器的消息处理机制。 我先说如果是单机的话，就不用考虑scale的问题。说了好多乱七八糟的system design的问题，但是他不是这个意思。后来搞了两三分钟才讲清楚是要实现一个类。说当访问量大的时候，怎么handle request，我说线程池。 他说yes， that's what i want。 然后其实就是实现线程池类。我不知道是怎么实现的，自己想了一个方法。然后说了思路，但是还没把代码写完，第四个面试官就开始用木头桩子撞门，额，轻轻地敲门。然后中东小哥说再给5min，第四个面试官说ok。然后这个题关键的是要用到wait和notify方法以避免繁忙等待。 我说了思路，最后还是没有时间写代码了。弄完他问我有没有什么问题，我说我没有写完这个代码会不会有影响，他说不会，反正solution你已经知道了。然后说完握手扬长而去。留下一个华丽的身影。

搜push 和poll机制，然后到时候被问到了，记得考虑两种结合起来，也就是hybrid的思想。

给手机的画图app写个屏幕旋转的method。同样问了许多design问题。

设计一个比较简单的Google doc。同样是design向的问题

给我看了一个snapchat给好友群发消息的功能，可以任意选中和删除想要投递消息的好友，并显示群发好友的list(按先后选中的顺序)，设计一个数据结构，实现 toggle(String username); getList(); LZ给的hashmap + doubly linked list的设计，类比LRU cache, 详细解释了一下，分析了时间复杂度。小哥说是最优解了，不coding

设计题，一些snapchat功能的设计，如何pre-load的strategy。题目很细节，我一开始完全木有概念，但是她给了很多信息，所以一点点也就答出来了了。然后就是很多聊天，聊得比较开心。。。

你需要design一个system，这个system可以用来evaluate各种我们设计的朋友推荐算法，你要怎么设计。说了一下大概的设计，面试官更看重我如何准备data来做evalution，

怎样test保证能在product work，扯了下unit test + modular test + system test

就是给了一个Person类，让实现打出这个Person的关系网中的所有朋友，Person有属性id, name和friendList, sample输出是这样子的：

/\* \* Mike(1) {Mitch, Ryan}   
\* Mitch(2) {Mike}   
\* Ryan(3) {Mike, Lila}   
\* Lila(4) {Ryan}   
\* Mike.printFriendsGraph() : Mitch, Ryan, Lila \*/

snap with wights, snap has next snaps list

input是start snap, max steps求出一个max total weights

BFS的方法做就可以.

follow up: start snap是一个list怎么办

follow up: 如果每个snap都可以是start nap怎么办， memorization search

(1) LRU (2) IP address filter，给一些ip的规则，然后问那些ip复合这些规则 (3) word pattern (其他面经里有) (4) 1. valid BST 2. 添加括号使得表达式值最大，表达式只有+ - \*