OA:.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
90min 2题 ＋ culture survey  
第一题 棒球题 就是数字、X、Z、＋  
注意：如果是非法输入 应该return 0 而不是return－1 这里耽误很多时间  
第二题 构建BST 然后找两个node之间距离 因为是BST 所以挺简单的 别当做普通binary tree 不然的话 我猜应该会超时  
test case：  
第一题 11/11  
第二题 14/14  
. 1point 3acres 璁哄潧  
  
**补充内容 (2017-4-3 12:08):**  
总结了一下 其实社招OA只有七道题  
1. 验证括号 LC 原题  
2. Anagram LC 原题. more info on [1point3acres.com](http://1point3acres.com/)  
3. 迷宫题 LC 略有修改  
4. 棒球题 （我做的第一题）  
5. BST题（我做的第二题）  
6. Movie Network  
7. Common Manager

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=225078&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26searchoption%255B3089%255D%255Bvalue%255D%255B5%255D%3D5%26searchoption%255B3089%255D%255Btype%255D%3Dcheckbox%26searchoption%255B3046%255D%255Bvalue%255D%3D5%26searchoption%255B3046%255D%255Btype%255D%3Dradio&page=1>

Q1： 丢棒球，输入一个字符串，其中包括整数，Z，X，或者+。整数代表此轮得分，X：当前成绩是double前面一个分数，+：当前成绩是前两个的和，Z：移除前一个成绩，求最后的总成绩和  
  
一颗栗子： 输入["5", "-2", "4", "Z","X", 9, "+", "+"]  
output: 27  
5 : sum = 5  
-2 : sum = 5 - 2 = 3. [1point3acres.com/bbs](http://1point3acres.com/bbs)  
4 : sum = 3 + 4 = 7.1point3acres缃�  
Z : sum = 7 - 4 = 3.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
X : sum = 3 + -2 \* 2 = -1 (4被移除了，前一个成绩是-2)  
9 : sum = -1 + 9 = 8  
+ : sum = 8 + 9 - 4 = 13 (前两个成绩是9和-4)  
+ : sum = 13 + 9 + 5 = 27 (前两个成绩是5 和 9)  
用一个stack解决。  
  
  
Q2：movie network假设有个Movie类，. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)   
public class Movie  
{. [1point3acres.com/bbs](http://1point3acres.com/bbs)  
   int movieId;.  
   float rating;  
   List<Movie> similarMoviesoint3acres缃�  
}.more info on [1point3acres.com](http://1point3acres.com/)-google 1point3acres  
要求找和movie相似的电影中排名前k个的电影（不包括当前movie）。就是找movie的所有neighbor中排名前k的电影。地里有很多关于这道题的讨论，就不多说了。

<http://www.1point3acres.com/bbs/thread-276519-1-1.html>

<http://www.1point3acres.com/bbs/thread-278634-1-1.html>

<http://www.1point3acres.com/bbs/thread-210746-1-1.html>

1. Employee\* findCommonManager(Employee\* ceo,  Employee\* employee1, Employee\* employee2) {. 鐣欏鐢宠璁哄潧-涓€浜╀笁鍒嗗湴
2. if (ceo == nullptr) {
3. return nullptr;
4. }
5. if (ceo == employee1) {
6. return employee1;
7. }
8. if (ceo == employee2) {
9. return employee2;
10. }. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)
11. bool isManagingEmployee1 = false;. Waral 鍗氬鏈夋洿澶氭枃绔�,
12. bool isManagingEmployee2 = false;
13. for (int i = 0; i < ceo->getReports().size(); i++) {
14. Employee\* e = findCommonManager(ceo->getReports()[i], employee1, employee2);
15. if (e == employee1) {
16. isManagingEmployee1 = true;
17. } else if (e == employee2) {
18. isManagingEmployee2 = true;
19. } else if (e != nullptr) {
20. return e;
21. }
22. }
23. if (isManagingEmployee1 && isManagingEmployee2) {
24. return ceo;
25. }. visit [1point3acres.com](http://1point3acres.com/) for more.
26. if (isManagingEmployee1) {
27. return employee1;
28. }
29. if (isManagingEmployee2) {
30. return employee2;
31. }
32. return nullptr;.鏈枃鍘熷垱鑷�1point3acres璁哄潧
33. }

1. Johns play a game in which he throws a baseball at various blocks marked with a symbol so as to  knock these out. A score is computed for each throw. The 'last score' is the score of the previous throw (or 0 if there is no previous throw) and the total score is the sum of the scores of all the throws. The symbol on a block can be an integer, a sign or a letter. Each sign or letter represents a special rule as given below.. visit [1point3acres.com](http://1point3acres.com/) for more.  
  
if a throw hits a block marked with an integer, the score for that throw is the value of that integer.  
if a throw hits a block marked with an 'X', the score for that throw is double the last score  
if a throw hits a block marked with an '+', the score for that throw is the sum of the last two scores.  
if a throw hits a block marked with an 'Z', the last score is removed, as though the last throw never happened. Its value doesn't count towards the total score, and the subsequent throws will ingnore it when computing their values.

第一题  
给一个数组A[5，6，3，1，2，4]，先建立BST，然后搜索两个node之间的距离。  
已知第一个元素5是root，剩下的是无序的！！注意后面有可能是右子树先出现（6，对应root-5），也有可能是左子树先出现（1，2，对应root-3）！. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
如果有node不存在的话，返回-1；  
要求实现的函数长这样：int distance（int\* values, int n，int node1, int node2）. more info on [1point3acres.com](http://1point3acres.com/)  
注：题目中没说，但是好像数组里应该没有重复元素。  
2个给好的tests，9个隐藏的。  
Recursion建树，while loop找距离，test都过了，其他方法不清楚会不会有TLE。  
  
第二题  
给一部电影，要求返回跟这部电影相关的，排名最高的N部电影，其中不包括输入的那部电影！！  
输出不需要排序。如果不够N部，就有多少输出多少部。  
电影长这样：  
class Movie { 鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�.   
  private:  
    int id; float rate; vector<Movie\*> similarMovies;  
  public:  
    int getID();  
    float getRate();. From 1point 3acres bbs  
    vector<Movie\*>& getSimilar();  
}  
要求实现的函数长这样：  
vector<Movie\*> find(Movie& movie, int N)  
2个给好的tests，10个隐藏的。  
BFS搜索所有电影，max heap挑选结果，这样做test可以都过，不清楚其他方法会不会有TLE。

// find distance between two node of BST

// http://algorithms.tutorialhorizon.com/find-the-distance-between-two-nodes-of-a-binary-tree/ （find distance between two node of Binary Tree）

**public** **static** TreeNode LCABST(TreeNode root, TreeNode p, TreeNode q) {

**if**(root == **null** || p == root || q == root) **return** root;

**if**((root.val > p.val && root.val < q.val) ||

(root.val < p.val && root.val > q.val)) {

**return** root;

} **else** **if**(root.val > p.val && root.val > q.val) {

**return** *LCABST*(root.left, p, q);

} **else** **if**(root.val < p.val && root.val < q.val) {

**return** *LCABST*(root.right, p, q);

} **else** {

**return** **null**;

}

}

**public** **static** **int** findlen(TreeNode root, **int** n) {

**return** *finddistancebst*(root, n)-1;

}

**public** **static** **int** finddistancebst(TreeNode root, **int** n) {

**if**(root == **null**) **return** 0;

**int** x = 0;

**if**(root.val == n) **return** x+1;

**if**(root.val > n) {

x = *finddistancebst*(root.left, n);

} **else** {

x = *finddistancebst*(root.right, n);

}

**if**(x > 0)

**return** x+1;

**else** **return** 0;

}

发一个亚麻社招 OA 的面经，OA是在amcat上做的，包括75 min Coding + 15 min Culture + 5 min Feedback，其实开始做coding之后时间有90min  
Q1 丢棒球砸砖块，貌似是地里没有出现过的题，输入是一个字符串数组，每一个值可能是一个整数，或者Z，或者X，或者+。整数代表现在拿的分，X代表当前成绩是前一个分数Double，+代表当前成绩是前两个的和，Z代表移除前一个成绩，然后要求的是最后的总成绩  
例子： 输入 ["5", "-2", "4", "Z", "X", 9, "+", "+"]  
输出 27  
5 : sum = 5  
-2 : sum = 5 - 2 = 3  
4 : sum = 3 + 4 = 7. 鍥磋鎴戜滑@1point 3 acres  
Z : sum = 7 - 3 = 4  
X : sum = 3 + -2 \* 2 = -1 (因为4被移除了，前一个成绩是-2)  
9 : sum = -1 + 9 = 8  
+ : sum = 8 + 9 - 4 = 13 (前两个成绩是9和-4)  
+ : sum = 13 + 9 + 5 = 27 (前两个成绩是5 和 9)  
  
用一个stack就可以秒掉  
  
Q2: Movie network，是地里出现过的原题，BFS + PriorityQueue秒  
. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
再附上整理的地里出现过的新版Amcat 社招OA面经，希望能帮助到大家，攒人品，

1. Employee\* findCommonManager(Employee\* ceo,  Employee\* employee1, Employee\* employee2) {. 鐣欏鐢宠璁哄潧-涓€浜╀笁鍒嗗湴
2. if (ceo == nullptr) {
3. return nullptr;
4. }
5. if (ceo == employee1) {
6. return employee1;
7. }
8. if (ceo == employee2) {
9. return employee2;
10. }. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)
11. bool isManagingEmployee1 = false;. Waral 鍗氬鏈夋洿澶氭枃绔�,
12. bool isManagingEmployee2 = false;
13. for (int i = 0; i < ceo->getReports().size(); i++) {
14. Employee\* e = findCommonManager(ceo->getReports()[i], employee1, employee2);
15. if (e == employee1) {
16. isManagingEmployee1 = true;
17. } else if (e == employee2) {
18. isManagingEmployee2 = true;
19. } else if (e != nullptr) {
20. return e;
21. }
22. }
23. if (isManagingEmployee1 && isManagingEmployee2) {
24. return ceo;
25. }. visit [1point3acres.com](http://1point3acres.com/) for more.
26. if (isManagingEmployee1) {
27. return employee1;
28. }
29. if (isManagingEmployee2) {
30. return employee2;
31. }
32. return nullptr;.鏈枃鍘熷垱鑷�1point3acres璁哄潧
33. }

1.打棒球得分，给了一个String[] input，求最终score.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
   如果是 integer， 就加给score（有负值）  
   如果是“x”, 将上一个值double ，加给score； 若没有上一个值，上一个值按0计算. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
   如果是“z”, 上一个成绩作废， score剪掉上一值  
   如果是“+”，将上两个值相加，然后加给score  
  
  我的解法是用一个stack挨个处理。  
  麻烦的是，input是个string[]; 每一个我都是用 string*.charAt(0) == 'x'来处理的；然后string转int调用了Integer.parseInt(string);. From 1point 3acres bbs  
            题目我看了半天，stack的操作应该是 z直接pop；   
                                         x先pop，然后再将double的值放进去；   
                                        +是先pop出来两个值，加给score后，在按原样放回去，并把他俩的和也放进去。  
   我贴的link 上有一个例子，大家可以过一下，我写的很是头疼。  
  
2. 给一个无序integer array，要求用insert的方法建一个BST，然后给出其中两个值在树上的距离，若是有不在树里的，返回-1  
    signature 大概这样 public  int bstDistance(int[] values, int n, int node1, int node2) n是数组里值的个数  
   我的解法是：先用insert建立BST， 然后找到Lowest Common Ancensor, 然后找node1 node2 和LCA的高度（BST的search法 iterative找），  
   返回 node1- LCA + node2 -LCA;  
   注意一下 记得使用n这个参数，还有helper方程的输入类型 ，LCA里传入的是TreeNode root， int node1,int node2; 然后看一下实在LCA还是GetHeight方程里处理找不到node的情况。  
    test case里11个过了9个，估计是边界corner case有问题。*

、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

//3. Find the min steps of maze

//意思是说有一个M\*N的maze，0代表可以通过，1代表不可以通过，然后给你一个出口（x,y），找从（0,0）到出口的最少steps，如果找不到path就返回-1

**static** **int** *minlen* = Integer.***MAX\_VALUE***;

**public** **static** **void** bfsfind(**int**[][] mazeinit) {

**if**(mazeinit == **null** || mazeinit.length == 0) **return**;

**int** row = mazeinit.length;

**int** col = mazeinit[0].length;

**boolean**[][] visited = **new** **boolean**[row][col];

**int**[] direcx = {-1,0,1,0};

**int**[] direcy = {0,-1,0,1};

*bfs*(mazeinit, visited, 0, 0, 0, direcx, direcy);

**return**;

}

**public** **static** **void** bfs(**int**[][] mazeinit, **boolean**[][] visited, **int** row, **int** col, **int** pathlen, **int**[] direcx, **int**[] direcy) {

**if**(!*isValid*(mazeinit, visited, row, col)) **return**;

**if**(row == mazeinit.length-1 && col == mazeinit[0].length-1) {

*minlen* = Math.*min*(*minlen*, pathlen);

**return**;

}

**for**(**int** i = 0; i < 4; i++) {

visited[row][col] = **true**;

*bfs*(mazeinit, visited, row+direcx[i], col+direcy[i], pathlen+1, direcx, direcy);

visited[row][col] = **false**;

}

}

**public** **static** **boolean** isValid(**int**[][] mazeinit, **boolean**[][] visited, **int** row, **int** col) {

**int** m = mazeinit.length;

**int** n = mazeinit[0].length;

**return** row >=0 && row < m && col >= 0 && col < n && mazeinit[row][col] == 0 && visited[row][col] == **false**;

}

**public** **static** **int** bdfsfind(**int**[][] mazeinit, Point src, Point tar) {

**if**(mazeinit == **null** || mazeinit.length == 0) **return** 0;

**int** row = mazeinit.length;

**int** col = mazeinit[0].length;

**boolean**[][] visited = **new** **boolean**[row][col];

**if**( !*isBfsValid*(mazeinit, visited, src.x, src.y) ) **return** 0;

**if**( !*isBfsValid*(mazeinit, visited, tar.x, tar.y) ) **return** 0;

**int**[] direcx = {-1,0,1,0};

**int**[] direcy = {0,-1,0,1};

PointNode root = **new** PointNode(**new** Point(src.x, src.y), 0);

visited[src.x][src.y] = **true**;

Queue<PointNode> queue = **new** LinkedList<PointNode>();

queue.add(root);

**while**(!queue.isEmpty()) {

**int** size = queue.size();

**for**(**int** i = 0; i < size; i++) {

PointNode temp = queue.poll();

//find a matched solution

**if**(temp.pt.x == tar.x && temp.pt.y == tar.y) **return** temp.dist;

**for**(**int** j = 0; j < 4; j++) {

// tranverse point is valid

**if**(*isBfsValid*(mazeinit, visited, temp.pt.x+direcx[j], temp.pt.y+direcy[j])) {

**int** rowtemp = temp.pt.x+direcx[j];

**int** coltemp = temp.pt.y+direcy[j];

visited[rowtemp][coltemp] = **true**;

PointNode qadd = **new** PointNode(**new** Point(rowtemp, coltemp), temp.dist+1);

queue.offer(qadd);

}

}

}

}

**return** 0;

}

**public** **static** **boolean** isBfsValid(**int**[][] mazeinit, **boolean**[][] visited, **int** row, **int** col) {

**int** m = mazeinit.length;

**int** n = mazeinit[0].length;

**return** row >= 0 && row < m && col >= 0 && col < n

&& mazeinit[row][col] == 0 && visited[row][col] == **false**;

}

**class** Point {

**int** x;

**int** y;

Point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

}

**class** PointNode {

Point pt;

**int** dist;

PointNode(Point pt, **int** dist) {

//this.x = x;

**this**.pt = pt;

**this**.dist = dist;

}

}

、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

// ball score

**public** **static** **int** ballscore(String[] strarr) {

**int** score = 0;

**if**(strarr == **null** || strarr.length == 0) **return** score;

Stack<Integer> stack = **new** Stack<Integer>();

**for**(**int** i = 0; i < strarr.length; i++) {

**char** ch = strarr[i].charAt(0);

**if**(ch != 'X' && ch != '+' && ch != 'Z') {

**int** ps = Integer.*parseInt*(strarr[i]);

stack.push(ps);

score += ps;

} **else** {

**switch**(ch) {

**case** 'Z' :

**if**(!stack.isEmpty()) {

**int** curr = stack.pop();

score -= curr;

}

**break**;

**case** 'X' :

**if**(!stack.isEmpty()) {

**int** curr = stack.peek();

score += 2\*curr;

stack.push(2\*curr);

}

**break**;

**case** '+' :

**if**(!stack.isEmpty()) {

**int** i1 = stack.pop();

**if**(!stack.isEmpty()) {

**int** i2 = stack.pop();

score += i1;

score += i2;

stack.push(i2);

stack.push(i1);

stack.push(i1+i2);

} **else** {

score += i1;

stack.push(i1);

stack.push(i1);

}

}

**break**;

**default** :

**break**;

} // end switch

} // end first if-else

} // end for

**return** score;

}

、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

// find distance between two node of BST

// http://algorithms.tutorialhorizon.com/find-the-distance-between-two-nodes-of-a-binary-tree/ （find distance between two node of Binary Tree）

**public** **static** TreeNode LCABST(TreeNode root, TreeNode p, TreeNode q) {

**if**(root == **null** || p == root || q == root) **return** root;

**if**((root.val > p.val && root.val < q.val) ||

(root.val < p.val && root.val > q.val)) {

**return** root;

} **else** **if**(root.val > p.val && root.val > q.val) {

**return** *LCABST*(root.left, p, q);

} **else** **if**(root.val < p.val && root.val < q.val) {

**return** *LCABST*(root.right, p, q);

} **else** {

**return** **null**;

}

}

**public** **static** **int** findlen(TreeNode root, **int** n) {

**return** *finddistancebst*(root, n)-1;

}

**public** **static** **int** finddistancebst(TreeNode root, **int** n) {

**if**(root == **null**) **return** 0;

**int** x = 0;

**if**(root.val == n) **return** x+1;

**if**(root.val > n) {

x = *finddistancebst*(root.left, n);

} **else** {

x = *finddistancebst*(root.right, n);

}

**if**(x > 0)

**return** x+1;

**else** **return** 0;

}

**public** **static** **int** bstdistance(**int**[] values, **int** n, **int** node1, **int** node2) {

// create the tree use values array and n

**if**(values == **null** || values.length == 0) **return** 0;

TreeNode root = **new** TreeNode(values[0]);

**for**(**int** i = 1; i < values.length; i++) {

*createbst*(root, values[i]);

}

//TreeNode root = createbst(values); // this is a func

**int** len1 = *findlen*(root, node1);

**if**(len1==-1) **return** 0;

**int** len2 = *findlen*(root, node2);

**if**(len2==-1) **return** 0;

**int** lcaval = *LCABST1*(root, node1, node2).val;

**int** lenmid = *findlen*(root, lcaval);

**return** len1+len2-2\*lenmid;

}

**public** **static** **void** createbst(TreeNode root, **int** value) {

**if**(value < root.val) {

**if**(root.left == **null**) {

root.left = **new** TreeNode(value);

} **else** {

*createbst*(root.left, value);

}

} **else** {

**if**(root.right == **null**) {

root.right = **new** TreeNode(value);

} **else** {

*createbst*(root.right, value);

}

}

}

**public** **static** TreeNode LCABST1(TreeNode root, **int** i1, **int** i2) {

**if**(root == **null** || i1 == root.val || i2 == root.val) **return** root;

**if**((root.val > i1 && root.val < i2) ||

(root.val < i1 && root.val > i2)) {

**return** root;

} **else** **if**(root.val > i1 && root.val > i2) {

**return** *LCABST1*(root.left, i1, i2);

} **else** **if**(root.val < i1 && root.val < i2) {

**return** *LCABST1*(root.right, i1, i2);

} **else** {

**return** **null**;

}

}

、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

// LCA of n-ary tree

//Employee firstEmployee = ceo.getReports().get(0).getReports().get(0).getReports().get(0).getReports().get(0);

//Employee secondEmployee = ceo.getReports().get(0).getReports().get(0).getReports().get(0).getReports().get(1);

**public** **static** Employee commonmanager(Employee ceo, Employee firstEmployee, Employee secondEmployee) {

Stack<Employee> first = **new** Stack<Employee>();

Stack<Employee> second = **new** Stack<Employee>();

Employee root = ceo;

*dfs*(root, firstEmployee, first);

*dfs*(root, secondEmployee, second);

**if**( (!first.isEmpty() && first.peek().getId() == firstEmployee.getId())

&& (!second.isEmpty() && second.peek().getId() == secondEmployee.getId())) {

**int** size1 = first.size();

**int** size2 = second.size();

**int** diff = Math.*abs*(size1-size2);

**if**(size1 > size2) {

// first stack pop element until size equals the second

*moveUp*(first, diff);

} **else** {

*moveUp*(second, diff);

}

**while**(first.peek().getId() != second.peek().getId()) {

first.pop();

second.pop();

}

**if**(first.size() > 0) {

**return** first.pop();

}

}

**return** **null**;

}

**public** **static** **void** moveUp(Stack<Employee> stack, **int** diff) {

**while**(diff > 0 && !stack.isEmpty()) {

stack.pop();

diff--;

}

}

**public** **static** **boolean** dfs(Employee root, Employee curr, Stack<Employee> stack) {

stack.push(root);

**if**(root.getId() == curr.getId()) {

**return** **true**;

}

**for**(Employee em : root.getReports()) {

**boolean** res = *dfs*(em, curr, stack);

**if**(res == **true**) {

**return** **true**;

}

}

stack.pop();

**return** **false**;

}

**public** **static** Employee commonmanageroftree(Employee ceo, Employee firstEmployee, Employee secondEmployee) {

**if**(ceo==**null**) **return** ceo;

**if**(ceo==firstEmployee) **return** ceo;

**if**(ceo==secondEmployee) **return** ceo;

**boolean** isEmployee1 = **false**;

**boolean** isEmployee2 = **false**;

**for**(Employee em : ceo.getReports()) {

Employee res = *commonmanageroftree*(em, firstEmployee, secondEmployee);

**if**(res == firstEmployee) {

isEmployee1 = **true**;

} **else** **if**(res == secondEmployee) {

isEmployee2 = **true**;

} **else** **if**(res != **null**) {

**return** res;

}

}

**if**(isEmployee1==**true** && isEmployee2==**true**) {

**return** ceo;

} **else** **if**(isEmployee1==**true**) {

**return** firstEmployee;

} **else** **if**(isEmployee2==**true**) {

**return** secondEmployee;

}

**return** **null**;

}

、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

private static void bfsSearchMovieNetwork(Movie movie, PriorityQueue<Movie> queue) {  
        for (Movie m : movie.getSimilarMovies()) {  
            if (!queue.contains(m)) {  
                queue.offer(m);.

                bfsSearchMovieNetwork(m, queue);  
            }  
        }  
    }

PriorityQueue<Movie> queue = new PriorityQueue<>(new Comparator<Movie>() {  
            @Override  
            public int compare(Movie m1, Movie m2) {  
                return new Float(m2.getRating()).compareTo(m1.getRating());  
            }  
        });

private static void bfsSearchMovieNetwork(Movie movie, PriorityQueue<Movie> queue) {  
        Queue que = new LinkedList;  
        que.offer(movie);  
        while(!que.isEmpty()) {  
               int size = que.size();  
               for(int i = 0; i < size; i++) {  
                      Movie temp = que.poll();  
                      for (Movie m : temp.getSimilarMovies()) {  
                               if(queue.size() < k) {  
                                       queue.offer(m);  
                                } else if(queue.peek().ratings < m.ratings) {  
                                       queue.poll();  
                                        queue.offer(m);  
                                }  
                      }  
               }  
        }  
    }

Johns play a game in which he throws a baseball at various blocks marked with a symbol so as to  knock these out. A score is computed for each throw. The 'last score' is the score of the previous throw (or 0 if there is no previous throw) and the total score is the sum of the scores of all the throws. The symbol on a block can be an integer, a sign or a letter. Each sign or letter represents a special rule as given below.  
  
if a throw hits a block marked with an integer, the score for that throw is the value of that integer.  
if a throw hits a block marked with an 'X', the score for that throw is double the last score  
if a throw hits a block marked with an '+', the score for that throw is the sum of the last two scores.  
if a throw hits a block marked with an 'Z', the last score is removed, as though the last throw never happened. Its value doesn't count towards the total score, and the subsequent throws will ingnore it when computing their values  
. 鐣欏鐢宠璁哄潧-涓€浜╀笁鍒嗗湴  
compute total score.

第一题： 括号match  给一个string 里面是类似(({{[[<<adfadfad>adfasdf>}])>，问是不是match，leetcode easy级别的。  
第二题： 类似于 [Lowest Common Ancestor of a Binary Tree](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree)   但是完全不一样。是说给一个 公司的级别树，最上面是CEO， 然后employee下面有需要向他report的人的话，这个employee 就是manager，每个employee 最多有一个manager。然后给你两个employee 让你找出他们的最低的common manager。employee的数据结构如下：  
Employee {  
    int id;  
    string  name;  
   vector<Employee\*>  reports;. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
   vector<Employee\*> getReports();  
   int getId();  
   string getName();  
}   
Employee\* findCommonManager(Employee\* ceo,  Employee\* employee1, Employee\* employee2) {  
}  
方法就是用DFS分别去找employee1 and employee2. 找的过程中把她们的manager存到stack里面。 然后对stack1 and stack2 的size进行比较。把size大的那个stack先干成一样的。然后调用getId()去比 较   
while (stack1.top() -> getId() != stack2.top() -> getId()) {  
         stack1.pop();  
         stack2.pop();  
}  
if(!stak1.empty() )   
      return  stak1.top();  
return nullptr;

1. Employee\* findCommonManager(Employee\* ceo,  Employee\* employee1, Employee\* employee2) {
2. if (ceo == nullptr) {.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�
3. return nullptr;
4. }
5. if (ceo == employee1) {
6. return employee1;
7. }
8. if (ceo == employee2) {
9. return employee2;
10. }
11. bool isManagingEmployee1 = false;
12. bool isManagingEmployee2 = false;
13. for (int i = 0; i < ceo->getReports().size(); i++) {
14. Employee\* e = findCommonManager(ceo->getReports()[i], employee1, employee2);
15. if (e == employee1) {.鏈枃鍘熷垱鑷�1point3acres璁哄潧
16. isManagingEmployee1 = true;
17. } else if (e == employee2) {
18. isManagingEmployee2 = true;. Waral 鍗氬鏈夋洿澶氭枃绔�,
19. } else if (e != nullptr) {
20. return e;
21. }
22. }
23. if (isManagingEmployee1 && isManagingEmployee2) {. 鍥磋鎴戜滑@1point 3 acres
24. return ceo;
25. }
26. if (isManagingEmployee1) {
27. return employee1;
28. }
29. if (isManagingEmployee2) {
30. return employee2;
31. }
32. return nullptr;
33. }

class Solution {  
private:  
    int dir[5] = {0, 1, 0, -1, 0};  
public:  
    int findMinSteps(vector<vector<int>> & maze, pair<int, int> start, pair<int, int>& end) {  
        int steps = 0;  
        queue<pair<int,int>> myQueue;  
        int m = (int)maze.size();  
        int n = (int)maze[0].size();  
        // use the visited matrix to mark all the visited pos  
        vector<vector<bool>> visited(m, vector<bool>(n, false));. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
        myQueue.push({0, 0});  
        visited[0][0] = true;  
        while(!myQueue.empty()) {  
            auto a = myQueue.front();  
            myQueue.pop();. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
            for(int k = 0; k < 4; k++) {  
                int i = a.first + dir[k];  
                int j = a.second + dir[k + 1];  
                if(i >= 0 && i < m && j >= 0 && j < n && !visited[i][j] && maze[i][j] == 0) {.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
                    visited[i][j] = true;. Waral 鍗氬鏈夋洿澶氭枃绔�,  
                    if (i == end.first && j == end.second) {  
                        return steps;  
                    }  
                    myQueue.push({i, j});  
                }  
                      
            }  
        }  
        return INT\_MAX;  
    }. Waral 鍗氬鏈夋洿澶氭枃绔�,  
};

*// Encodes a tree to a single string.*

**public** String **serialize**(TreeNode root) {

**if**(root == **null**) **return** **null**;

String left = serialize(root.left);

String right = serialize(root.right);

**if**(left == **null** && right == **null**) **return** root.val+"";

StringBuilder sb = **new** StringBuilder();

sb.append(root.val);

**if**(left != **null**) sb.append(","+left);

**if**(right != **null**) sb.append(","+right);

**return** sb.toString();

}

*// Decodes your encoded data to tree.*

**public** TreeNode **deserialize**(String data) {

**if**(data == **null**) **return** **null**;

String[] nodeStrings = data.split(",");

**int**[] nodes = **new** **int**[nodeStrings.length];

**int** i = 0;

**for**(String node : nodeStrings) {

nodes[i++] = Integer.parseInt(node);

}

**return** deserialize(nodes, 0, nodes.length-1);

}

**private** TreeNode **deserialize**(**int**[] nodes, **int** start, **int** end) {

**if**(start > end) **return** **null**;

**if**(start == end) **return** **new** TreeNode(nodes[start]);

**int** leftEnd = start, rightBegin = start+1;

TreeNode root = **new** TreeNode(nodes[start]);

**for**(**int** i = start+1; i <= end; i++) {

**if**(nodes[i] > root.val) **break**;

leftEnd = i;

rightBegin = leftEnd+1;

}

root.left = deserialize(nodes, start+1, leftEnd);

root.right = deserialize(nodes, rightBegin, end);

**return** root;

}

**Given an array of integers A and an integer k, find a subarray that contains the largest sum, subject to a constraint that the sum is less than k**

发一个亚麻社招 OA 的面经，OA是在amcat上做的，包括75 min Coding + 15 min Culture + 5 min Feedback，其实开始做coding之后时间有90min  
Q1 丢棒球砸砖块，貌似是地里没有出现过的题，输入是一个字符串数组，每一个值可能是一个整数，或者Z，或者X，或者+。整数代表现在拿的分，X代表当前成绩是前一个分数Double，+代表当前成绩是前两个的和，Z代表移除前一个成绩，然后要求的是最后的总成绩  
例子： 输入 ["5", "-2", "4", "Z", "X", 9, "+", "+"]  
输出 27  
5 : sum = 5  
-2 : sum = 5 - 2 = 3  
4 : sum = 3 + 4 = 7  
Z : sum = 7 - 3 = 4  
X : sum = 3 + -2 \* 2 = -1 (因为4被移除了，前一个成绩是-2)  
9 : sum = -1 + 9 = 8  
+ : sum = 8 + 9 - 4 = 13 (前两个成绩是9和-4)  
+ : sum = 13 + 9 + 5 = 27 (前两个成绩是5 和 9)  
  
用一个stack就可以秒掉  
  
Q2: Movie network，是地里出现过的原题，BFS + PriorityQueue秒. [1point3acres.com/bbs](http://1point3acres.com/bbs)  
  
再附上整理的地里出现过的新版Amcat 社招OA面经，希望能帮助到大家，攒人品，求店面求onsite求[大米](http://www.1point3acres.com/bbs/forum.php?mod=collection&action=view&ctid=5&fromop=all)！  
  
  
. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)   
[Amazon](http://https/www.amazon.com/b?_encoding=UTF8&tag=1point3acres-20&linkCode=ur2&linkId=89c11e2c5b86155c5422f19cca1e9880&camp=1789&creative=9325&node=5) OA 社招  
  
1. Valid parentheses  
  
Leetcode 原题  
  
  
  
2. Find All Anagrams in a String  
. visit [1point3acres.com](http://1point3acres.com/) for more.. 1point 3acres 璁哄潧  
lintcode原题  
  
  
  
3. Find the min steps of maze.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
意思是说有一个M\*N的maze，0代表可以通过，1代表不可以通过，然后给你一个出口（x,y），找从（0,0）到出口的最少steps，如果找不到path就返回-1  
  
Leetcode原题 Unique Path II. visit [1point3acres.com](http://1point3acres.com/) for more.  
  
  
. From 1point 3acres bbs  
4. 寻找公司员工的最低共同上司    . 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
  
变化1:不是bineary tree  
变化2:ceo不是manager 如果共同上司是ceo return null  
  
变化3:员工可能不在这个公司 return null. [1point3acres.com/bbs](http://1point3acres.com/bbs)  
  
  
  
5. construct BST 后找俩node之间distance  
  
Given a list of unique integers, construct the binary tree by given order without rebalancing, then find out the distance between two nodes.. visit [1point3acres.com](http://1point3acres.com/) for more.  
  
. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
  
public static int bstDistance(int[] values, int n, int node1, int node2)  
  
for example,  
  
values= [5,6,3,1,2,4], n is the size of values, node1 is 2, node2 is 4, then function return 3. 1point 3acres 璁哄潧  
构建完BST如下，2和4呢，距离就是3  
      5  
   3      6  
1   4  
  
  2.  
  
  
  
  
6.  movie network.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
  
题目如下：  
假设有个Movie类，  
public class Movie  
{. visit [1point3acres.com](http://1point3acres.com/) for more.  
   int movieId;.  
   float rating;  
   List<Movie> similarMovies  
还有其他的getters  
}  
现在要求找到 k个和movie最相似 的movies。. visit [1point3acres.com](http://1point3acres.com/) for more.  
  
函数的signature大概是这样的：  
public static List<Movie> getNearest(Movie movie, int numSimilar)。   
  
举个栗子：  
m0 <-->m1, similarity 2-google 1point3acres  
mo <--> m2, similarity 3. 鐣欏鐢宠璁哄潧-涓€浜╀笁鍒嗗湴  
m1 <--> m3, similarity 4   
m2 <--> m5, similaity 5  
  
如果要返回和mo最相似的movie, 那么应该返回 m5 (只有有一条路径从 m1到 m5, 并且 5是最大的）； 如果返回3个最相似的就返回 m2, m3， m5（顺序不重要）； 如果需要返回10个，但是相似的只有9个，那么就返回9个。  
source movie本身不能在返回结果里面。  
  
可以的一个做法是 dfs + min-Heap(PriorityQueue)， 我们一直做dfs， 每次碰到一个新的movie，如果现在queue的size比 k小的话，直接加到queue里面； 如果新movie的rating比queue top movie的rating高的话， 把顶部movie.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
踢出队列，加上这个新的。  
  
  
update: 应该返回 m5 (只有有一条路径从 m1到 m5, 并且 5是最大的） --> 应该返回 m5 (只要有一条路径从 m1到 m5, 并且 5是最大的）

3月收到HR的邀请，要发简历过去，照做了，过了一段时间说给个OA，题目貌似就那么几个，地里都总结好了  
1. 算比赛分数的那个题，地里的原题  
2. 生成BST找LCA的那个题