# 類似Leetcode 224, 227

# Now let's switch over to the back-end of our social network. We have some automated batch jobs that we use to handle expensive tasks that run periodically throughout the day, like updating statistics for the most popular posts. We've been given some input that shows the dependencies between each of these batch jobs.

# For example, in this input, "clean" must be executed before "mapper" can execute.

# Given the last execution time for each step of the workflow, we want to find the set of all steps that are "stale" -- steps that have not executed since the last time one of their precursor steps executed. For example, in this case, "update" is in the output because "mapper" must occur before "update", but "update" has not been executed since the last time "mapper" was executed. If a task is stale, all tasks after it are stale too -- so "statistics" is stale because of "mapper".

#                          /--> reducer

#          /---> mapper --|

# clean --|                \--> update --\

#          \                              --> statistics

#           \---> metadata --------------/

#                         \

#                          \--> timestamp

# Sample input:

.留学论坛-一亩-三分地

# precursor\_steps = [

#   ["clean", "mapper"],

#   ["metadata", "statistics"],

#   ["mapper", "update"],

#   ["update", "statistics"],

#   ["clean", "metadata"],

#   ["mapper", "reducer"],

#   ["metadata", "timestamp"],

. 牛人云集,一亩三分地

# last\_execution\_times = [

#   ["clean", "20170302-1129"],

#   ["mapper", "20170302-1155"],

#   ["update", "20170302-1150"],

#   ["statistics", "20170302-1153"],

#   ["metadata", "20170302-1130"],

#   ["reducer", "20170302-1540"],

# Sample output (in any order):

# find\_stale\_steps(precursor\_steps, last\_execution\_times) =

#   ["update", "statistics", "timestamp"]

firend cycle  
1st Question: 输出所有的employee的friendlist -> 就是用一个map存起来然后打印就好了（这个是无向图，e.g: 1和2是朋友，2的列表里也要有1）  
2nd Question: 输出每个department里有多少人的朋友是其他部门的 ->也就是遍历一遍就好了. 涓€浜3rd Question: 输出是否所有employee都在一个社交圈 -> 我当时想的就是随便找一个点，用DFS遍历一遍，如果所有点都被遍历到就return true，不然就是false

有一个employList  
    String[] employeesInput = {  
      "1,Richard,Engineering",. 牛人云集,一亩三分地  
      "2,Erlich,HR",  
      "3,Monica,Business",. From 1point 3acres bbs  
      "4,Dinesh,Engineering",  
      "6,Carla,Engineering",  
      "9,Laurie,Directors"  
    };  
一个friendshipList，friend关系是双向的  
    String[] friendshipsInput = {  
      "1,2",. visit 1point3acres for more.  
      "1,3",  
      "1,6",  
      "2,4"  
    };

1. 写一个函数返回每个人的friend的adjacency list. 围观我们@1point 3 acres  
比如这个例子里返回  
1：2 3 6  
2：1 4  
3：1  
4：2  
6：1  
9：

1. public Map<Integer, Set<Integer>> generateMap(String[] employees, String[] friendships) {
2. Map<Integer, Set<Integer>> map = new HashMap<>();
3. for (String employee : employees) {
4. String[] parts = employee.split(","); 来源一亩.三分地论坛.
5. int employeeId = Integer.parseInt(parts[0]);
6. map.put(employeeId, new HashSet<>());
7. }
8. . from: 1point3acres
9. for (String friendship : friendships) {
10. String[] parts = friendship.split(",");
11. int id1 = Integer.parseInt(parts[0]);. from: 1point3acres
12. int id2 = Integer.parseInt(parts[1]);
13. map.get(id1).add(id2);
14. map.get(id2).add(id1);
15. }
17. return map;
18. }

复制代码

2. 按照department分类，统计每一个department人数，和这个department有多少人有其他department的朋友  
比如这个例子里. 1point 3acres 论坛  
engineering: 3(人数) 2(有其他部门朋友的员工数)  
hr: 1 1

business: 1 1 S  
directors: 1 0

1. public Map<String, int[]> getDepartmentStat(String[] employees, String[] friendships) {
2. Map<String, int[]> res = new HashMap<>();
3. Map<Integer, Set<Integer>> friendsMap = generateMap(employees, friendships);
4. . 围观我们@1point 3 acres
5. Map<String, Set<Integer>> departmentsMap = new HashMap<>();
6. for (String employee : employees) {
7. String[] parts = employee.split(",");. more info on 1point3acres
8. int employeeId = Integer.parseInt(parts[0]);
9. if (!departmentsMap.containsKey(parts[2])) {.1point3acres网
10. departmentsMap.put(parts[2], new HashSet<>());
11. res.put(parts[2], new int[] { 0, 0 });
12. }
13. departmentsMap.get(parts[2]).add(employeeId);
14. res.get(parts[2])[0]++;-google 1point3acres
15. }
17. for (String employee : employees) {
18. String[] parts = employee.split(",");. visit 1point3acres for more.
19. int employeeId = Integer.parseInt(parts[0]);
20. String department = parts[2];
22. Set<Integer> employeeSet = departmentsMap.get(department);
23. Set<Integer> friendsList = friendsMap.get(employeeId);
24. for (int friend : friendsList) {
25. if (!employeeSet.contains(friend)) {
26. res.get(department)[1]++;
27. break;   // Terrible! Misunderstand the problem
28. }
29. }. 牛人云集,一亩三分地
30. }
32. return res;
33. }

复制代码

这里我理解错了，理解成了这个部门有多少在部门外的朋友，结果最后输出总是不和需要的相符，面试官似乎也没有发现我理解错了，最后没改完，面试结束发现我理解错了。这真是个教训，面试官复制粘贴了一大段题目我懒得看，然后就听他的解释，然后听错了，最后gg，哎，吸取教训了还是要自己读一遍题。

算task和pretask輸出task by level的那題. Just calculate indegree, bfs  
是像這樣  
input = {  
{"cook", "eat"},   // do "cook" before "eat"  
{"study", "eat"},  
{"sleep", "study"}}  
  
output (steps of a workflow):  
{{"sleep", "cook"},.  
{"study"},  
{"eat"}}  
  
  
// We are working on a security sy[stem](http://www.1point3acres.com/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%9C%9F%E5%AE%89%E5%85%A8%E9%83%A82012%E5%B9%B4%E5%BA%A6%E6%9B%B4%E6%96%B0stem%E4%B8%93%E4%B8%9A%E5%90%8D%E5%8D%95-%E7%9C%8B%E7%9C%8B%E4%BD%A0%E7%9A%84%E4%B8%93%E4%B8%9A/) for a badged-access room in our company's building.  
// Given an ordered list of employees who used their badge to enter or exit the room, write a function that returns two collections:  
// 1. All employees who didn't use their badge while exiting the room – they recorded an enter without a matching exit.

+1

Set<String> getlist(String[][] record) {

Set<String> seta = new HashSet<>();

Set<String> setb = new HashSet<>();

If(record == null || record.length == 0) {

Return new HashSet<>();

}

Map<String,Integer> map = new HashMap<>();

For(int[] r : record) {

Integer curr = map.get(r[0]);

If(r[1].equals(“enter”) {

If(curr == null || curr == 0) {

Curr = 1;

}else {

Seta.add(r[0]);

Curr = 1;

}

Map.put(r[0],curr);

}else if (r[1].equals(“exit”)) {

If (curr == null || curr<0) {

Setb.add(r[0]);

Curr = 0;

}else{

Curr = curr-1;

}

Map.put(r[0],curr);

}

}

}  
  
// 2. All employees who didn't use their badge while entering the room  – they recorded an exit without a matching enter.  
exit = -1, enter = 1;  
// badge\_records = [  
//   ["Martha",   "exit"],-google 1point3acres  
//   ["Paul",     "enter"],. [1point3acres.com/bbs](http://1point3acres.com/bbs)  
//   ["Martha",   "enter"],  
//   ["Martha",   "exit"],  
//   ["Jennifer", "enter"],. more info on [1point3acres.com](http://1point3acres.com/)  
//   ["Paul",     "enter"],. From 1point 3acres bbs  
//   ["Curtis",   "enter"],  
//   ["Paul",     "exit"],. visit [1point3acres.com](http://1point3acres.com/) for more.  
//   ["Martha",   "enter"],  
//   ["Martha",   "exit"],  
//   ["Jennifer", "exit"],. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
// ]

// find\_mismatched\_entries(badge\_records). 鐣欏鐢宠璁哄潧-涓€浜╀笁鍒嗗湴  
// Expected output: ["Paul", "Curtis"], ["Martha"]  
/\*\*  
  
  
\* We want to find employees who badged into our secured room unusually often. We have an unordered list of names and access times over a single day. Access times are given as three or four-digit numbers using 24-hour time, such as "800" or "2250".  
  
\* Write a function that finds anyone who badged into the room 3 or more times in a 1-hour period, and returns each time that they badged in during that period. (If there are multiple 1-hour periods where this was true, just return the first one.)  
Map<String,List<Integer>> map = new HashMap<>();

For(int[] r : records) {

List<Integer> list = map.get(r[0]);

If(list == null) {

List = new ArrayList<>();

}

List.add(convert(r[1]));

Map.put(r[0],list);

}

For(Map.Entry<String,List<Integer>> entry: map.entrySet()) {

List<Integer> list = entry.getValue();

If(list.size()>=3){

Collections.sort(list);

For(int i=0; i<list.size();i++) {

Int curr = list.get(i);

Int index = find(list,curr+60);

If(index-i>=3) {

Res.addAll(list,I,index);

Break;

}

}

}

}  
  
\* badge\_records = [  
  
\* ["Paul", 1355],  
\* ["Jennifer", 1910],  
  
\* ["John", 830],  
  
\* ["Paul", 1315],  
  
\* ["John", 835],  
鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�.   
\* ["Paul", 1405],  
  
\* ["Paul", 1630],  
  
\* ["John", 855],  
  
\* ["John", 915],  
  
\* ["John", 930],  
  
\* ["Jennifer", 1335],  
  
\* ["Jennifer", 730],  
  
\* ["John", 1630],  
  
\* ]-google 1point3acres  
  
\* <p>  
. visit [1point3acres.com](http://1point3acres.com/) for more.  
\* Expected output:  
  
\* John: 830 835 855 915 930  
  
\* Paul: 1315 1355 1405.1point3acres缃�  
  
  
  
  
  
  
Intuit 网上coding competition的一道题给一个矩阵，矩阵里的每个元素是1，但是其中分布着一些长方形区域， 这些长方形区域中的元素为0. 要求输出每个长方形的位置（用长方形的左上角元素坐标和右下角元素坐标表示）。  
example：  
input:  
[  
[1,1,1,1,1,1],    
[0,0,1,0,1,1],  
[0,0,1,0,1,0],  
[1,1,1,0,1,0],  
[1,0,0,1,1,1]  
]  
output:. From 1point 3acres bbs  
[  
[1,0,2,1],  
[1,3,3,3],  
[2,5,3,5],  
[4,1,4,2]  
]  
如果 Matrix 中有多个由0组成的长方体，请返回多套值（前提每两个长方体之间是不会连接的，所以放心）. 不改变输入的做法  
不过还有第三问，就是connected components. 1point 3acres 璁哄潧  
第三问 基本上就是leetcode connected components,只不过是返回一个listoflist，每个list是一个component的所有点坐标  
那个图是1,01,0组成的矩阵，0组成的就是各种图形。跟前面关系的确不大  
如果矩阵里有多个不规则的形状，返回这些形状。这里需要自己思考并定义何谓“返回这些形状”  
public List<List<Integer>> getrectangle(int[][] matrix) {

If(matrix == null || matrix.length == 0) {

Return new ArrayList<>();

}

Boolean[][] visit = new boolean[matrix.length][matrix[0].length];

List<List<Integer>> list = new ArrayList<>();

For(int i=0; i<matrix.length; i++) {

For(int j=0;j<matrix[0].length;j++) {

If(matrix[i][j]==0 && !visit[i][j]) {

Int[] coordinate=new int[4];

Coordinate[0]=I;

Coordinate[1]=j;

Coordinate[2]=I;

Coordinate[3]=j;

Dfs(matrix,i,j,visit,coordinate);

List.add(Arrays.asList(coordinate[0],coordinate[1],coordinate[2],coordinate[3]);

}

}

}

Return list;

}

Private void dfs(int[][] matrix, int row, int col, Boolean[][] visit, int[] coordinate) {

If(row<0 || row>=matrix.length||col<0||col>=matrix[0].length||visit[row][col]||matrix[row][col]!=0) {

Return;

}

Visit[row][col]=true;

Int[][] dirs.={{-1,0},{1,0},{0,1},{0,-1}};

Coordinate[0]=Math.min(coordinate[0],row);

Coordinate[1]=Math.min(coordinate[1],col);

Coordinate[2]=Math.max(coordinate[2],row);

Coordinate[3]=Math.max(coordinate[3],col);

For(int[] dir: dirs.) {

Int r = row+dir[0];

Int c=col+dir[1];

Dfs(matrix,r,c,visit,coordinate)

}

}

第一題是給你一個string例如"2+3-999"回傳計算結果int.

第二題加上parenthesis 例如"2+((8+2)+(3-999))"一樣回傳計算結果  
第三道题是加了变量名的。。会给你一个map比如{'a':1, 'b':2, 'c':3}，假设输入为"a+b+c+1"输出要是7，如果有未定义的变量，比如"a+b+c+1+d"输出就是7+d

LC基础计算机题  
只有+ - 没有 \* or / 也不需要考虑空白  
  
follow up1 : with (, )  
只考虑 + - ，和leetcode原题一样  
做完这题大概剩下15分钟  
  
follow up2  
input string has +, - , (, ),  and 字符 / 字串变量  
code = {  
  e : 1,  
  y : 2,.

  temperature : 5  
}         
input string:  
(e + 8) - pressure + 3 - temperature  
--> 9 - pressure + 3 - 5  
--> 7 – pressure

**public** String expresscalculator(String input, Map<String,Integer> map) {

**if**(input == **null** || input.length()==0) {

**return** "";

}

String simple = simplify(input,map);

**int** i = 0;

Deque<cell> stack = **new** LinkedList<>();

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

**int** num = 0;

**int** sum=0;

**int** sign = 1;

**while**(i<simple.length()){

**if**(isdigit(simple.charAt(i))) {

num=num\*10+simple.charAt(i)-'0';

i++;

}**else** **if** (simple.charAt(i)=='+'||simple.charAt(i)=='-'){

sum=sum+num\*sign;

sign=simple.charAt(i)=='+'?1:-1;

num=0;

i++;

}**else** **if** (ischar(simple.charAt(i))){

**if**(sign==1){

sb.append('+');

}**else**{

sb.append('-');

}

**while**(i<simple.length()&&ischar(simple.charAt(i))){

sb.append(simple.charAt(i));

i++;

}

}**else** **if** (simple.charAt(i)=='(') {

stack.offerFirst(**new** cell(sum,sign,sb.toString()));

sum=0;

sign=1;

sb = **new** StringBuilder();

i++;

}**else** **if** (simple.charAt(i)==')') {

cell prev = stack.pollFirst();

sum=sum+sign\*num;

**int** t = prev.sign\*sum;

num=0;

sum=prev.sum+t;

StringBuilder next = **new** StringBuilder();

next.append(prev.str);

**if**(sb.length()>0){

next.append(update(prev.sign,sb.toString()));

}

sb = next;

sign=1;

i++;

}

}

**if**(num!=0){

sum=sum+sign\*num;

}

sb.append('+');

sb.append(sum);

**return** sb.toString();

}

**private** String update(**int** sign, String s){

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

**if**(sign==1){

**return** s;

}**else**{

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

**if**(s.charAt(i)=='+'){

sb.append('-');

}**else** **if** (s.charAt(i)=='-'){

sb.append('+');

}**else**{

sb.append(s.charAt(i));

}

}

}

**return** sb.toString();

}

**class** cell{

**int** sum;

**int** sign;

String str;

**public** cell(**int** s, **int** sign, String str){

**this**.sum=s;

**this**.sign=sign;

**this**.str=str;

}

}

**private** String simplify(String input, Map<String,Integer> map) {

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

**int** i = 0;

**while**(i<input.length()){

**if**(ischar(input.charAt(i))) {

**int** fast = i;

StringBuilder temp = **new** StringBuilder();

**while**(fast<input.length() && ischar(input.charAt(fast))){

temp.append(input.charAt(fast));

fast++;

}

String s = temp.toString();

**if**(map.containsKey(s)){

sb.append(map.get(s));

i=fast;

}**else**{

sb.append(s);

i=fast;

}

}**else**{

sb.append(input.charAt(i));

i++;

}

}

**return** sb.toString();

}

**private** **boolean** ischar(**char** c){

**return** c>='a' && c<='z';

}

**private** **boolean** isdigit(**char** c){

**return** c>='0' && c<='9';

}

Q1: 設計一個 sparse vector class  
  
sparseVector v = new sparseVector(100); //size constructor; size is 100  
    v.set(0, 1.0);  
    v.set(3, 2.0);  
    v.set(80,-4.5);  
  
    Sy[stem](http://www.1point3acres.com/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%9C%9F%E5%AE%89%E5%85%A8%E9%83%A82012%E5%B9%B4%E5%BA%A6%E6%9B%B4%E6%96%B0stem%E4%B8%93%E4%B8%9A%E5%90%8D%E5%8D%95-%E7%9C%8B%E7%9C%8B%E4%BD%A0%E7%9A%84%E4%B8%93%E4%B8%9A/).out.println(v.get(80)); //should print -4.5  
    System.out.println(v.get(50)); //should print 0.0  
  
    try {  
       System.out.println(v.get(101)); //error -- index out of range  
       throw new IllegalStateException("We should not get here, an exception should have been thrown");   
    } catch (IndexOutOfBoundsException t) {  
       // success  
    }    System.out.println(v.toString()); //should print something like [1.0, 0.0, 0.0, 2.0, 0.0, ...]    
  
  
Q2:  
  
Add these operations to your library: Addition, dot product, and cosine. Formulae for each are provided below; we’re more interested in you writing the code than whether you’ve memorized the formula. For each operation, your code should throw an error if the two input vectors are not equal length

Sample input/output:.

//Note: This is pseudocode. Your actual syntax will vary by language..鏈枃鍘熷垱鑷�1point3acres璁哄潧  
v1 = new vector(5)  
v1[0] = 4.0  
v1[1] = 5.0  
. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)   
v2 = new vector(5)  
v2[1] = 2.0  
v2[3] = 3.0. From 1point 3acres bbs  
  
v3 = new vector(2)  
. from: [1point3acres.com/bbs](http://1point3acres.com/bbs)   
print v1.add(v2) //should print [4.0, 7.0, 0.0, 3.0, 0.0]  
print v1.add(v3) //error -- vector lengths don’t match  
  
print v1.dot(v2) //should print 10  
print v1.dot(v3) //error -- vector lengths don’t match  
  
print v1.cos(v2) //should print 0.433  
print v1.cos(v3) //error -- vector lengths don’t match  
  
  
Formulae:  
Addition  
a.add(b) = [a[0]+b[0], a[1]+b[1], a[2]+b[2], ...]

Dot product  
a.dot(b) = a[0]\*b[0] + a[1]\*b[1] + a[2]\*b[2] + ...  
  
Cosine  
a.cos(b) = a.dot(b) / (norm(a) \* norm(b))  
//norm(a) = sqrt(a[0]^2 + a[1]^2 + a[2]^2 + ...). 鐣欏鐢宠