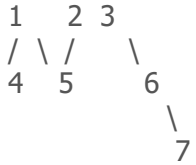


输入是 int[][] input, input*[0]是 input[1] 的 parent，比如 {{1,4}, {1,5}, {2,5}, {3,6}, {6,7}}会形成上面的图*

*第一问是只有 0 个 parents 和只有 1 个 parent 的节点*

*第二问是 两个指定的点有没有公共祖先*

*第三问是就一个点的最远祖先，如果有好几个就只需要输出一个就好，举个栗子，这里 5 的最远祖先可以是 1 或者 2，输出任意一个就可以*

*时间很紧，一个钟头基本上说说 oa 的思路就过去一刻钟了，然后昨晚还要分析时间复杂度和空间辅助度*

```
List<Integer> findparent(int[][] edges){
        If(edges==null || edges.length==0) {
                Return new ArrayList<>();
        }
        Map<Integer,List<Integer>> map = new HashMap<>();
        Map<Integer,Integer> indegreemap = new HashMap<>();
        For(int[] edge : edges) {
                List<Integer> list = map.get(edge[1]);
                If(list == null) {
                        List = new ArrayList<>();
                }
                List.add(edge[0]);
                Map.put(edge[1],list);
                List = map.get(edge[0]);
                If(list == null) {
                        List = new ArrayList<>();
                }
                Map.put(edge[0],list);
        }
        For(Map.Entry<Integer,List<Integer>> entry: indegreemap.entrySet()){
                If(entry.getValue().size()==0) {
                        Res1.add(entry.getKey());
                }else if (entry.getValue().size()==1) {
                        Res2.add(entry.getKey());
                }
        }
}

Private Boolean hascommon(int[][] edges, int a, int b) {

        Set<Integer> a_parent= new HashSet<>();
        getall(map,a,a-parent);
        Set<Integer> b_parent = new HashSet<>();
```

```
        getall(map,b,b_parent);


        For(int n: bparent) {
                If(a_parent.contains(n)) {
                        Return true;
                }
        }
        Return false;
}
Private void getall(Map<Integer,List<Integer>> map, int a, Set<Integer> parent) {
        List<Integer> list = map.get(a);
        If(list == null || list.size()==0) {
                Return;
        }
        For(int p : list) {
                Parent.add(p);
                Getall(map,p,parent);
        }
}

Private void highestparent(Map<Integer,List<Integer>> map, int a, int[] p, int step) {
        List<Integer> next = map.get(a);
        if(step>p[0]) {
                p[0]=step;
                p[1]=a;
        }
        If(next!=null && next.size()>0) {
                For(int pa:next) {
                        Highestparent(map,pa,p,step+1);
                }
        }
}
```

*下面是我整理的地里所有的 karat 面筋，攒人品*

*以下内容需要积分高于 180 才可浏览*

*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*

*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*

*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 system 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"],
//   ["Paul",     "enter"],. 1point3acres.com/bbs
//   ["Martha",   "enter"],
//   ["Martha",   "exit"],
//   ["Jennifer", "enter"],. more info on 1point3acres.com
//   ["Paul",     "enter"],. From 1point 3acres bbs
//   ["Curtis",   "enter"],
//   ["Paul",     "exit"],      1point3acres.com
//   ["Martha",   "enter"],
//   ["Martha",   "exit"],
//   ["Jennifer", "exit"],
// ]


// find_mismatched_entries(badge_records)
// 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],

* ["Paul", 1405],

* ["Paul", 1630],

* ["John", 855],

* ["John", 915],

* ["John", 930],

* ["Jennifer", 1335],

* ["Jennifer", 730],

* ["John", 1630],

* ]

* <p>
```

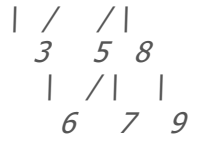
```
* Expected output:

* John: 830 835 855 915 930

* Paul: 1315 1355 1405
```



1 2 4



*第一题：输出只有 1 个和 0 个 parent 的 individual*

*第二题是，任意给两个节点，给出他们共同的 ancestor 节点（可以有不只一个）。*

*第三题是，任意给一个节点，找出离它最远的 ancestor 节点，可以是多个。第*

*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:*

*[*

*[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*

*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],coor
        dinate[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 int calculator(String input) {
        If(input == null || input.length()==0) {
                Return 0;
        }
        Int sum=0;
        Int num=0;
        Int sign = 1;
        For(int i=0; i<input.length();i++) {
         If(input.charAt(i)>='0' && input.charAt(i)<='9') {
                Num=num*10+input.charAt(i)-'0';
        }else{
                Sum=sum+num*sign;
                Sign=input.charAt(i)=='+'?1:-1;
                Num=0;
        }
        }
        Return sum;
}

Public int calculator(String input) {
        If(input == null || input.length()==0) {
                Return 0;
        }
        Int sum=0;
        Int num=0;
        Int sign = 1;
        For(int i=0; i<input.length();i++) {
         If(input.charAt(i)>='0' && input.charAt(i)<='9') {
                Num=num*10+input.charAt(i)-'0';
        }else if(input.charAt(i)=='+'||input.charAt(i)=='-'){
                Sum=sum+num*sign;
                Sign=input.charAt(i)=='+'?1:-1;
                Num=0;
        }else if (input.charAt(i)=='(') {
                Stack.offerFirst(sum);
                Stack.offerFirst(sign);
                Sign=1;
                Sum=0;
        }else if (input.charAt(i)==')') {
                Sum=sum+sign*num;
                Int temp = stack.pollFirst()*sum;
                Sum=temp+stack.pollFirst();
                Sign = 1;
                Num=0;
        }
        }
        Return num==0?sum:sum+sign*num;
}
```

```
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';
        }
```

*第一题：给广告在每个 domain 上被 click 的次数* 

*要求返回 domain 及其所有 sub domain 被 click 的总次数*

*输入：[*

*["google.com", "60"],*
*["yahoo.com", "50"],*
*["sports.yahoo.com", "80"]*
*]*

*输出：[*

*["com", "190"], (60+50+80)*
*["google.com", "60"],*
*["yahoo.com", "130"] (50+80)*
*["sports.yahoo.com", "80"]*
*]*

```
List<List<String>> countClick(String[][] array) {

    If(array == null || array.length==0) {
        Return new ArrayList<>();
    }
    Map<String,Integer> map = new HashMap<>();
    for(int i=0; i<array.length; i++) {
        String curr = array[i][0];
        Int count = convert(array[i][1]);
        Int j=curr.length()-1;
        While(j>=0) {
            While(j>=0 && curr.charAt(j)!='.') {
                Sb.append(curr.charAt(j));
                j--;
            }
            String temp = sb.toString();
            Integer c = map.get(temp);
            If(c == null) {
                C=count;
            }else{
                C=c+count;
            }
            Map.put(temp,c);
            If(j>=0 && curr.charAt(j)=='.') {
                Sb.append(curr.charAt(j));
```

```
                        j--;
                    }
                }
            }
            List<List<String>> res = new ArrayList<>();
            For(Map.Entry<String,Integer> entry : map.entrySet()) {
                List<String> temp = new ArrayList<>();
                String key = entry.getKey();
                Integer value = entry.getValue();
                Temp.add(key);
                Temp.add(Integer.toString(value));
                Res.add(new ArrayList(temp));
            }
}
```

*第二题：给每个 user 访问历史记录，找出两个 user 之间 longest continuous common history*

*输入：[*

*["3234.html", "xys.html", "7hsaa.html"], // user1*

*["3234.html", "sdhsfjdsh.html", "xys.html", "7hsaa.html"] // user2*

*], user1 and user2（指定两个 user 求 intersect）*

*输出：["xys.html", "7hsaa.html"]*

user0 = [ "/nine.html", "/four.html", "/six.html", "/seven.html", "/one.html" ]
user2 = [ "/nine.html", "/two.html", "/three.html", "/four.html", "/six.html", "/seven.html" ]
user1 = [ "/one.html", "/two.html", "/three.html", "/four.html", "/six.html" ]
user3 = [ "/three.html", "/eight.html" ]

Sample output:

(user0, user2):
  /four.html
  /six.html
  /seven.html

(user1, user2):

  /two.html
  /three.html
  /four.html
  /six.html

(user0, user3):
  None

(user1, user3):
  /three.html

```
public List<String> lc(String[] one, String[] two){
            int[][] dp = new int[one.length+1][two.length+1];
            int max = 0;
```

```
            int end = -1;
            for(int i=1;i<one.length+1;i++){
            for(int j=1; j<two.length+1;j++) {
                  if(one[i-1].equals(two[j-1])) {
                  dp[i][j]=dp[i-1][j-1]+1;
            }else{
                  dp[i][j]=0;
            }
            if(dp[i][j]>max){
                  max = dp[i][j];
                  end = i-1;
            }
            }
            }
            List<String> list = new ArrayList<>();
            for(int j=end; j>end-max;j--){
                   list.add(one[j]);
            }
            return list;
      }
```

*第一题：类似 meeting rooms，输入是一个 int[][] meetings, int start, int end, 每个数都是时间，13：00 =》 1300， 9：30 =》 930， 看新的 meeting 能不能安排到 meetings*
*ex: {[1300, 1500], [930, 1200],[830, 845]}, 新的 meeting[820, 830], return true; [1450, 1500] return false;*

Int[] start = new int[meetings.length];
Int[] end = new

*第二题：类似 merge interval，唯一的区别是输出，输出空闲的时间段，merge 完后，再把两两个之间的空的输出就好，注意要加上 0 - 第一个的 start time*

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);

   System.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.
v1 = new vector(5)
v1[0] = 4.0
v1[1] = 5.0

v2 = new vector(5)
v2[1] = 2.0
v2[3] = 3.0

v3 = new vector(2)

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 + ...)

```
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';
```

```
}
```