Hệ thống CodeLearn mới ra đời và muốn phát triển tính năng chống tấn công

Hệ thống lưu lịch sử truy cập của người dùng với thông tin là thời gian truy cập (bao gồm năm, tháng, ngày, giờ, phút, giây) và tài khoản của người dùng. Để đơn giản nên bỏ qua ko xét đến các link truy cập

Thuật toán đơn giản như sau. Đếm xem trong m phút liên tục, nếu người dùng có nhiều hơn hoặc bằng n truy cập (bao gồm cả lượt truy cập tại thời điểm đầu và kết thúc) thì coi như là người dùng đang tấn công  
  
Nhớ rằng trong cùng 1 thời điểm, 1 người dùng có thể truy cập nhiều hơn 1 lần. Trong trường hợp này, hệ thống sẽ lưu nhiều dòng có nội dung giống hệt nhau. Bạn cần chú ý trường hợp này

Cho 1 danh sách lịch sử truy cập của người dùng, mỗi phần tử trong danh sách là 1 xâu kí tự với định dạng: "yyMMddhhmmss username" với ý nghĩa username truy cập vào thời gian hh:mm:ss ngày dd/MM/yy. Cho 2 tham số m và n, hãy xác định danh sách các người dùng bị coi là tấn công (tồn tại khoảng thời gian m phút liên tục có trên n truy cập). Chú ý rằng thời gian truy cập có thể ko được sắp xếp theo thứ tự tăng dần và có thể trùng nhau (nếu trùng nhau thì coi như người đó truy cập nhiều lần vào 1 thời điểm)  
  
**Ví dụ:**

* Với accessList=["190920175400 vietcv", "190920175600 vietcv"], m=2, n=2 thì kết quả attackList(accessList, m, n) = ["vietcv]  
  Trong 2 phút, người dùng vietcv truy cập 2 lần vào lúc 17:54 và 17:56 ngày 20/09/19 nên bị coi là tấn công

**Đầu vào/đầu ra:**

* **Thời gian chạy:** 0.5 giây với C++, 3 giây với Java/C# và 4 giây với Python/Js
* **Đầu vào: array.string accessList**  
  Mảng xâu kí tự. Mỗi xâu kí tự trong mảng tuân theo định dạng "yyMMddhhmmss username". Dữ liệu đảm bảo đúng đắn (yy trong khoảng 17 tới 19, MM trong khoảng 01 tới 12, dd trong khoảng 01 tới 31 tuỳ theo tháng, hh trong khoảng 0 tới 23, mm và ss trong khoảng 00 tới 59, username chỉ chứa kí tự in thường và chữ số như vietcv1, vietcv100)   
    
  0<= accessList.length() <= 10000  
  1<= username.length() <= 100
* **Đầu vào: integer m, n**2 tham số thể hiện số phút và số lượt truy cập bị nghi ngờ là tấn công  
  1 <= m <= 1000  
  1 <= n <= 1000
* **Đầu ra: array.string**  
  Danh sách các người dùng bị nghi ngờ tấn công theo luật trên, sắp xếp theo thứ tự từ điển tăng dần

An đang tập làm bảng đèn điện tử  
  
Ở phiên bản đơn giản, An chỉ muốn hiển thị số  
  
Theo thiết kế của An, mỗi chữ số được thể hiện bằng một bảng có 5\*3 bóng đèn xếp liên tục với nhau giống như bảng sau: (\* là ô có bóng đèn sáng, còn . là ô bóng đèn tối).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| \*\*\* \*.\* \*.\* \*.\* \*\*\* | ..\* ..\* ..\* ..\* ..\* | \*\*\* ..\* \*\*\* \*.. \*\*\* | \*\*\* ..\* \*\*\* ..\* \*\*\* | *\*.\** \*.\* *\*\*\* ..\* ..\** | \*\*\* \*.. \*\*\* ..\* \*\*\* | \*\*\* \*.. \*\*\* *\*.\* \*\*\** | \*\*\* ..\* .\*\* ..\* ..\* | \*\*\* *\*.\* \*\*\* \*.\* \*\*\** | \*\*\* \*.\* \*\*\* ..\* \*\*\* |

Như vậy, để thể hiện một có n chữ số, An cần tạo ra một bảng có 5\*3n bóng đèn xếp liên tục nhau (giả định giữa các số không có khoảng cách)

Cho trước danh sách gồm 5 xâu kí tự có độ dài bằng nhau thể hiện trạng thái của bảng đèn giống như miêu tả (\* thể hiện đèn sáng, . thể hiện đèn tắt), nhiệm vụ của bạn là tìm xem số đang được hiển thị trên bảng đèn là bao nhiêu?

**Ví dụ:**

* Với lightStatus = ["\*\*\*","\*.\*","\*.\*","\*.\*","\*\*\*"], thì kết quả digitalNumber(lightStatus) = "0"
* Với lightStatus = ["..\*\*\*\*","..\*\*.\*","..\*\*.\*","..\*\*.\*","..\*\*\*\*"], thì kết quả digitalNumber(lightStatus) = "10"

**Đầu vào/đầu ra:**

* **[Thời gian chạy]** 0.2 giây (C++)
* **[Đầu vào]** **array.string lightStatus**  
  Một mảng gồm 5 xâu kí tự có độ dài bằng nhau (độ dài của các xâu là 1 số chia hết cho 3) chỉ chứa kí tự \* và .  
  3 <= lightStatus.length() <= 3000  
  Dữ liệu input đảm bảo tính đúng đắn, output là chuỗi các số từ 0-9
* **[Đầu ra]** **string**  
  Số đang được hiển thị trên bảng đèn

Với số tự nhiên n>=1, n giai thừa (ký hiệu là n!) được tính bằng tích các số từ 1 tới n

Ví dụ 1! = 1, 3! = 1\*2\*3=6

Việt xác định hàm F(n) = 1! \* 2! \* 3! \* ... \* n! và muốn đếm xem F(n) có tận cùng bao nhiêu chữ số 0? Chú ý rằng n là khá lớn, nên bạn không thể dùng cách thông thường đc đâu nhé

**Ví dụ:**

* Với n=3 thì numberOfZeroDigits(n) = 0  
  Ta có F(3) = 1! \* 2! \* 3! = 1 \* 2 \* 6 = 12. Ko có số 0 nào ở phía cuối của kết quả
* Với n=5 thì numberOfZeroDigits(n) = 1  
  Ta có F(5) = 1! \* 2! \* 3! \* 4! \* 5! = 1 \* 2 \* 6 \* 24 \* 120 = 34560. Số này có 1 chữ số 0 ở phía cuối

**Đầu vào/đầu ra:**

* **Thời gian chạy: 0.2 giây**
* **Đầu vào: integer n**  
  Số nguyên n  
  1 <= n <= 2 \* 10^8
* **Đầu ra: int64**   
  Số lượng chữ số 0 tận cùng của F(n)

Nước X có n thành phố (n<=20). Mỗi thành phố có tên được tạo ra bởi các chữ cái (không phân biệt hoa thường) và chữ số (không có các kí tự khác). Ví dụ các tên như A1, A100 hay ba là các tên thành phố hợp lệ. Lưu ý, ở nước X chữ cái không phân biệt hoa thường. Do đó, tên thành phố A1 với a1 là cùng chỉ 1 thành phố.

Đường đi giữa các thành phố là các đường đi 2 chiều (nghĩa là nếu có đường đi giữa 2 thành phố a và b thì có thể đi từ a đến b, và đi từ b đến a).

Cho bản đồ đường đi giữa các thành phố thuộc nước X. Bạn cần xác định xem, có tồn tại 1 lộ trình đi du lịch, xuất phát từ thành phố A nào đó, đi qua lần lượt mỗi thành phố đúng 1 lần rồi quay trở lại chính thành phố A hay không?

**Ví dụ:**

* Với pathList = ["A-B", "B-C"] thì kết quả travelPath(pathList) = false  
  Bạn có thể xuất phát từ A, đi tới B. Rồi từ B đi tới C. Nhưng không thể quay trở lại được A. Do đó câu trả lời là false
* Với pathList = ["A1-B", "B-C", "a1-C"] thì kết quả travelPath(pathList) = true  
  Bạn có thể xuất phát từ A1, đi tới B. Rồi từ B đi tới C và quay trở lại A1.

**Đầu vào/đầu ra:**

* **Thời gian chạy:** 0.2 giây
* **Đầu vào: array.string pathList**  
  Mảng các xâu kí tự thể hiện đường đi trực tiếp giữa các thành phố của nước X. Mỗi xâu kí tự có dạng "A-B", trong đó A và B là tên hai thành phố thuộc nước X, A và B chỉ chứa các chữ cái và chữ số có độ dài không quá 10  
    
  Đảm bảo dữ liệu đúng đắn, ko có đường đi từ thành phố đến chính nó.   
  1 <= pathList.length() <= 100
* **Đầu ra: boolean**  
  True nếu tồn tại chu trình, false nếu ngược lại

Bình đã tạo ra 1 hệ thống mã hoá chữ cái và chữ cái như sau:

a               <-> 1  
b               <-> 2  
c               <-> 3  
...  
z               <-> 26  
aa             <-> 27  
ab             <-> 28  
...  
codewar    <-> 1,106,899,344  
codewas    <-> 1,106,899,345  
...  
codelearn  <-> 748,258,989,102

Bạn cần viết chương trình mã hóa hai chiều:

* Nếu xâu đầu vào chứa các ký tự chữ cái in thường 'a' tới 'z' thì kết quả là 1 số dưới dạng xâu (cứ 3 chữ số thì thêm dấu ',' ví dụ như 1,234 hay 2,345,563).
* Nếu xâu đầu vào chứa các ký tự chữ số thì kết quả sẽ là 1 xâu chứa các ký tự chữ cái 'a' tới 'z'.

**Ví dụ**

* Cho s = "25", thì kết quả convert(s) = "y".
* Cho s = "bcd", thì kết quả convert(s) = "1,434".
* Cho s = "zz", thì kết quả convert(s) = "702".
* Cho s = "codelearn", thì kết quả convert(s) = "748,258,989,102".
* Cho s = "748258989102", thì kết quả convert(s) = "codelearn".

**Đầu vào/Đầu ra**

* **[Thời gian chạy] 0.5s (C++), 3s (C#, Java), 4s (python, js)**
* **[Đầu vào] string s**Xâu kí tự s chỉ chứa các tự chữ cái in thường hoặc chỉ chứa các ký tự chữ số  
  1 <= s.size() <= 50.
* **[Đầu ra] string**

Cho số n. Hãy đếm số bộ 3 số nguyên không âm a, b, c (0 <= a < b < c) thoả mãn:

a2 + b2+ c2= n

**Ví dụ:**

* Với n=1 thì kết quả sumOfSquare(n)=0Không có bộ 3 số nào thoả mãn đề bài. Có bộ (0,0,1) thoả mãn 02 + 02+ 12= 1 nhưng không thoả mãn điều kiện 3 số phải khác nhau
* Với n=10 thì kết quả sumOfSquare(n)=1  
  Có 1 bộ (0,1,3) thoả mãn 02 + 12+ 32=10

**Đầu vào/đầu ra:**

* **[Thời gian chạy]** 0.2 giây với C++, 1.2 giây với Java/C# và 1.6 giây vs Python
* **[Đầu vào] integer n**  
  1 <= n <= 5\*10^6
* **[Đầu ra] integer**  
  Số bộ 3 số thoả mãn yêu cầu đề bài

Tại khu phố nọ có n căn nhà nằm cạnh nhau được đánh số từ 1 tới n theo chiều từ trái qua phải. Mỗi ngôi nhà được sơn một màu khác nhau (màu của ngôi nhà được thể hiện bằng một xâu ký tự gồm các ký tự chữ cái in thường, ví dụ như red, blue, green, ...

Bạn được cho 1 danh sách chứa miêu tả về vị trí các ngôi nhà thuộc 1 trong các cú pháp sau:

* A is next to the right of B: ngôi nhà màu A nằm ngay bên phải ngôi nhà màu B (giả sử ngôi nhà màu B là vị trí i thì màu A ở vị trí i+1)
* A is on the right of B: ngôi nhà màu A nằm phía bên phải ngôi nhà màu B (giả sử ngôi nhà màu B nằm ở vị trí i thì A ở vị trí i+x với x>=1)
* A is next to the left of B: ngôi nhà màu A nằm ngay bên trái ngôi nhà màu B (giả sử ngôi nhà màu B là vị trí i thì A ở vị trí i-1)
* A is on the left of B: ngôi nhà màu A nằm phía bên trái ngôi nhà màu B (giả sử ngôi nhà màu B nằm ở vị trí i thì màu A ở vị trí i-x với x>=1)
* A is in the middle of B and C: ngôi nhà A nằm chính giữa ngôi nhà màu B và ngôi nhà màu C (giả sử B ở vị trí i và C ở vị trí j thì A ở vị trí (i+j)/2 với điều kiện (i+j)%2=0)
* A is in between of B and C: ngôi nhà A nằm giữa ngôi nhà màu B và ngôi nhà màu C (giả sử B ở vị trí i và C ở vị trí j thì A ở vị trí x sao cho i<x<j hoăc j<x<i)

Dựa vào các điều kiện đã cho, bạn cần xác định được vị trí của từng căn nhà. Trong trường hợp có nhiều đáp án, bạn cần đưa ra đáp án có thứ tự từ điển nhỏ nhất.  
  
Giả sử có 2 đáp án là X = {a1, a2, ..., an} và Y = {b1, b2, ..., bn) thì đáp án X được gọi là có thứ tự từ điển nhỏ hơn Y nếu với k nào đó thì a1=b1, a2=b2, ..., ak<bk với a1, a2, b1, b2, ..., an, bn là màu của các ngôi nhà  
Ví dụ như đáp án {"green", "blue", "red"} được gọi là có thứ tự từ điển nhỏ hơn đáp án {"green", "red", "blue"} vì "blue" < "red"

**Ví dụ:**

* Với houseColors=["red", "blue"] và houseConditions=[] thì housesList=["blue", "red"]  
  Có 2 đáp án thoả mãn điều kiện đưa ra là ["red", "blue"] và ["blue", "red"] nhưng đáp án ["blue", "red"] có thứ tự từ điển nhỏ hơn
* Với houseColors=["red", "blue"] và houseConditions=["red is next to the right of blue"] thì housesList=["blue", "red"]  
  Chỉ có 1 đáp án thoả mãn điều kiện đề bài là ["blue", "red"] vì ngôi nhà màu đỏ (red) phải nằm bên phải ngôi nhà màu xanh (blue)

**Đầu vào/đầu ra:**

* **[Thời gian chạy]** 0.5 giây với C++, 1.2 giây với Java/C# và 1.6 giây với Python/Js
* **[Đầu vào] array.string houseColors**mảng chứa các xâu chỉ gồm các kí tự in thường thể hiện màu sắc của các ngôi nhà trong khu phố. Đảm bảo không có hai màu giống nhau trong danh sách  
  2 <= houseColors.length() < 10
* **[Đầu vào] array.string houseConditions**mảng chứa các xâu có cú pháp "A is next to the right of B", "A is on the right of B", "A is next to the left of B", "A is on the left of B", "A is in the middle of B and C" hoặc "A is in between of B and C" trong đó A, B, C là các màu nằm trong mảng houseColors (đảm bảo A # B # C)0 <= houseConditions.length() <= 200
* **[Đầu ra] array.string**  
  Danh sách thể hiện màu của từng ngôi nhà thoả mãn các điều kiện đầu vào có thứ tự từ điển nhỏ nhất. Trong trường hợp không có đáp án, hãy trả ra mảng rỗng