4. Bài toán điển hình.

\_ Khi nào thì sử dụng cấu trúc dữ liệu HashMap?

+ Khi bạn cần ánh xạ giữa cái gọi là khoá (key) sang giá trị của khoá đó (gọi là value).

+ Khi bạn cần một số loại dữ liệu phức tạp làm khóa.

+ Đếm tần số xuất hiện của một đối tượng (Competitive programming).

\_ Ví dụ ứng dụng của map:

+ Cho một danh sách các số điện thoại kèm theo tên của chủ thuê bao đó

=> Yêu cầu đầu vào là một số điện thoại (key), hãy đưa ra tên của chủ thuê bao (value).

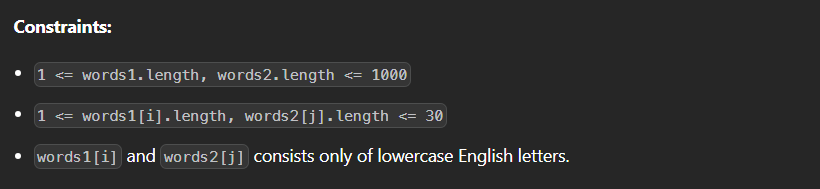
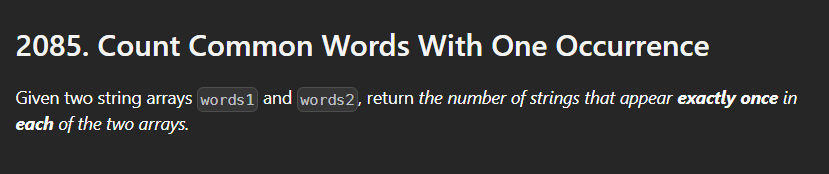
+ Cho danh sách thể hiện lịch sử đi muộn của các nhân viên một công ty nào đó => Tìm xem nhân viên (key) nào có số lần đi muộn (value) nhiều nhất?

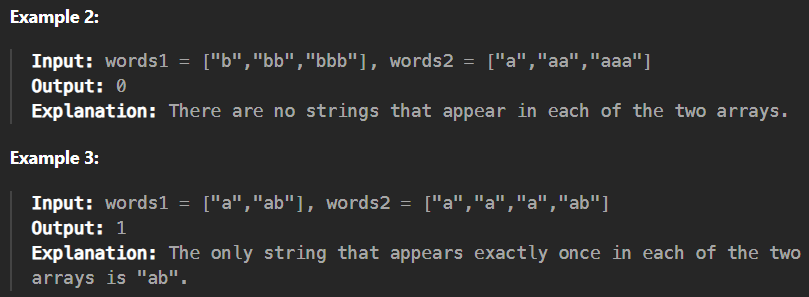
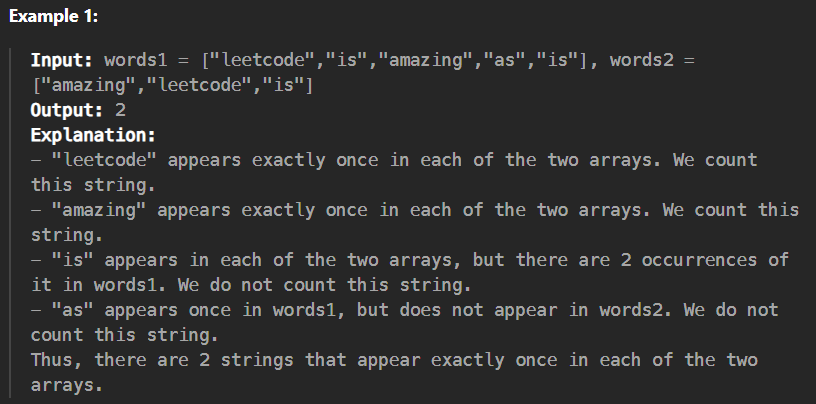
+ Cho một danh sách các IP kèm theo các domain

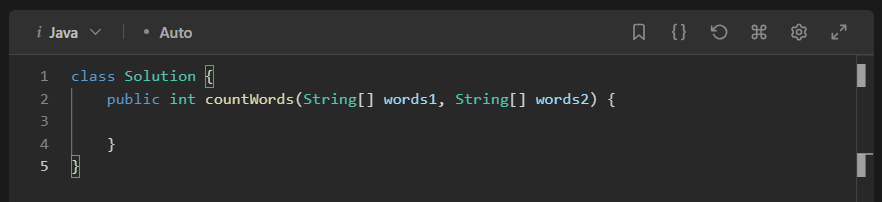
=> Trả ra ip (value) tương ứng domain (key) hoặc ngược lại trong thời gian nhanh nhất?

4.1. Sử dụng cấu trúc dữ liệu HashMap để giải quyết bài toán với các đối tượng nguyên thủy.

<https://leetcode.com/problems/count-common-words-with-one-occurrence/description/>

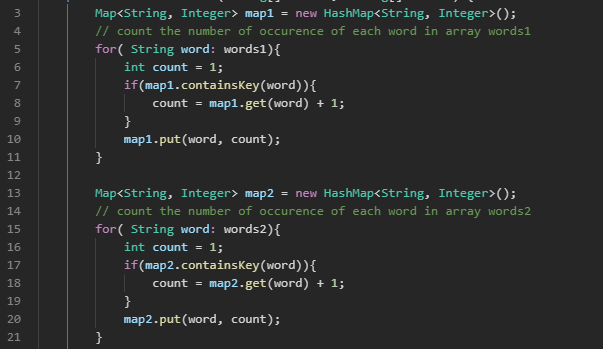




\_ Template code được cho sẵn : 

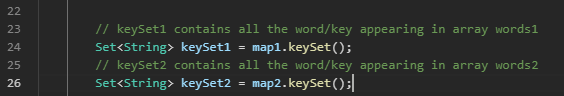
\_ Ý tưởng giải quyết bài toán bằng cấu trúc dữ liệu map :

+ Sử dụng 2 map : map1 và map2 để đếm số lần xuất hiện của mỗi từ trong mảng words1 và words2 được cho. Cả 2 map có key là từ xuất hiện trong mảng và value là số lần xuất hiện của từ đó trong 2 mảng tương ứng.



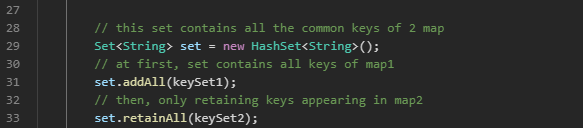
+ Sử dụng cấu trúc dữ liệu Set, tạo 2 biến keySet1 và keySet2 để chứa các từ đã xuất hiện trong mảng words1 và words2.

* Chính là các key của map1 và map2.



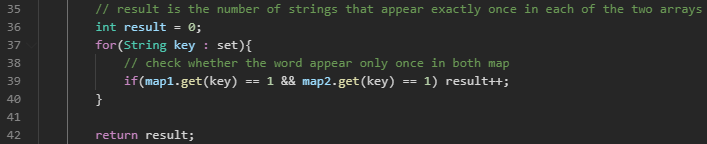
+ Sử dụng thêm 1 set để chứa các từ chung (đã xuất hiện ở cả 2 mảng).

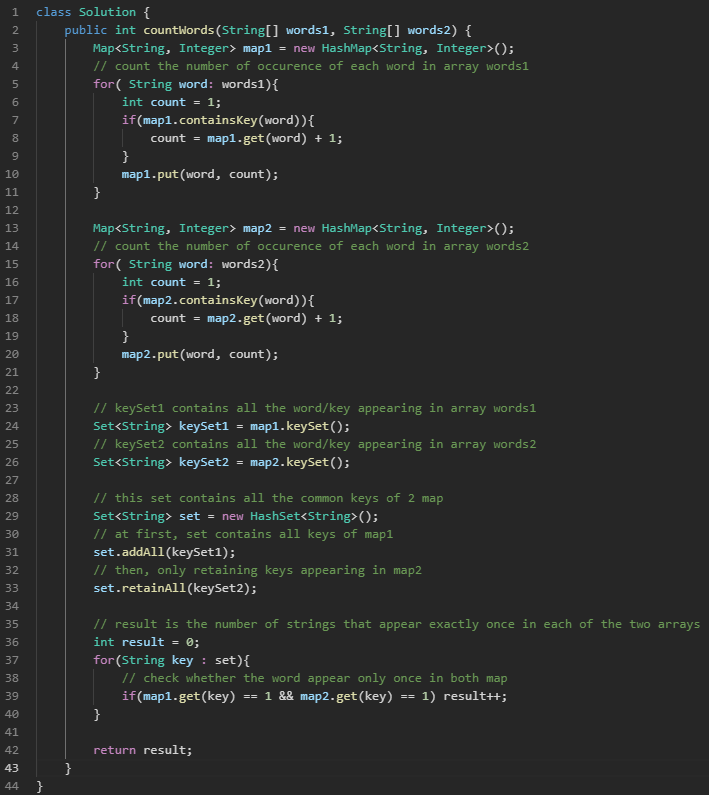
* Chính là các key xuất hiện trong cả set1 và set2.

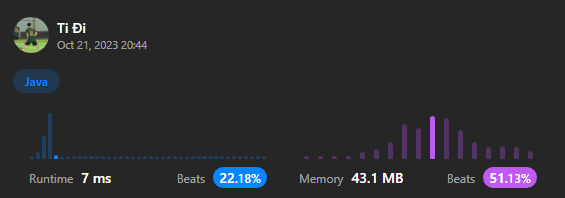


+ Cuối cùng, đếm tất cả các từ (xuất hiện trong set chung vừa tạo) mà có số lần xuất hiện trong mảng words1 và words2 đều bằng 1, trả về số lượng đếm được.

* Kết quả của bài toán được lưu ở biến result, duyệt qua tất cả key có trong set chung đã tạo, biến result được tăng khi số lần xuất hiện của key đó trong cả 2 map1 và map2 đều bằng 1.

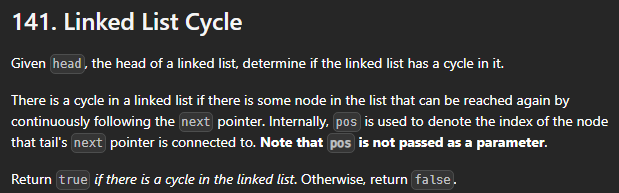
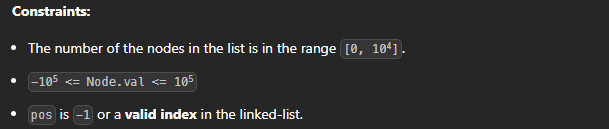


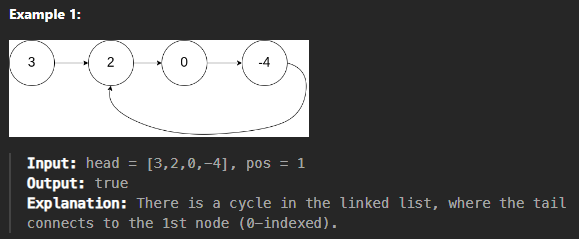
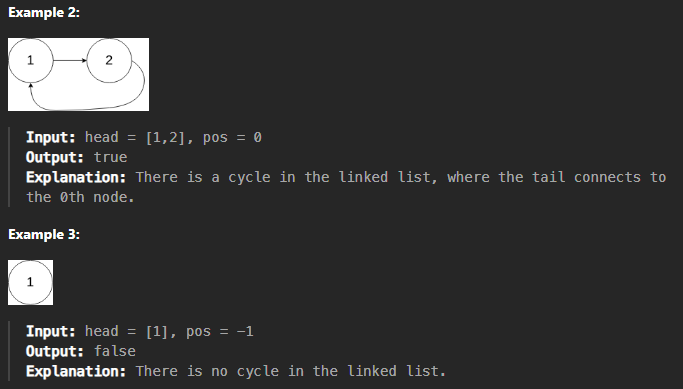
Code đầy đủ của bài toán: 

\_Với phương pháp giải bằng cách sử dụng Map kết hợp với Set, sau khi chạy bộ test case từ trình duyệt leetcode, code trên được Accept, chạy trong 7ms và sử dụng 43.1MB bộ nhớ. 

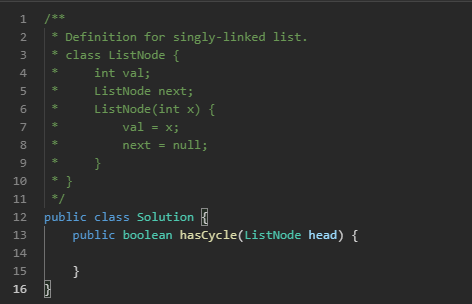
4.2. Sử dụng cấu trúc dữ liệu HashMap để giải quyết bài toán với các đối tượng tự định nghĩa.

<https://leetcode.com/problems/linked-list-cycle/description/>

\_ Template code được cho sẵn:



\_ Ý tưởng giải quyết bài toán :

+ Sử dụng 1 map có key là 1 ListNode, value của key đó đánh dấu key đó đã được thăm hay chưa. 

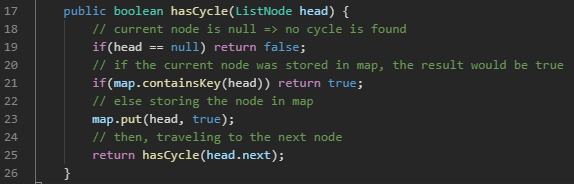
+ Sử dụng đệ quy với phương thức hasCycle có đầu vào là 1 ListNode.

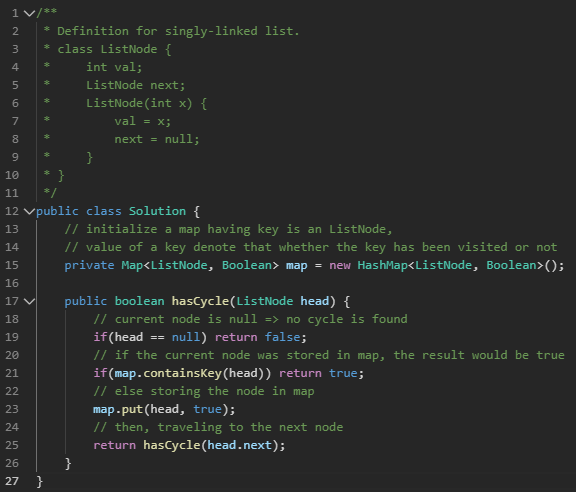
Nếu ListNode đang xét là null => Không chu trình nào được tạo.

Nếu map được tạo đã đánh dấu ListNode đang xét => ListNode đó đã xuất hiện từ trước => Chu trình được tạo.

Nếu không thì ta đánh dấu ListNode đó trong map.

Sau đó tiếp tục gọi đệ quy xét ListNode next của ListNode đang xét.



Code đầy đủ cho bài toán: 

\_Với phương pháp giải bằng cách sử dụng Map, sau khi chạy bộ test case từ trình duyệt leetcode, code trên được Accept, chạy trong 4ms và sử dụng 42.7MB bộ nhớ. 