CHỦ ĐỀ: GREEDY

NHÓM 5:

Nguyễn Quốc Huy Hoàng – 20520051

Lê Nguyễn Khánh Nam – 20520073

DOITIEN_MEDIUM

- **Hướng tiếp cận 1:** Greedy
  - Bước 1: Sắp xếp giá trị các đồng tiền giảm dần, số đồng tiền ban đầu là 0
  - Duyệt lần lượt các đồng tiền theo thứ tự đã sắp xếp, nếu số tiền hiện tại lớn hơn giá trị của đồng tiền thì ta lấy đồng tiền đó, đồng thời số tiền bị trừ đi và số đồng tiền tăng lên 1.
  - Nếu số tiền bằng 0 thì ta có kết quả, ngược lại, giải thuật greedy bị sai hoặc không có kết quả. Nhưng luôn luôn có kết quả nên trường hợp này thì giải thuật này bị sai.

  ĐPT: O(NlogN) với N là số đồng tiền

- **Hướng tiếp cận 2:** Greedy + Brute forces

  Nhận thấy khi sử dụng Greedy để giảm số tiền về một cái khoảng mốc giá trị nào đó thì sẽ không còn đúng nữa. Nên ta dùng giải thuật tham lam để giảm số tiền lại lại một mốc nào đó đủ lớn (mình chọn là 10000, vì nó lớn so với chỉ có 10 loại tiền, và mỗi loại chỉ có mệnh giá tối đa là 100), sau đó dùng vét cạn để giải quyết khi số tiền nhỏ.

  - Bước 1: Tìm đồng tiền có mệnh giá lớn nhất, nếu số tiền mà lớn hơn 10000 thì ta sẽ trừ đi mệnh giá lớn nhất đó, và số đồng tiền tăng lên 1. Bây giờ số tiền nhỏ hơn hoặc bằng 10000 và chỉ có 10 đồng tiền nên ta có thể Brute forces

  - Bước 2: Ta có hàm recursive(s, a, n) là số đồng tiền ít nhất với s là số tiền hiện có, n là số loai đồng tiền, a là mảng lưu giá trị của các đồng tiền đó. Ta có s = 0 thì kết quả là 0, nếu s < Min(a) và s != 0 thì kết quả là INF. Nếu không thì đi qua bước 3.

- Bước 3: Ban đầu kết quả là res = INF, ta duyệt mọi đồng tiền. Nếu s >= a[i] tức là có thể chọn được đồng thứ a[i], ta lấy min kết quả với trường hợp chọn thêm 1 đồng đó, nghĩa là

  res = min(res, 1 + recursive(s – a[i], a, n))

  Để tính được recursive(s – a[i], a, n) thì ta qua bước 2 để tính.

- Bước 4: Lấy kết quả tham lam lúc đầu cộng với kết quả của Brute force thì ta có kết quả cần tìm.

ĐPT: Tùy thuộc vào giá trị biến động của a[i] nên rất khó để tính. Giả sử trong trường hợp tốt thì mỗi đồng tiền chỉ chọn 1 lần thì đpt sẽ là O(N!), nên ĐPT thực tế có thể lớn hơn O(N!), nhưng khi mà a[i] lớn so với s thì có thể chỉ chọn được vài đồng tiền mà thôi, nên ĐPT lại có thể nhỏ hơn O(N!)

- **Cải tiến tiếp cận 2:**

  Nhận thấy khi mà Brute forces ta có thể phải tính giá trị của recursive(s, a, n) có cùng giá trị tham số nhiều lần, nên ta dùng một mảng để lưu lại các giá trị của hàm, vì hàm recursive chỉ có tham số s là đổi, nên ta có

  d[s] = recursive(s, a, n). Như vậy nếu như mỗi lần ta gọi lại hàm

  recursive(s, a, n) mà đã tính trước đó thì ta chỉ mất O(1) để gọi d[s]

DOITIEN_HARD

Bài này tương tự với bài trên, nhưng giới hạn số đồng tiền tăng lên. Với n <= 100 thì ta không thể nào làm như cách ở trên được, nhưng vì mục tiêu của bài toán là đạt được điểm cao nhất, nên ta sẽ cố phủ hết trường hợp mà nằm trong khả năng của mình.

- **Hướng tiếp cận 1:** Ý tưởng giống bài Medium

  Chia bài toán thành 2 trường hợp:

  - TH1: Với n nhỏ thì ta làm như bài MEDIUM. Mình chọn n = 20, vì bài medium có n <= 10 thì thuật toán trên giải quyết khá là nhanh nên để vét được nhiều test ta cho n lớn hơn 1 tí.

- TH2: Giải quyết như tiếp cận 1 của bài MEDIUM. Tuy cách này có sai sót, nhưng ngoài các này ra thì những cách đã học trước đó bao gồm Brute force, Divide & conquer không thể giải quyết được.

Giải quyết theo hướng này được 90/100

- **Hướng tiếp cận 2:** Greedy + Brute force

  - Có khả năng với n lớn (n <= 100) và giá trị của những đồng tiền đó lại nhỏ (Giá trị của đồng tiền không vượt quá 100), nên có khả năng rất nhiều đồng tiền sẽ không được sử dụng đến, vì nếu số tiền lớn thì dùng đồng tiền to, đến khi nhỏ thì chỉ cần dùng đồng tiền nhỏ, những đồng tiền trung bình sẽ có khả năng không dùng tới.

  - Đầu tiên ta dùng tham lam như cách tiếp cận 1 của bài Medium, để lấy kết quả của trường hợp có thể phải dùng tới những đồng tiền có giá trị ở giữa (**Solution 1**)

  - Ta sắp xếp giá trị của các đồng tiền tăng dần, nếu số đồng tiền lớn hơn 20 thì ta chỉ lấy 10 đồng tiền đầu tiên, và 10 đồng tiền cuối cùng. Sau đó làm như cách tiếp cận 2 (có kèm theo cải tiến) của bài Medium. Lấy 20 đồng tiền vì với n <= 20 thì brute force có khả năng giải quyết được, nếu lấy nhiều hơn thì TLE, lấy ít hơn thì độ chính xác thấp hơn (**Solution 2**)

  - Ta lấy min kết quả của solution 1 và solution 2 thì ta có được đáp án cần tìm

Giải quyết theo hướng này được 100/100