Kĩ thuật chính: Brute Force

Tóm tắt ý tưởng:

Gọi số tiền cần quy đổi là s.

mệnh giá các đồng thứ i (trong một list), là denomination[i].

số lượng đồng nhỏ nhất để quy đổi s là numOfCoin[s]

Để tìm số lượng đồng để quy đổi cho s, ta quy đổi lượng tiền s - denomination[i] ra số đồng sau đó cộng thêm 1 đồng denomination[i].

Vậy numOfCoin[s] bằng min của các ( numOfCoin[ s - denomination[i] ] + 1 ).

Mô tả giải thuật:

* B1: Input và Sort list mệnh giá denomination tăng dần.
* B2.1: Liên tục giảm tiền cần quy đổi s đi mệnh giá của đồng lớn nhất, đến khi s nhỏ hơn 2000.
* B2.2: Lưu số lần giảm vào biến kết quả res
* Giải thích B2: Giảm s đi để áp dụng brute force kịp thời gian. Chọn 2000 vì đoán rằng 2000 là đủ nhỏ để cho ra kết quả đủ tốt (Chưa chứng minh được).
* B3: Khởi tạo một list numOfCoin giá trị vô cực cho index từ 1 đến s, giá trị 0 cho index 0.
* Giải thích B3: Tạo list numOfCoin với mục đích trả về số đồng quy đổi nhỏ nhất, và index là lượng tiền cần quy đổi.
* B4: Cho m chạy từ 1 đến s. Sau đó cho i chạy từ 0 đến n - 1.
* Giải thích B4: m đại diện cho money và là index của numOfCoin, i đại diện cho index của denomination.
* B5: Nếu m < denomination[i], thoát khỏi vòng lặp biến i.
* Giải thích B5: Do denomination được sắp xếp tăng dần, một khi m đã lớn hơn mệnh giá thứ i, thì các mệnh giá thứ i+1, i+2, …, n-1 chắc chắn lớn hơn m.
* B6: Nếu m >= denomination[i]

và nếu numOfCoin[m] > numOfCoin[m - denomination[i]] + 1,

Gán numOfCoin[m] bằng numOfCoin[m - denomination[i]] + 1

* Giải thích B6 nằm ở Tóm tắt ý tưởng.
* B7: Sau khi vòng lặp ở B4 chạy xong, cộng thêm numOfCoin[s] vào res.
* Giải thích 6: Sau bước 2, một phần của s lúc input đã được quy đổi và lưu kết quả vào res. Do đó ở bước này ta cần quy đổi s (s này là s đã giảm) ra số đồng.
* B8: In kết quả res.