# las1:

Quy hoạch động.

Gọi F(i) là độ dài cấp số cộng dài nhất kết thúc tại a\_i. Kết quả bài toán là max( F(1), F(2), ..., F(n) )

Ta có: F(i) = max( F(j) + 1) với mọi j thỏa mãn: j < i và a\_j = a\_i - d.

**Nhận xét:** nếu có j1, j2 thỏa mãn điều kiện trên, nếu j1 < j2 thì F(j1) <= F(j2)

=> trong số tất cả các j thỏa mãn, ta chỉ cần quan tâm j lớn nhất Vậy F(i) = F(j) + 1 trong đó j < i, a\_j = a\_i - d và a\_j lớn nhất có thể. Với mỗi i, có thể tìm j tương ứng bằng cách chặt nhị phân.

Đpt: O(N log N)

# las2:

Quy hoạch động.

Gọi F(i, j) là độ dài dãy cấp số cộng dài nhất kết thúc tại a\_j và a\_i (j < i). Kết quả bài toán là max( F(i, j) ) với mọi j < i

Ta có: F(i, j) = max( F(j, k) + 1) với mọi k thỏa mãn: k < j và a\_j - a\_k = a\_i - a\_j hay a\_k = 2a\_j - a\_i.

**Nhận xét:** nếu có k1, k2 thỏa mãn điều kiện trên, nếu k1 < k2 thì F(j, k1) <= F(j, k2)

=> trong số tất cả các k thảo mãn, ta chỉ cần quan tâm k lớn nhất.

Vậy F(i, j) = F(j, k) + 1 trong đó k < j, a\_k = 2a\_j - a\_i và k lớn nhất có thể. Có thể tìm k bằng cách chặt nhị phân.

Đpt: O(N^2 log N)

# las3:

Nhận xét rằng một dãy cấp số cộng luôn tăng dần hoặc giảm dần. Nếu cấp số cộng giảm dần, thì khi đảo ngược nó, nó vẫn là cấp số cộng, nhưng tăng dần.

=> chỉ cần xét cấp số cộng tăng dần.

# Thuật toán O(N^2 log N):

Sort tập hợp số được cho tăng dần. Sau đó dùng thuật toán của bài las2 để giải.

# Thuật toán O(N^2):

Sort tập hợp số được cho tăng dần. Gọi dãy sau khi đã sắp xếp là dãy a. Quy hoạch động tương tự bài las2.

Gọi F(i, j) là độ dài dãy cấp số cộng dài nhất kết thúc tại a\_j và a\_i (j < i). Kết quả bài toán là max( F(i, j) ) với mọi j < i

Ta có: F(i, j) = max( F(j, k) + 1) với mọi k thỏa mãn: k < j và a\_j - a\_k = a\_i - a\_j hay a\_k = 2a\_j - a\_i.

**Nhận xét:** nếu có k1, k2 thỏa mãn điều kiện trên, nếu k1 < k2 thì F(j, k1) <= F(j, k2)

=> trong số tất cả các k thảo mãn, ta chỉ cần quan tâm k lớn nhất.

Vậy F(i, j) = F(j, k) + 1 trong đó k < j, a\_k = 2a\_j - a\_i và k lớn nhất có thể.

**Nhận xét:** Khi j cố định, cho i chạy tăng dần thì k tương ứng với cặp (i, j) này sẽ giảm dần.

=> với mỗi j cố định, có thể tìm k cho mọi i trong O(N) bằng kĩ thuật "trượt": khi i tăng dần ta cho k "trượt" giảm dần để tìm k tương ứng với mỗi i.

Đpt: O(N log N + N^2) = O(N^2)