**Ví dụ**

**Input**

3

1 2 3

l = [1, 2, 3]

Chỉ có thể đặt 3 thanh gỗ riêng biệt → Có 3 tòa tháp, mỗi tòa chỉ có chiều cao = 1

**Output**

1 3

**Input**

4

6 5 6 7

l = [6, 5, 6, 7]

Đặt thanh số 3 chồng lên thanh số 1, và thanh số 2, 4 đặt riêng → Tạo ra được 3 tòa tháp, chiều cao lần lượt là 2, 1, 1

**Output**

2 3

Thấy là, yêu cầu xây dựng số lượng tháp là ít nhất

→ Đặt chồng những thanh có cùng chiều dài lên nhau

**\*\*Số lượng tháp ít nhất: Số lượng độ dài khác nhau**

**\*\*Chiều cao cao nhất: Tháp có số lượng thanh cùng chiều dài nhiều nhất**

**Cách 1:** Sử dụng mảng đếm phân phối

cnt[x] là số lượng thanh có độ dài x.

→ Số lượng tháp ít nhất = số lượng giá trị x mà cnt[x] khác 0

→ Chiều cao cao nhất = max(cnt)

Bài toán này, với cái l[i] <= 1000 → **oke**

Nếu chiều dài lên tới 10^9 hoặc 10^18 → Không còn tốt

**Cách 2:** Sử dụng sort

Khi mà chúng ta sắp xếp (**tăng dần** hoặc giảm dần)

l1 <= l2 <= l3 <= … <= ln

→ Các độ dài bằng nhau, sẽ nằm cạnh nhau

* Nếu l[i] == l[i-1] → số lượng thanh giống nhau và giống với l[i] tăng lên 1
* Nếu l[i] != l[i-1] → số lượng thanh khác nhau tăng lên

**Giải thuật**

**Bước 1:** Đưa thông tin chiều dài các thanh gỗ vào mảng.

**Bước 2:** Sắp xếp mảng tăng dần.

**Bước 3:** Lấy thanh gỗ đầu tiên làm gốc, ta sử dụng một biến đếm số lượng thanh gỗ phân biệt, khởi tạo bằng 1 (tức xem phần tử ở đầu mảng là phần tử phân biệt đầu tiên). Biến đếm số lượng thanh gỗ có cùng chiều dài với thanh gỗ đang xét, khởi tạo bằng 1.

**Bước 4:** Lần lượt duyệt qua các thanh gỗ có chiều dài từ bé đến lớn, không tính thanh đầu tiên:

Nếu thanh gỗ này có cùng chiều dài với thanh trước đó thì ta tăng biến đếm số lượng thanh gỗ có cùng chiều dài lên 1. Đồng thời, so sánh số lượng thanh gỗ có cùng chiều dài hiện tại với số lượng lớn nhất và liên tục cập nhật lại.

Ngược lại, tăng biến đếm số lượng phần tử phân biệt lên 1 và cập nhật lại số lượng thanh gỗ có cùng chiều dài với thanh gỗ đang xét bằng 1.

**Bước 5:** In kết quả.

**Chạy thử trên test ví dụ thứ 2:**

l = [6, 5, 6, 7]

sort(l);

l = [5, 6, 6, 7]

n\_towers = 1 // số lượng tháp nhỏ nhất tạo được, số lượng độ dài khác nhau

max\_height = 1; // Chiều cao cao nhất, số lượng phần tử giống nhau nhiều nhất

cur\_height = 1; // lượng thanh gỗ có cùng chiều dài cho tới thời điểm hiện tại

**i = 1:**

l[i] != l[i-1] (6 != 5)

→ n\_towers += 1 → n\_towers = 2

cur\_height = 1

**i = 2:**

l[i] == l[i-1]

→ cur\_height += 1 → cur\_height = 2

max\_height = max(cur\_height, max\_height) = 2

**i = 3:**

l[i] != l[i-1] (7 != 6)

→ n\_towers += 1 → n\_towers = 3

cur\_height = 1

→ max\_height = 2

n\_towers = 3

**Mã giả**

**read(n);** O(1)

**for i = 0 to n-1:** O(n)

**read(l[i]);**

**sort(l);** O(nlogn)

**n\_towers = 1;**

**max\_height = 1;**

**cur\_height = 1;**

**for i = 1 to n-1: {** O(n)

**if l[i] == l[i-1]:**

**cur\_height += 1;**

**max\_height = max(max\_height, cur\_height);**

**else:**

**n\_towers += 1;**

**cur\_height = 1;**

**}**

**print(max\_height, n\_towers);**

**Độ phức tạp:**

Không gian: **O(n)**

Thời gian: **O(nlogn)**