Nhóm: 10

**Đốn cây**

**Abstraciton:**

* Phát biểu lại bài toán như sau:
* Cho 1 số nguyên dương n.
* Cho 1 mảng gồm n số nguyên dương h1, h2,…,hn (độ dài của h không vượt quá 10­6).
* Tìm và in ra tất cả các số nguyên i là vị trí của phần tử trong mảng n sao cho các hi phải lớn hơn các số ở bên phải (tại vị trí i<j<i+hi) nếu i lớn hơn 0 hoặc lớn hơn các số ở bên trái (tại vị trí i-hi<j<i) nếu i bé hơn 0, các số bé hơn hi sẽ bị đánh dấu, nếu các phần tử trong mảng n được đánh dấu hết (trừ hi) thì kết thúc.

**Pattern Recognition:**

* Dạng bài toán: tối ưu hóa quy hoạch động.
* Hướng tiếp cận: tìm kiếm tuyến tính.
* Đặc điểm nhận biết: Tìm một số nguyên.

**Algorithm Design:**

* Bước 1: Chuẩn bị
* Ta sẽ xây dựng hai mảng L[] và R[], trong đó L[i] là vị trí j nhỏ nhất mà bị cây i làm đổ nếu đẩy về bên trái, tương tự với R..
* Để tính L[] ta duy trì một stack chứa các chỉ số tăng dần. Trước khi thêm một cây i mới vào, các cây bị nó trực tiếp làm đổ sẽ bị pop ra, đồng thời ta cập nhật L[i].
* Bước 2: Quy hoạch động
* Gọi F(i) là số cây cần phải đổ nhỏ nhất để các cây có chỉ số 1 … i đều đổ.
* Để tính F(i) cần xét 2 trường hợp:
* Nếu ta đẩy cây i qua trái: F(i)=min[F(j−1)+1] với L[i] ≤ j ≤ i (1).
* Nếu cây i bị đẩy qua phải bởi cây j F(i)=min[F(j−1)+1] với 1≤ j ≤ i và R[j] ≥ i (2).
* Có thể dễ dàng tính các F[] trong O(N2). Có thể dùng các cấu trúc dữ liệu quản lí đoạn để giảm xuống O(NlogN).
* Ta có thể sử dụng stack để giảm độ phức tạp xuống O(N).
* Để xử lí ***(1)***, ta cài đặt được ngắn gọn như thế này:

với

* Để xử lí ***(2)*** ta sẽ sử dụng một ***stack*** để lưu các vị trí có ***R[]*** giảm dần, đồng thời luôn duy trì sao cho giá trị ở ***top*** của ***stack*** luôn là tốt nhất. Chú ý là với và thì . Như vậy nếu tại mỗi bước ta ***pop*** các vị trí ***j*** có ra khỏi stack, thì sẽ luôn duy trì được tính chất của ***stack*** vì lúc này đảm bảo được ***R[i]*** là nhỏ hơn các ***R[]*** đang ở trong stack, đồng thời nếu ***F(i−1)*** không tốt bằng giá trị ở đầu ***stack*** thì ta sẽ không đẩy ***i*** vào (để đảm bảo giá trị ở top luôn là tốt nhất).

**Time Complexity:** O(NlogNN)