Dưới đây là cách giải thích phương pháp tối ưu hóa thuật toán bằng tiếng Việt:

**Bài toán**

Chúng ta cần tính giá trị lớn nhất của các giá trị nhỏ nhất cho tất cả các khoảng con của một mảng có độ dài từ 1 đến N.

**Phương pháp sử dụng Stack đơn điệu**

**1. Tìm giới hạn bên trái và bên phải của từng phần tử**

* **Giới hạn bên trái (leftLimit):** Với mỗi phần tử arr[i], ta cần tìm chỉ số của phần tử nhỏ hơn gần nhất nằm bên trái của nó. Nếu không có phần tử nào nhỏ hơn, thì giới hạn bên trái là 0.
* **Giới hạn bên phải (rightLimit):** Tương tự, với mỗi phần tử arr[i], ta tìm chỉ số của phần tử nhỏ hơn gần nhất nằm bên phải của nó. Nếu không có phần tử nào nhỏ hơn, thì giới hạn bên phải là N + 1.

**2. Sử dụng Stack đơn điệu để tìm các giới hạn**

* **Giới hạn bên trái:**
  + Sử dụng stack để duyệt mảng từ trái sang phải.
  + Với mỗi phần tử arr[i], loại bỏ các phần tử trong stack mà có giá trị lớn hơn hoặc bằng arr[i] vì chúng không thể là giới hạn bên trái cho arr[i].
  + Nếu stack không rỗng sau khi loại bỏ, đỉnh của stack chính là giới hạn bên trái của arr[i]. Nếu stack rỗng, không có phần tử nhỏ hơn nào bên trái.
* **Giới hạn bên phải:**
  + Sử dụng stack để duyệt mảng từ phải sang trái.
  + Quy trình tương tự như tìm giới hạn bên trái.

**3. Tính giá trị lớn nhất của các giá trị nhỏ nhất cho mỗi độ dài khoảng con**

* **Tính độ dài khoảng con:** Đối với mỗi phần tử arr[i], độ dài của khoảng con mà arr[i] là giá trị nhỏ nhất được tính bằng rightLimit[i] - leftLimit[i] - 1.
* **Cập nhật giá trị lớn nhất cho mỗi độ dài:** Sử dụng độ dài khoảng con để cập nhật giá trị lớn nhất của các giá trị nhỏ nhất cho mỗi độ dài.
* **Điền các giá trị còn thiếu:** Để đảm bảo mọi độ dài từ 1 đến N đều có giá trị, cập nhật giá trị lớn nhất cho các độ dài không bị bỏ qua bằng cách sử dụng giá trị lớn nhất của các độ dài lớn hơn.

**4. In kết quả**

* Xuất giá trị lớn nhất của các giá trị nhỏ nhất cho từng độ dài khoảng con từ 1 đến N.

**Mã nguồn**

cpp

Sao chép mã

#include<bits/stdc++.h>

#define ll long long

#define endl '\n'

using namespace std;

const int MAXN = 200005;

int arr[MAXN];

int leftLimit[MAXN], rightLimit[MAXN];

int result[MAXN];

int main() {

ios::sync\_with\_stdio(false);

cin.tie(0);

int n;

cin >> n;

for (int i = 1; i <= n; i++) {

cin >> arr[i];

}

stack<int> st;

// Tìm giới hạn bên trái

for (int i = 1; i <= n; i++) {

while (!st.empty() && arr[st.top()] >= arr[i]) {

st.pop();

}

leftLimit[i] = st.empty() ? 0 : st.top();

st.push(i);

}

while (!st.empty()) st.pop();

// Tìm giới hạn bên phải

for (int i = n; i >= 1; i--) {

while (!st.empty() && arr[st.top()] >= arr[i]) {

st.pop();

}

rightLimit[i] = st.empty() ? n + 1 : st.top();

st.push(i);

}

// Khởi tạo mảng kết quả

for (int i = 1; i <= n; i++) {

result[i] = 0;

}

// Tính giá trị lớn nhất của các giá trị nhỏ nhất cho mỗi độ dài

for (int i = 1; i <= n; i++) {

int length = rightLimit[i] - leftLimit[i] - 1;

result[length] = max(result[length], arr[i]);

}

// Cập nhật các giá trị còn thiếu

for (int i = n - 1; i >= 1; i--) {

result[i] = max(result[i], result[i + 1]);

}

// Xuất kết quả

for (int i = 1; i <= n; i++) {

cout << result[i] << " ";

}

cout << endl;

return 0;

}

**Tóm tắt**

Phương pháp sử dụng stack đơn điệu giúp giảm độ phức tạp của bài toán từ O(N2)O(N^2)O(N2) xuống O(N)O(N)O(N), làm cho thuật toán có thể xử lý được mảng với kích thước lên đến 200,000.