First Element to Left Guide

by fbrunodr

May 16, 2023

1 Introduction

Guide to use `firstElementToLeft` and `firstElementToRight` implemented in https://github.com/fbrunodr/CompetitiveProgramming.

2 Motivation

Given a zero indexed array a of size n, output an array b of size n such that b_i is the largest index $< i$ such that $a_{b_i} > a_i$. If there is no such index, $b_i = -1$. In plain English, for each array element a_i, we want to find the index of the first element to the left that is greater than a_i. A naive approach to this problem would be, for each index, to iterate over the elements to the left until we find the first one that is larger. This takes $O(n^2)$. This problem can be solved in $O(n)$ using the following observation: Let b_{i-1} be as described above. Then, for all indexes $b_{i-1} < k < i-1$ we have $a_k \leq a_{i-1}$. If $a_{i-1} > a_i$ then $b_i = i-1$. Else, we have $a_k \leq a_{i-1} \leq a_i$ for all $b_{i-1} < k < i-1$. So we don't have to check whether $a_k > a_i$ for all $b_{i-1} < k < i-1$. This allows for the following solution:

```
stack<int> st;
vector<int> b(n);
for(int i = 0; i < n; i++){
    while(!st.empty() && a[st.top()] <= arr[i])
        st.pop();
    if(st.empty())
        b[i] = -1;
    else
        b[i] = st.top();
    st.push(i);
}
return b;
```

3 First Element to Left

The idea in the previous section can be generalized. Let $\oplus$ be a binary relation (as in https://en.wikipedia.org/wiki/Binary_relation). For each element a_i we want the largest index $j < i$ such that $a_j \oplus a_i$ is true. If no such index exists, then $b_i = -1$. If $a_{i-1} \oplus a_i$ returns true, then $b_i = i-1$. Otherwise, can we skip all indexes $b_{i-1} < k < i-1$? That is, given that $a_{i-1} \oplus a_i$ is false and $a_k \oplus a_{i-1}$ is false, can we guarantee that $a_k \oplus a_i$ is also false (so we can skip those checks)?

Let $\ominus$ be the opposite of $\oplus$, that is $x \ominus y = !(x \oplus y)$. We can skip the checks described above as long as $(a_k \ominus a_{i-1})$ and $(a_{i-1} \ominus a_i) \Rightarrow (a_k \ominus a_i)$. That is, the operator $\ominus$ is transitive. So we can use the solution described in the previous section and implemented in the repository as long as the negative of $\oplus$ is transitive. Some examples of such $\oplus$ are $<, \leq, >, \geq, \neq$, `not related`, etc.