

The Problem: Counting Monotone Subsequences

Given a finite sequence $X = (x_1, \dots, x_n)$ of distinct values in an ordered domain (such as the integers) compute N , the number of monotone increasing subsequences of x .

We say “distinct” because the problem, as stated, is ambiguous if there could be duplicate values. If duplicates are allowed, we can reduce the problem to the unique version by using the indices as tie-breakers; if $x_i = x_j$, we say x_i is smaller than x_j if $i < j$.

Subproblems

For each $1 \leq i \leq n$, let N_i be the number of monotone increasing subsequences of $X_i = (x_1, \dots, x_i)$ which end at x_i .

The Basic Algorithm

We solve for N_i in order of increasing i . If Y is a monotone subsequence of X_i which ends at x_i , then $Y = Z + (x_i)$, where Z is a monotone increasing subsequence of X_{i-1} . Z could be empty, but otherwise, the last term of Z is x_j for some $j < i$ such that $x_j < x_i$. N_i is the number of such choices of Z . Our final output is obtained by adding the N_i .

$$N_i = 1 + \sum_{(j < i) \wedge (x_j < x_i)} N_j \quad (1)$$

$$N = \sum_{1 \leq i \leq n} N_i \quad (2)$$

Our basic algorithm uses (1) and (2) to solve the problem.

Basic Algorithm

```
1: for  $i \leftarrow 1$  to  $n$  do
2:    $N_i \leftarrow 1$ 
3:   for  $j \leftarrow 1$  to  $i - 1$  do
4:     if  $x_j < x_i$  then
5:        $N_i \leftarrow N_i + N_j$ 
6:     end if
7:   end for
8: end for
9:  $N \leftarrow 0$ 
10: for  $i \leftarrow 1$  to  $n$  do
11:    $N \leftarrow N + N_i$ 
12: end for
```

The time complexity of the basic algorithm is $\Theta(n^2)$.

Example

Let $n = 10$, and $X = (3, 6, 8, 2, 0, 9, 4, 1, 5, 7)$. Our computation is as follows:

$$\begin{aligned} N_1 &= 1 &= 1 \\ N_2 &= 1 + N_1 &= 2 \\ N_3 &= 1 + N_1 + N_2 &= 4 \\ N_4 &= 1 &= 1 \\ N_5 &= 1 &= 1 \\ N_6 &= 1 + N_1 + N_2 + N_3 + N_4 + N_5 &= 10 \\ N_7 &= 1 + N_1 + N_4 + N_5 &= 4 \\ N_8 &= 1 + N_5 &= 2 \\ N_9 &= 1 + N_1 + N_4 + N_5 + N_7 &= 8 \\ N_{10} &= 1 + N_1 + N_2 + N_4 + N_5 + N_7 + N_8 + N_9 &= 20 \\ N &= N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 + N_9 + N_{10} &= 53 \end{aligned}$$

Range Query Structures

We can speed up the basic algorithm by using a *range query structure* to hold certain subtotals of the $\{N_i\}$.

A range query structure consists of the following parts.

1. An array A of n values. In our example, A will be a one dimensional array, but range query structures can have any dimension.
2. A designated set of subarrays of A , of various sizes, called *ranges*. (In the one dimensional case, these are intervals.) There must be n ranges of size one, one for each entry in the array. It is normal (but not absolutely necessary) that each range which has more than one entry is the exact union of some small number of smaller ranges. (In our example, each range of size more than one is the exact union of two smaller ranges.)
3. Each range has a *total*, which is the sum of the values of the entries in the range. That is, if R is a range, the total of R is $T[R] = \sum_{i \in R} A[i]$.

The following operations are available for a range query structure.

1. *Initialize*. The initial values of the array are given. All range totals are calculated.
2. *Query*. A subset Q of index values is specified. (In our example, Q is an interval, which may not be one of the ranges.) The value of the query is the sum of the entries with indices in Q .
3. *Update*. A value $A[i]$ is changed for some index i . The totals for all ranges which contain i must be be updated.