Computing the Spans in a sequence

Jayadev Misra

1/22/2001

The following problem has been treated as an example in a book by Goodrich and Tamassia [1, section 3.5]. Given is a sequence of integers, $p_0, \ldots, p_{N-1}$. Henceforth, assume that p_0 exceeds all other p_is. For each i, $1 \le i < N$, define its *span* $s.i$ to be the length of the longest segment ending at i whose elements are at most p_i in value. That is, $p_{i-s.i} > p_i$ and for all k, $i-s.i < k \le i$, $p_k \le p_i$. Since, by assumption, p_0 exceeds p_i, $1 \le i < N$, $s.i$ is well defined. Henceforth, the range of i is given by $1 \le i < N$ unless stated otherwise.

Define $f.i$ to be $i - s.i$; it is the largest index preceding i such that $p_{f.i} > p_i$. That is,

(F1) $f.i < i$,
(F2) $p_{f.i} > p_i$, and
(F3) $\langle \forall k : f.i < k \le i : p_k \le p_i \rangle$

Since p_0 exceeds all p_is, $f.i$ is well defined for all i. In this note, we suggest a linear algorithm for computing all $f.i$s (and, hence, all $s.i$s).

Index $f.i$ is strictly smaller than i (from F1); hence, f induces a tree structure over the indices i, $0 \le i < N$: the root is 0 and $f.i$ is the father of i. The algorithm is based on the following observation. For any i, $0 \le i < N$, define the sequence of *ancestors* of i to be

$$f^0.i, f^1.i, f^2.i, \ldots 0,$$

where $f^0.i$ is i and $f^{t+1}.i = f.(f^t.i)$. Note that an index is its own ancestor, and the sequence of ancestors is strictly decreasing, from (F1).

Observation Let j be the first ancestor of $i-1$ such that $p_j > p_i$. Then $f.i = j$.

Proof: We show that j satisfies the definition of $f.i$, i.e., substituting j for $f.i$ in (F1, F2, F3) we show:

1. $j < i$,
2. $p_j > p_i$, and
3. $\langle \forall k : j < k \le i : p_k \le p_i \rangle$.

1. Proof of $j < i$:
   We show that all ancestors of $i-1$ precede i, i.e., $f^t.(i-1) < i$ for all t, whereever $f^t.(i-1)$ is defined. Proof is by induction on t. First, $f^0.(i-1) < i$ because $f^0.(i-1) = i-1$. Next, suppose $f^{t+1}.(i-1)$ is defined. From the induction hypothesis, $f^t.(i-1) < i$ and, from (F1), $f^{t+1}.(i-1) = f.(f^t.(i-1)) < f^t.(i-1)$; hence, we have $f^{t+1}.(i-1) < i$.

2. Proof of $p_j > p_i$: From the definition of j.

3. Proof of $\langle \forall k : j < k \le i : p_k \le p_i \rangle$:
   Any k smaller than i, $k \ne 0$, lies between two adjacent ancestors of $i-1$. That is, for all k, $j < k < i$, $f.s < k \le s$, for some ancestor s of $i-1$.

   For $k = i$, $p_k \le p_i$ is immediate. Next, we establish that $p_k \le p_i$ for any k, $j < k < i$.

   $p_k \le p_s$, use $f.s < k \le s$ and substitute s for i in condition (F3)
   $p_s \le p_i$, definion of j:
   value at any ancestor of $i-1$ prior to j is at most p_i,
   s is such an ancestor
   $p_k \le p_i$, from above two □

Program The following program is immediate from the observation.

```
$i := 1$;
while  $i < N$  do
   {compute $f.i$ using the observation}
   $j := i - 1$;
   while  $p_j \le p_i$  do  $j := f.j$  enddo ;
   $\{p_j > p_i\}$
   $f.i := j$;  $i := i + 1$
enddo
```

Analysis of the Execution Time We show that the execution time of this algorithm is linear in N. The outer loop is executed N times; hence the two assignment statements in the outer loop consume time proportional to N. Next, we count the number of assignments executed in the inner loop. Note that the number of tests executed in the inner loop is one more than the number of assignments executed in it, for each iteration of the outer loop; so, the total number of executed tests, $p_j \le p_i$, is N more than the number of executed assignments $j := f.j$.

To count the number of executed assignments in the inner loop, associate an integer *cost* with each index. Initially, all indices have zero cost. The assignment $j := f.j$ is accompanied by increasing the cost of j by 1. Then, the number of executions of $j := f.j$ is the sum of all costs. We show that

Claim: cost of any index is at most 1.

Therefore, the number of executions of $j := f.j$ is at most N.

We prove the claim by first annotating the program with assertions. In the annotation predicate $q(k)$ stands for

all ancestors of k have zero cost.

We do not show the annotation for the inner loop; it is discussed afterwards. An invariant of the outer loop —cost of i is zero— is not proven explicitly; it follows from the initial condition (all costs are zero) and that costs are increased in an iteration only for indices below i.

```
  $\{q(0)\}$
$i := 1;$
  $\{q(i-1)\}$
**while** $i < N$ **do**
    $\{q(i-1)\}$
  $j := i - 1;$
    $\{q(j)\}$
  **while** $p_j \leq p_i$ **do** increase cost of $j$ by 1; $j := f.j$ **enddo** ;
    $\{q(j)\}$
    $\{$cost of $i$ is zero, $q(j)\}$
  $f.i := j;$
    $\{q(i)\}$
  $i := i + 1$
    $\{q(i-1)\}$
**enddo**
```

Next, we show that $q(j)$ is an invariant of the inner loop. As the annotation shows, the cost of any j is at most 1, thus proving the claim.

```
  $\{q(j)\}$
  $\{j$ has zero cost, $q(f.j)\}$
increase cost of $j$ by 1; $j := f.j$
  $\{j$'s cost is 1, $q(j)\}$
```

Program for Computing the Spans The following program can be used to compute the spans $s.i$ directly.

```
$s.0 := 1;$
**for** $i := 1$ to $N - 1$ **do**
  $j := i - 1;$
  **while** $p_j \leq p_i$ **do** $j := j - s.j$ **enddo** ;
  $s.i := i - j$
**endfor**
```

References

[1] Michael T. Goodrich and Roberto Tamassia. *Data Structures and Algorithms in Java.* John Wiley and Sons, Inc., 1998.