Maximum Segment Sum

Consider a constant array A of N integers. E.g.

$$A: \begin{array}{|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} \\ \hline -2 & +2 & -1 & +3 & -3 & +2 & +2 & -1 \\ \hline \end{array}$$

$$\uparrow i = 1 \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{} \uparrow k = 7$$

A segment is a region of the array defined by two integers (i, k) with $0 \le i \le k \le N$.

The sum of a segment (i, k) is defined as

$$ss(i, k) \triangleq \sum_{j=i}^{k-1} A(j)$$

Note that a segment may be empty ($i = k$) and the sum of any empty segment is 0.

The problem is to find the largest sum of all the segments

$$f = \langle m' = (\max i, k \mid 0 \le i \le k \le N \cdot ss(i, k)) \rangle$$

For the example above, what is m'?

First solution tried all segments and hence time increased as the square of N.

Then someone found a clever algorithm so that time was proportional to $N \log_2 N$.

Finally David Gries was shown the problem and, by considering the invariant, quickly arrived at a solution with time proportional to N.

Rewrite f as

$$f = \langle m' = (\max k \mid 0 \le k \le N \cdot msse(k)) \rangle$$

where $msse$ stands for "maximum sum segment ending at"

$$msse(k) \triangleq (\max i \mid 0 \leq i \leq k \cdot ss(i,k))$$

Now we can do a linear search to solve f. The invariant is found by replacing a constant N with a variable n.

$$\begin{aligned} I &: 0 \leq n \leq N \ \wedge\ m = (\max k \mid 0 \leq k \leq n \cdot msse(k)) \\ g &= (\langle I \rangle \Rightarrow f) \\ f &\sqsubseteq n, m := 0, 0\ ;\ g \\ g &\sqsubseteq \mathbf{while}\ n < N\ \mathbf{do}\ (\ n := n + 1; \\ &\qquad\qquad\qquad\qquad m := m \max msse(n)\ ) \end{aligned}$$

[Exercise: Prove the above refinements in detail.]

But of course this is incomplete because we haven't given an algorithm to compute $msse(n)$.

We could compute $msse(n)$ with an inner loop — this leads to the slow algorithm.

Better. Assign a variable p to *track* the $msse(n)$.

[Remember the tracking variable we used in the 'slightly faster (and smaller) square root' algorithm?]

I.e. add another conjunct to the invariant

$$p = msse(n)$$

Now when n changes, so must p.

$$\begin{aligned}
I \quad : \quad & \quad 0 \le n \le N \\
& \wedge\ m = (\max k \mid 0 \le k \le n \cdot msse(k)) \\
& \wedge\ p = msse(n)
\end{aligned}$$

$$\begin{aligned}
g \ &= \ \langle I \rangle \Rightarrow f \\
f \ &\sqsubseteq \ n, m, p := 0, 0, 0 \ ; \ g \\
g \ &\sqsubseteq \ \textbf{while}\ n < N\ \textbf{do}\ (\ n, p := n+1, msse(n+1) \ ; \\
& \qquad\qquad\qquad\qquad\quad m := m \max p \ )
\end{aligned}$$

Is this progress? Yes, if we can compute $msse(n+1)$ from p

How does $msse(n+1)$ relate to $msse(n)$?

Suppose the maximum sum segment ending at $n+1$ is not an empty segment.

- Then the $msse(n+1)$ will be $A\,(n)$ plus the sum of some segment ending at n.
- Which one? The largest of course. Otherwise $msse(n+1)$ would not be maximal. So in this case
$$msse(n+1) = A(n) + msse(n)$$

Suppose the maximum sum segment ending at $n+1$ is empty.

- Then $msse(n+1) = 0$

So $msse(n+1) = 0 \max\, (A\,(n) + msse(n))$.

We get

$$f \sqsubseteq n, m, p := 0, 0, 0\ ;\ g$$
$$g \sqsubseteq \textbf{while}\ n < N\ \textbf{do}\ (\ n, p := n+1, 0 \max\, (A(n) + p)\ ;\ m := m \max p\ )$$

This is another example of a data transformation. We augmented the state space by adding a variable "p" and adding to the invariant a constraint that relates its value to the values of the other variables.

Challenge: Further modify this algorithm to find i and k such that $m = ss(i, k)$.