Truffle Search

Isaac Simon

January 15, 2021

First it is important to note this program takes a square matrix of n elements, with that framing in mind, my program runs in O(n). The algorithm I use is a dynamic algorithm, and needs $2n + 2\sqrt{n}$ extra space.

So the first part of the program is of course reading the input into an internal array of arrays of integers. This part takes $n + 2\sqrt{n}$ and honestly doesn't warrant much further discussion. The only thing I could add is that, I think knowing the input is square, one could cut this part down to $n + \sqrt{n}$.

The next part is where my algorithm begins. First we need 2 more matrices of about the same size as the input. Lets call the input A, one of the new matrices $sums$, and the last of the three $backtrace$. I said about the same size earlier because $sums$ has another row to avoid bugs. In essence this is the controlling recurrence:

$$sums[i][j] = \Big\{ max(sums[i-1][j-1], sums[i-1][j], sums[i-1][j+1]) + A[i][j]$$

$$backtrace[i][j] = \Big\{ k \quad \text{for } k \text{ such that: } sums[i-1][k] = max(sums[i-1][j-1], sums[i-1][j], sums[i-1][j+1])$$

This recurrence is in a nested loop such that the above recurrence is used on all of the the elements of A. I can actually get the above recurrence even in a double loop to only take $2n - 2\sqrt{n}$ comparisons. The minus portion is from the fact that you have to not make the left comparisons if you are in the left most column and not make the right comparisons in the right most column.

The next 2 parts are fairly simple. First we search the last row of sums and get the index of the maximum possible sum. This is just a linear search of $\sqrt{n}$ elements.

Second we start returning the answer, but we end up recovering the optimal choices in the opposite order we would need to make them. Our last choice is the index (lets call this index key) of the optimal sum we found earlier and the choice before that is at $backtrace[\sqrt{n} - 1][key]$ and we keep using the recurrence:

$$key = \Big\{ backtrace[i+1][key]$$

To recover all of the choices as we iterate down through i. Note that i starts at $i = \sqrt{n} - 2$ for a 0 index matrix as the first key value is not from this recursion. The optimal choices are the keys as above.

In conclusion my program is O(n) as there is no phase which produces a n dependent term with greater magnitude than n.