Hitting Rock Bottom

2016-10-20 Thu

Finding the global minimum—the single smallest element—in an unsorted array of numbers requires $\Omega(n)$ time. A **local** minimum is a number that is smaller than its neighbour(s). (For $n > 1$, the first and last elements have only one neighbour each and so it's "easier" for them to be local minima.) A local minimum can be found more quickly.

For example, the three local minima in the following array are underlined: $[8, \underline{2}, 3, 7, 9, \underline{5}, 6, 12, \underline{10}]$.

1. Give an efficient algorithm to find a **local** minimum of an array of n distinct numbers. (You may assume $n > 1$ if needed.) **HINT:** try to quickly find a portion of the larger array in which there's guaranteed to be at least one local minimum.

2. Give and briefly justify a good asymptotic bound on the runtime of your algorithm.

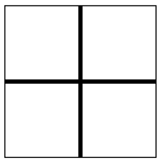
3. Sketch the key points in a proof that your algorithm is correct. (You may find it useful to establish and take advantage of this condition: "the leftmost and rightmost elements of the portion of A under consideration are smaller than any neighbours they have outside the portion of A under consideration" or to put it another way "I can check if the leftmost and rightmost elements are local minima without having to consider anything outside the portion of A under consideration".)

1 Rock Bottom Tiles

Now, consider an $n \times n$ 2-dimensional array of distinct numbers. A local minimum is still a number smaller than its neighbour(s); however, an element now neighbours the four elements above, below, to the left, and to the right. (On the edges and corners, an element has fewer neighbours and so it's "easier" for it to be a local minimum.)

1. Give an example of such a 2-dimensional array with $n = 5$ and circle all the local minima.
2. Give an efficient algorithm to find a **local** minimum in such a 2-dimensional array. (You may assume $n > 1$ if needed.)

   HINT: Consider the set of array elements down the two middle lines of the array, in this shape:

   The smallest of these elements tells you something about where a local minimum might be. It will likely help to compare against your example!
3. Give and briefly justify a good asymptotic bound on the runtime of your algorithm.
4. Sketch the key points in a proof that your algorithm is correct. (It may help to note that in the inital array of n distinct numbers, there is always at least one local minumum because the global minimum is also guaranteed to be a local minimum.)