CS 577 HW 2, P4

William Jen, Matt Henricks

4. (a) We can prove this via contradiction. Suppose the algorithm does not find a local maximum in a $n \times n$ grid, but finds some other point (i, j). Since this is not a local maximum, this point (i, j) must have a neighbor that is greater than it, by definition. However, the algorithm's terminating condition requires that you select the larger neighbor before you can terminate. Because these conditions are contradictory, the algorithm can only terminate on a local maximum.

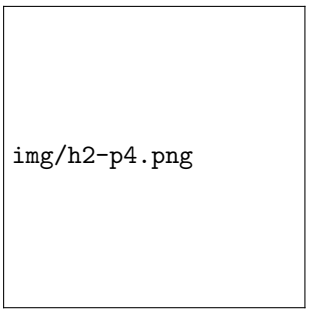
   An example where this algorithm has a worst case of $\Omega\left(n^2\right)$ is a grid of the following:

$$\begin{array}{cccccc} 0 & 0 & 2n & 2n+1 & \dots & k^2-n \\ 1 & 0 & 2n-1 & 0 & \dots & k^2-n+1 \\ 2 & 0 & 2n-2 & 0 & \dots & k^2-n+2 \quad \text{for some } 0<k<n. \\ \dots & \dots & \dots & \dots & \dots & \dots \\ n & n+1 & n+2 & 0 & \dots & k^2 \end{array}$$

   If one were to visualize this, it would be a snake, with one column of smaller values. Thus, the number of elements to traverse would be equal to $n\left(\frac{n}{2}\right)$, or $\Omega\left(n^2\right)$.

   (b)
   - Find the global maximum of the boundary and the middle row and column.
   - From there, check the following two cases:
     - Is this maximum a local maximum? Yes - done!
     - A bordering quadrant holds a point greater than the global maximum. If so, select that quadrant and recurse.

   (c)
   - Base case: 3×3 grid. Searching along boundary for global maximum will, by definition, yield a local maximum. The algorithm searches every single element in a grid of this size, since every element is on a quadrant boundary.
   - Inductive hypothesis: If the algorithm correctly finds a local minimum for a $k \times k$ grid, then does the algorithm correctly find a local minimum for a $(k+1) \times (k+1)$ grid?
   - Inductive case: The global maximum on the boundary can fall in any of the following 19 points:



   Naturally, if any of these points are local maxima, then the algorithm is complete, and thus $k+1$ size holds. However, we are more interested in the reduction case. Thus, we assume that the boundary point has a neighbor greater than it.

   Points 3, 9, 10, 11, and 16 may be ignored, as they are immediately be local maxima. Looking at the points who border a single quadrant (pts 2, 4, 6, 8, 12, 14, 16, and 18), we know that we will recurse into a new problem size of $k/2$. Through the inductive hypothesis, we know that the algorithm will correctly find a local maximum since $k/2 < k$. Even still for the ones bordering two quadrants, selecting the quadrant with the larger neighbor will reduce the problem size to $k/2$. It still proves itself correct.

   However, we must prove the claim that there exists a local maximum within the selected quadrant. But we know this is true, since the algorithm forces us to choose the quadrant that contains a point larger than the one on the boundary (pts. 1-19).

(d) The problem size is the number rows n. Each recurse step into a quadrant reduces the problem size to $\frac{n}{2}$. The global maximum search of the boundaries is $6n$, which is $O(n)$. Thus, the recurrence relation is:

$$T(n) = T\left(\frac{n}{2}\right) + 6n$$

By Master's Theorem, we determine c and c_{crit}:

$$f(n) = O(n)$$
$$c = 1$$
$$c_{crit} = \log_2 1 = 0$$

Since $c_{crit} < c$, $f(n)$ dominates the running time. Therefore, $T(n)$ is $O(n)$.