Peak Finding In Arrays
1D:

Definition:
Let A be an array of size n.
Let A_{i-1}, A_i, and A_{i+1}, ($i<n$; $(A_{i-1}, A_i, A_{i+1}) \in A$) be any three subsequent items in A.
Let A_0, A_1 be the first and the second items in the array, respectively.
Let A_{n-1}, A_n be the second to last, and the last items in the array, respectively.
An item A_i is a peak iff $A_i \geqslant A_{i-1} \wedge A_i \geqslant A_{i+1}$.
A_0 is a peak if $A_0 \geqslant A_1$.
A_n is a peak if $A_n \geqslant A_{n-1}$.

Corollary 1: The largest item in the array is always the peak.

Proof of Corollary 1:
By the definition of the peak, A_i is a peak iff $A_i \geqslant A_{i-1} \wedge A_i \geqslant A_{i+1}$.
Let A_{max} be the largest item in A; let A_j be any item in A, such that $max \neq j$.
=> $A_{max} \geqslant A_j, \forall j \in (1,2,\ldots,n)$, where n is the size of A.
=> A_{max} is a peak.

Theorem 1: An array has at least one peak.

Very simple proof of Theorem 1: (I'm still leaving the first one below)
By *Corollary 1*, the largest item in the array is always the peak. Every array has at least one item such that $A_{max} \geqslant A_j, \forall j \in (1,2,\ldots,n)$.
=> A_{max} has a peak.

Proof of Theorem1: (*By induction*)
Let P(n) = "an array of size n has at least one peek".

Base case: P(1) => an array of size 1 has one element, which therefore is a peak. => Base case holds.
Inductive step: Assume P(n) is true. => $\exists$ A of size n that has a peak.
Assume P(n+1) by adding an element k into the array A creating a new array A'. There are three possibilities:

- *a)* an element is added to the end of the array A'
- *b)* an element is added to the beginning of the array A'
- *c)* an element is added somewhere in the array A' that is not beginning nor end.

Let's analyze each of the cases.
In the a) case, there are two possibilities. A'_0 was a peak, or A'_0 was not a peak.

1) If A'_0 was a peak and $A'_0 \geqslant k$, then A'_0 is still a peak, since $A'_0 \geqslant A'_1 \wedge A'_0 \geqslant k$; otherwise k will be the new peak.
2) If A'_0 was not a peak, then k will be a peak if $A'_0 \leqslant k$. Otherwise k would not be the peak, but, by P(n), A' would have a peak of A.

In the b) case, there are two possibilities. A'_n was a peak, or A'_n was not a peak.

3) If A'_n was a peak and $A'_n \geqslant k$, then A'_n is still a peak, since $A'_n \geqslant A'_{n-1} \wedge A'_n \geqslant k$; otherwise k will be the new peak.
4) If A'_n was not a peak, then k will be a peak if $A'_n \leqslant k$. Otherwise k would not be the peak,

but, by P(n), A' would have a peak of A.

The c) case is a little bit more complicated. Let's assume that k is inserted between A'_i and A'_{i+1}. Then there are a couple of possibilities, but the rules from the previous two cases apply:

1) If $k \geqslant A'_i \wedge k \geqslant A'_{i+1}$, k will be a peak.
2) If $k \leqslant A'_i$, and A'_i was a peak, it will remain a peak. If A'_i was not a peak, then, by P(n), A' would have a peak of A.

=> P(n+1) holds.

Algorithms' asymptotic complexity:

Greedy approach: Worst case, T(n) = $\Theta(n)$

Recursive approach:

Worst case, T(n) = T(n/2) + $\Theta(1)$.

Assuming the peak will be found on the last try, this yields: T(n) = $\sum_{1}^{\log_2(n)} \Theta(1) = \Theta(\log_2(n))$.