The trick is to find the right recursive formulation of the problem. The idea is similar to the line segment intersection problem we discussed in the tutorial. We devise a procedure \texttt{MaxStabbing}(P,S,I) which takes as argument an interval $I \subseteq \mathbb{R}$, a set of points $P \subset I$, and a set of intervals $S$ such that every interval in $S$ has at least one endpoint in $I$. The result of this procedure is that every point $p$ in $P$ is annotated with the maximum weight of all intervals in $S$ that contain $p$ or $-\infty$ if no such interval exists. Clearly, the invocation \texttt{MaxStabbing}(P,S,\mathbb{R}) solves the problem we are asked to solve because every point in $P$ belongs to $\mathbb{R}$ and every interval in $S$ has both its endpoints in $\mathbb{R}$.

Now, if $|P| = 1$, we obtain a simple linear-time solution to the problem: Scan all intervals in $S$. If none of them contains the one point $p \in P$, then annotate $p$ with a maximum weight of $-\infty$. Otherwise, annotate $p$ with the maximum weight of all the intervals in $S$ that do contain $p$.

If $|P| > 1$, we split $P$ into two subsets $P_l$ an $P_r$ containing the $\lceil n/2 \rceil$ smallest and $\lfloor n/2 \rfloor$ largest elements in $P$, respectively. Let $x$ be a value such that all elements in $P_l$ are less than $x$ and all elements in $P_r$ are greater than $x$ (e.g., the average of the largest element in $P_l$ and the smallest element in $P_r$), and let $I_l := I \cap (-\infty,x]$ and $I_r := I \cap [x,\infty)$. We construct two sets $S_l$ and $S_r$ containing all intervals in $S$ with at least one endpoint in $I_l$ and $I_r$, respectively. Note that every interval in $S$ belongs to at least one of these two sets but may contain both. Now consider a point $p \in P_l$. The maximum-weight interval in $S$ that contains $p$ is the one with larger weight among the maximum-weight interval in $S_l$ that contains $p$ and the maximum-weight interval in $S_r \setminus S_l$ that contains $p$. We find the former for all $p \in P_l$ by making the recursive call \texttt{MaxStabbing}($P_l,S_l,I_l$). For the latter, we observe that, since any such interval has one endpoint in $S_r$, no endpoint in $S_l$, and contains a point in $P_l$, it must completely span $I_l$. In particular, it contains all points in $P_l$, so the maximum-weight interval in $S_r \setminus S_l$ containing $p$ is the same for all points $p \in P_l$. We find this interval by scanning $S_r$ and remembering the maximum interval in $S_r$ we have seen so far and which spans $I_l$. This clearly takes linear time.
The analysis for the points in $P_r$ is symmetric to the one just given. Thus, we obtain the following algorithm. In this algorithm, each point in $P$ is represented by as a record with fields $x$ and $w$. $x$ is the position of the point and $w$ is the weight we are to compute. Each interval in $S$ is represented as a record with fields $l$, $r$, and $w$, which are the left endpoint, right endpoint, and weight of the interval, respectively.

**MaxStabbing($P$,$S$)**
1. **MergeSort($P$)**  // Sort the points by their positions.
2. **RecMaxStabbing($P$,$S$,$\emptyset$)**

**RecMaxStabbing($P$,$S$,$I$)**
1. if $|P| = 1$
2. $P[0]$.w = $-\infty$
3. for $i = 0$ to $|S| - 1$
4. if $S[i]$.w > $P[0]$.w
5. $P[0]$.w = $S[i]$.w
6. else $m = |P|/2$
7. $x = (P[m-1]$.x $+ P[m]$.x$)/2$
8. $I_l = I \cap (-\infty,x]$
9. $I_r = I \cap [x,\infty)$
10. $S_l = \emptyset$
11. $S_r = \emptyset$
12. for $i = 0$ to $|S| - 1$
13. if $S[i]$.l $\in I_l$ or $S[i]$.r $\in I_l$
14. Append $S[i]$ to $S_l$
15. if $S[i]$.l $\in I_r$ or $S[i]$.r $\in I_r$
16. Append $S[i]$ to $S_r$
17. **RecMaxStabbing($P[0..m-1]$,$S_l$,$I_l$)**
18. **RecMaxStabbing($P[m..|P|-1]$,$S_r$,$I_r$)**
19. $w_l = -\infty$
20. $w_r = -\infty$
21. for $i = 0$ to $|S| - 1$
22. if $I_l \subseteq [S[i]$.l,$S[i]$.r$]$ and $S[i]$.w > $w_l$
23. $w_l = S[i]$.w
24. if $I_r \subseteq [S[i]$.l,$S[i]$.r$]$ and $S[i]$.w > $w_r$
25. $w_r = S[i]$.w
26. for $i = 0$ to $m - 1$
27. if $w_l > P[i]$.w
28. $P[i]$.w = $w_l$
29. for $i = m$ to $|P| - 1$
30. if $w_r > P[i]$.w
31. $P[i]$.w = $w_r$

The running time of **MaxStabbing($P$,$S$)** is the running time of **MergeSort($P$)** plus the
running time of $\text{RecMaxStabbing}(P, S, \mathbb{R})$. The former is in $\Theta(n \lg n)$. We prove that the same is true for the latter, so the total running time is in $\Theta(n \lg n)$ as required.

First observe that, excluding the recursive calls it makes, the running time of an invocation $\text{RecMaxStabbing}(P, S, I)$ is in $\Theta(|P| + |S|)$: For the case when $|P| = 1$, we execute lines 2–5, which take $\Theta(1 + |S|) = \Theta(|P| + |S|)$ time. For the case when $|P| > 1$, lines 7–11 and 19–20 take constant time, lines 12–16 and 21–25 take $\Theta(|S|)$ time, and lines 26–31 take $\Theta(|P|)$ time.

Next observe that there are $\Theta(\lg |P|) = \Theta(\lg n)$ levels of recursion because, for each invocation, $P[0..m-1]$ has size $\lceil |P|/2 \rceil$ and $P[m..|P|-1]$ has size $\lfloor |P|/2 \rfloor$.

By these two observations, it suffices to show that the total input size of all invocations at the same level of recursion is in $\Theta(n)$. Consider an invocation $\text{RecMaxStabbing}(P, S, I)$. This invocation has an input point $p$ in $P$ if and only if $p \in I$ and an input interval $I_j$ in $S$ if and only if $I_j$ has an endpoint in $I$. Since the intervals associated with invocations at the same level of recursion are disjoint, every input point contributes to the input size of exactly one invocation at each level of recursion and every input interval contributes to the input size of at most two invocations at each level of recursion. This shows that the total input size of all invocations at the same level of recursion is at most $n + 2n \in \Theta(n)$. 
