Case $t = A[m]$ is obvious as the function works

~~Case $t < A[m]$~~
~~We know, since the array is sorted, that t must lie if it exists between $A[left]$ and $A[m-1]$. So if the recursive call works correctly this call will as well. Then for this is $n = m-1-left$ or $n = \lfloor \frac{left+right}{2} \rfloor - 1 - left < right - left$~~

Case t does not exist, since the array is sorted when $A[m]$ should be t it will not find an $A[m] = t$ but instead $A[m] < t$ this will cause the function to be called again where $left > right$ thus returning -1

Case $t < A[m]$, We know, as the array is sorted, that t, if it exists, must lie between $A[left]$ and $A[m-1]$. If the recursive call works correctly, so will this call, so $n = m-1-left$ or $n = \lfloor \frac{left+right}{2} \rfloor - 1 - left$. So if $left+right$ is even then $n = \frac{left+right}{2} - 1 - left = \frac{left+right-2left}{2} - 1 = \frac{right-left}{2} - 1$ which is smaller than $right-left+1$. If $right+left$ is odd then $n = \frac{left+right-1}{2} - 1 - left = \frac{right-left-1}{2} - 1$ which is also smaller than $right-left = k > 0$. So the recursive call must be between 1 and n and must be correct by Hyp.

Case $t > A[m]$ this is symmetrical to case $t < A[m]$.

$\therefore$ as all cases the ind step works we can conclude expSearch works $\square$

TCA: $T(n) = T(\frac{n}{2}) + O(1)$; $a=1, b=2, f(n) = O(1)$; $n^{\lg 1} = n^0 = 1$

as $O(1) \in \Theta(1)$ where $k=0$ then by case 2 of masters theorem

$T(n) = \Theta(1 \lg^a n)$ or $T(n) = \Theta(\lg n)$ $\square$