R-4.5

Let the function be $\mathsf{search}(x, A, n)$. If n is equal to 1 check whether $A[0] = x$, and return true if so. Otherwise, let $m = \lfloor n/2 \rfloor$. Split A into two subarrays $A_1 = A[0..m-1]$ and $A_2 = A[m..n-1]$. Recursively search the first subarray for x by calling $\mathsf{search}(x, A_1, m)$. If it is not found there then recursively search the second subarray by calling $\mathsf{search}(x, A_2, n-m)$.

In the worst case, x is not in the array, and this algorithm effectively searches every element of A before realizing this. The resulting in a running time of $O(n)$. Since the size of the subarray in each recursive call is roughly half of the original, the total space needed along any single recursion path is at most $n + (n/2) + (n/4) + \ldots + 1 \leq 2n$. Thus, the space requirements are also $O(n)$.