Algorithm 1:

The algorithm runs in O(n log n) time because each recursive splits the list of buildings in half. It produces the recurrence of $T(n) = 2T(n/2) + O(n)$. The merge requires canning the visible buildings from each half, which causes the worst case to be O(n). By applying the Master Theorem, it gives an overall running time of O(n log n). The recursion depth is log2(n), so the algorithm uses O(log n) by using extra space on the call stack.

Algorithm 2:

The greedy activity-selection algorithm runs in O(n log n) time because sorting the activities by their finish time is expensive, taking n log n time. After the sorting, the algorithm performs a single linear scan through the list, in which it selects each activity whose start time is compatible, which is O(n). The total running time is dominated by the sort where n + n log n is equal to O(n log n). The algorithm uses O(1) extra space because it only stores a few variables and it appends compatible activities directly to the output list.