My answers

These counterexamples are a bit simpler than mine, so yea, you can use them as well

5-7-4-1 **(a)** Consider the sequence of weights 2, 3, 2. The greedy algorithm will pick the middle node, while the maximum weight independent set consists of the first and third.

1-9-3-2-4 **(b)** Consider the sequence of weights 3, 1, 2, 3. The given algorithm will pick the first and third nodes, while the maximum weight independent set consists of the first and fourth.

(c) Let S_i denote an independent set on $\{v_1, \ldots, v_i\}$, and let X_i denote its weight. Define $X_0 = 0$ and note that $X_1 = w_1$. Now, for $i > 1$, either v_i belongs to S_i or it doesn't. In the first case, we know that v_{i-1} cannot belong to S_i, and so $X_i = w_i + X_{i-2}$. In the second case, $X_i = X_{i-1}$. Thus we have the recurrence

$$X_i = \max(X_{i-1}, w_i + X_{i-2}).$$

We thus can compute the values of X_i, in increasing order from $i = 1$ to n. X_n is the value we want, and we can compute S_n by tracing back through the computations of the *max* operator. Since we spend constant time per iteration, over n iterations, the total running time is $O(n)$.

The question is simple, there is a graph in a straight line v1-v2-v3-...-vn. Finding an independent set S_i = finding set of vertices with no common edge, so if we pick say, v3 we can't pick v2 or v4 as they have an edge. Moreover, each vertex v_i has weight w_i. Just add all weights to get the weight of the independent set S_i.

(C) Case 1: We take v_i in S_i : Then v_i-1 cannot be in S_i, as they have an edge ofc! so, X_i = w_i + X_i-2

Case 2: We DO NOT take v_i. Then simply, X_i = X_i-1

[1] ex301.516.319