```
P1 A. [] → [C]
          | → [D] → [D,G] → [D,G,H]
          |         | → [D,H]
          | → [E] → [E,F] → [E,F,H]
          |         | → [E,G] → [E,G,H]
          |         | → [E,H]
          | → [F] → [F,G] → [F,G,H]
          |         | →[F,H]
          | → [G] → [G,H]
          | → [H]
```

```
B. [] → [C]
     | → [D] → [D,G] → [D,G,H]
```

```
C.        [] → [C]
           | → [D] → [D,G]
           |        | → [D,H]
           | → [E] → [E,F]
           | → [F]
           | → [G]
           | → [H]
```

P2 A. Yes, it is a tree, as there's only one way to get to any vertex from the source.
B. This tree has at most depth of N, as we make decisions for each of N objects.
C. The branching factor is N, as N objects to choose from.
D. Not known in advance, because it depends on the constraint cost.

P3 A. State:

A set of k different vertices, where k is in [0, K], in alphabetic order that are not directly connected to any other vertices in the set

Successor:

Adding a new vertex to the current set and the added vertex is not connected to any vertex already in the set

Start State:

An empty set with no vertices

Goal State:

A set with K vertices and with no two vertices in the set directly connected by an edge.

B. Depth of Goal States:

The depth of the goal states is known in advance to be K, as specified in the question to find K vertices without any edge connecting each other

Search Algorithm:

Due to the certainty in the depth of goal states, DFS would be suitable since we don't need to iteratively test the depth of the goal states for the result.