1a Number of times you can halve N, so $O(\log N)$

1b The precise answer is $\binom{n}{2}$, I think. So $O(n^2)$.

2a The answer is Subtree.

2b This is quite cute. It's clear that we need to recurse on the left and right children of the root. What makes this exercise not completely trivial is that you need to send two different items of information "back up": the depth of the largest perfect subtree (rooted anywhere) as well as the depths of the largest perfect subtree *rooted here*. So D denote the depth of the largest perfect subtree and let H denote depth of the largest perfect subtree rooted at the root. Then there's some simple logic about how these values are combined. I don't trust myself, so I wrote it up in python and it seems to work:

```
def f(N):
    if N == None: return (0,0)
    l,r = f(N.left), f(N.right)
    H = 1 + min(l[1], r[1])
    D = max(l[0], r[0], H)
    return (D, H)
```

2c $O(n)$, because the conquer step takes constant time. [1]

[1] Perhaps surprisingly. Don't assume all divide-and-conquer questions are $O(n \log n)$.

3a Clearance (B)

3b *Here's the best solution I can think of*: "Find an MST T in the input graph (using Prim or Kruskal, say) and return the largest value on the unique path in T between s and t (using BFS, say). Running time $O(m \log m)$ because of Prim, say." *There are other ways of doing it, such as adding edges to an empy graph in ascending order of their security level and running your favourite connectivity algorithm on each. That would take $O(m \log m)$ or $O(nm)$ or $O(m^2)$ time, depending on the details.*

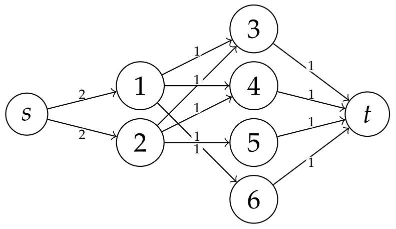
4 Prototypical Dynamic Programming question because the underlying graph is a DAG.[2] The multiple-choice answers are C, C, B, A.

[2] Otherwise the problem would be NP-hard, reduction from Hamiltonian cycle

5a The flow problem is V.

5b Construct a flow network H with vertex set $V = \{s, t\} \cup L \cup R$ and arc set $A = \{\, su \colon u \in L \,\} \cup \{\, vt \colon v \in R \,\} \cup E$, where E is the edge set of the input graph G. The capacities on the arcs outgoing from s are 2, all other capacities are 1.[3] For the example input,

[3] The capacities between L and R aren't really important.



If there is a flow of size $2n$ in H then there is a V-matching in G; the matching consists of the arcs with flow 1 in E.

6a The NP-hard problem is Crossing

6b Hamiltonian path

6c Hamiltonian path $\leq_P$ Crossing.

6d Let H denote an instance to Hamiltonian path. We construct an instance k, G to Crossing as follows.[4] Set $G = H$. Set $k = 0$. This finishes the construction. Observe that a tree $k = 0$ crossings is a path, so a spanning tree with 0 crossings is a "spanning path". Thus, if T a solution to Crossing k, G then T is also a solution to Hamiltonian path for H. Conversely, every Hamiltonian path in H is a spanning tree with 0 crossings in G.[5]

[4] These two sentences alone are very, very important. Start with them, otherwise you will mess up.

[5] The last sentence is necessary to show that we don't produce "false negatives", but this particular problem is so easy that it seems like empty formalism.