(i) DPV Exercise 8.2.

Suppose, given a graph G, the procedure $D(G)$ returns *true*, if G has a Rudrata path, and *false*, otherwise. Here is how to use D to find a path if it exists.

```
function RudrataPath(G)   // where (V,E) = G
  if not D(G) then return "no path"
  E' ← E
  for each e ∈ E do
    // See if we still have a Rudrata path
    // when we leave out e
    G' ← (V, E' − {e})
    if D(G') then E' ← E' − {e}
  return E'
```

The only edges left in E' at the end are the edges making up the Rudrata path—all the other edges could be left out.

(ii) DPV Exercise 8.3.

Reduction from SAT: Given an instance of SAT I, let (I,k) be an instance of stingy SAT where k = the number of variables in SAT instance I. We have to show that: I is a yes-instance of SAT if and only if (I,k) is a yes-instance of stingy SAT.

($\Rightarrow$) Suppose I has a satisfying assignment S. Then no more than k variables in S can be true, because there are a total of k variables. So S works as a positive solution for (I,k) too.

($\Leftarrow$) Suppose (I,k) has a satisfying assignment S with no more than k variables assigned *true*. Then obviously S is a positive solution to I also.

(iii) DPV Exercise 8.4.

(a) An instance of clique-3 consists of a graph G and an integer k. A possible solution consists of a set S of k-many vertices. The checking algorithm $C(G,k,S)$ checks that there really are k vertices in S and each of them have edges going to the other $k-1$ many vertices in S. All of this clearly can be done in $O(|G|+k)$ time.

(b) The reduction goes the wrong way. To work it has to be from a known NP-complete problem (e.g., clique) to the problem we want to show NP-complete (e.g., clique-3).

(c) Consider ①—②—③. The set $\{2\}$ is a vertex cover of size 1, but $\{1,3\}$ sure isn't a clique. So the business about the complement of a vertex cover giving you a clique is bogus.

(d) Given (G,k): If $k>3$ return *"no clique, the vertices can only be adjacent to 3 other vertices!"* If $k=1$, pick a vertex and return it. If $k=2,3$, search all k elements subsets of V to find a k-clique. If you find one, return it. If you don't, return *"no k-clique"*. Since there are $n\cdot(n-1)/2$ subsets of V of size 2 and $n\cdot(n-1)\cdot(n-2)/6$ subsets of V of size 3, the $k=2,3$ case are clearly $O(|G|^3)$ time.

(iv) DPV Exercise 8.10.

(a) Reduction from *Clique*: Given a graph G and $g>0$, construct K_g = the complete graph on g vertices. Then (K_g,G) is an instance of subgraph isomorphism that has a positive answer iff G has a clique subgraph on g vertices.

(c) Reduction from *SAT*: Given ϕ a conjunctive normal form formula, let g = the number of clauses in ϕ. Then (ϕ,g) is an instance of *MAX SAT* that has a positive solution iff ϕ is satisfiable; and in fact, the same satisfying assignment works for both cases.

(d) Reduction from *Clique:* Given a graph G and $g>0$, let $a=g$ and $b=g\cdot(g-1)/2$. Then (K_b,a,b) is an instance of *Dense subgraph* that has a positive answer iff G has a clique subgraph on g vertices. (A clique on g vertices always has $g\cdot(g-1)/2$ edges.)

(v) DPV Exercise 8.18.

As the hint suggested, we consider the *Bounded Factor Problem:*

Given: positive integers n and b.

Question: Is there an $m\in\{2,\dots,b\}$ such that m evenly divides n?

Here is a checking algorithm for the *Bounded Factor Problem.*

```
function C((n,b),m)
  if 2 ≤ m ≤ b and (n mod m) = 0
    then return true
    else return false
```

C runs in $O(|b|+|n|^2)$ by the result of Chapter 1. So the *Bounded Factor Problem* is in NP. So suppose P=NP. Then we can solve the Bounded Factor Problem in poly-time, and by using the binary search trick, we can then factor in poly-time. Hence, we can break RSA.