1. A d-regular graph cannot contain a clique of size $d+2$, since the latter is a $(d+1)$-regular graph. Here's a polynomial algorithm that solves the problem:

   Given an instance (G,k), we answer $false$ if $k > d+1$ or $n = |V| < k$. Otherwise, we check the $\binom{n}{k} \leq n^k \leq n^{d+1}$ sets of size k in G and answer $true$ if any of them is a clique.

   Time Complexity: $\leq \binom{k}{2} n^{d+1} = O(n^{d+1})$.

   The result follows for $d = 2005$.

   Note: Can you find a $\binom{d}{k}\binom{k}{2}n = O(n)$ algorithm?

2. Any single clause c in a DNF formula is a 1-CNF formula, so we can polynomially check its satisfiability (i.e., verify that if x appears in c, $\neg x$ doesn't appear and vice versa).

   Now, the whole DNF formula ϕ is satisfiable if and only if *any* of its clauses is satisfiable, so we can consider the clauses one by one and return $\bigvee_{c\in\phi}$ 1-CNF-SAT(c).

3. SAT $\in NP \subset EXP$, so there exists a deterministic TM M accepting SAT. Consider the deterministic TM M' which simulates M and halts if and only if M accepts.

   Let $\langle M' \rangle$ be a description of M', and consider the reduction $f(\phi) = (\langle M' \rangle, \phi)$ from SAT to HALT.

   Correctness: $\phi \in$ SAT $\Leftrightarrow$ M accepts $\phi \Leftrightarrow$ M' halts on ϕ.

   Space Complexity: Logarithmic, as $\langle M' \rangle$ is constant and ϕ should be simply copied as-is.

4. (a) Assume we have an oracle for CLIQUE(G,k). Let $\omega = \omega(G)$ be the size of the maximal clique in G; note that $1 \leq \omega \leq n$.

Algorithm 1 FIND-MAX-CLIQUE(G)

```
ω ← ω(G) {use any search algorithm (naive, linear, binary)}
K ← V
for v ∈ K do
    if CLIQUE(G[K − {v}], ω) = true then
        K ← K − {v}
return K
```

Correctness:
The algorithm erases all vertices not crucial so the survival of the largest clique, so at the end we have $|K| = \omega(K) = \omega(G)$ and K is the requested clique.

Time Analysis:
The oracle is queried $\leq 2n$ times.

(b) The interesting case is of course $G_1 \sim G_2$. Denote G_1's vertices by $\{u_1, u_2, \ldots, u_n\}$ and G_2's vertices by $\{v_1, v_2, \ldots, v_n\}$, with the isomorphism being $v_i \sim u_{\pi(i)}$.
We need a set of n distinguishable gadgets $\{H_i\}_{i=1}^n$ that haven't appeared in G_1, G_2, for instance $H_i = K_{n+i}$.

Correctness:
In step i we recover $\pi(i)$ assuming we know $\pi(j)$ for $j < i$ by checking all possibilities. Any isomorphism $G_1 \overset{\tau}{\sim} G_2$ must have $\tau(j) = \pi(j)$ for $j < i$ due to the gadget H_j. By the end of step n, we know π entirely.

Algorithm 2 FIND-ISOMORPHISM(G_1, G_2)

for $i \leftarrow 1$ to n **do**
  connect H_i to u_i in G_1
  for $k \leftarrow 1$ to n **do**
    $G_2' \leftarrow G_2$
    connect H_i to v_k in G_2'
    if ISOMORPHIC(G_1, G_2') **then**
      $\pi(i) \leftarrow k$
      $G_2 \leftarrow G_2'$
      break from the k for-loop
  if we got here not by breaking from the loop (can only happen for $k = 1$), return $false$
return π

Time Analysis:
The oracle is queried $\leq n^2$ times.

Note that by erasing vertices (instead of marking them) we might get the wrong permutation (e.g., consider a path of length 3).

5. (a) Let L be an arbitrary NP language, accepted by a DTM $M(x, y)$.
   Recall the reduction $L \leq_P$ 3-SAT presented in Cook's theorem:
   Given x we construct $f(x) = \phi_{M,x}$ in variables $\{y_i\}_{i=1}^n$ that is satisfiable if and only if $M(x, y)$ accepts. Since the same y is a witness for both $x \in L$ and $f(x) \in$ 3-SAT, we conclude that 3-SAT is NP-hard under Levin reductions as $(f, identity)$ are the requested functions.

   (b) Consider the polynomial self-reduction for SAT presented in class. We needed a SAT oracle, but as A is assumed to be NP-hard, we can polynomially reduce every SAT query to an A query. Thus, there's a polynomial oracle TM M^A that given a formula ϕ, computes a satisfying assignment $\rho(\phi)$ for it, if exists.
   Consider the Levin reduction (f, g) from A to 3-SAT (shown to exist in 5a since $A \in NP$). Given $x \notin L$, we have $f(x) \notin$ 3-SAT so $M^A(f(x))$ rejects; Given $x \in L$, we have $f(x) \in$ 3-SAT so $M^A(f(x))$ computes the witness $w = \rho(f(x))$. Using g, we can compute $y = g(w)$, a witness for $x \in L$.
   Obviously, all the reductions are done in polynomial time.