1. (a) True. A 3DNF formula is satisfiable iff some of its clauses does not contain two conflicting literals (i.e. x_k and $\bar{x}_k$ for some k). This can be easily checked in polynomial time.

(b) False. A counterexample:

$A = \{0^n 1^n \mid n \geq 0\}$ is non-regular. $(\Sigma = \{0,1\})$

$B = \{0\}$ is regular.

Let $f: \Sigma^* \to \Sigma^*$ be $f(x) = \begin{cases} 0, & \text{if } x \in A \\ 1, & \text{o.w.} \end{cases}$

f is computable because there exists a decider M for A. Also, $x \in A$ iff $f(x) \in B$. So $A \leq_m B$.

(c) True. In fact, $\forall L_1, L_2 \in P$, $L_1, L_2 \neq \phi, \Sigma^*$, we have $L_1 \leq_p L_2$. To prove this, pick $y \in L_2$ and $z \notin L_2$.

arbitray

Then let $f: \Sigma^* \to \Sigma^*$ be $f(x) = \begin{cases} y, & \text{if } x \in L_1 \\ z, & \text{o.w.} \end{cases}$

Then $x \in L_1$ iff $f(x) \in L_2$. Also, f is polynomial-time computable because we can compute it using the polynomial-time decider M_1 for L_1.

So the statement holds for arbitrary $L_1, L_2, \cdots, L_n \in P$ satisfying $L_i \neq \phi, \Sigma^*$ and $L_i \neq L_j$ $\forall$ $i \neq j$.

2. Recall that $ALL_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$ is undecidable. Since CFG and PDA are interconvertible, this implies $ALL_{PDA} = \{\langle P \rangle \mid P \text{ is a PDA and } L(P) = \Sigma^*\}$ is also undecidable.

Now we prove L is undecidable by showing $ALL_{PDA} \leq_m L$.

Let D_0 be a DFA that accepts nothing, i.e. $L(D_0) = \phi$.

Then for any PDA P, let $f(\langle P \rangle) = \langle P, D_0 \rangle$.

Then $L(P) = L(P) \cup L(D_0)$

$\Rightarrow$ $L(P) = \Sigma^*$ iff $\overline{L(P) \cup L(D_0)} = \phi$

$\Rightarrow$ $\langle P \rangle \in ALL_{PDA}$ iff $\langle P, D_0 \rangle \in L$.

Obviously, f is a computable function. Thus, $ALL_{PDA} \leq_m L$.

3. We construct a TM that decides PrefixFreeREX as follows.

On input R, reject if R is not a valid regular expression. Otherwise, construct a DFA D for the language $L(R)$. (Recall that we can construct an equivalent NFA for $L(R)$ from R, then we can convert an NFA to a DFA.)

By running a depth-first search (DFS) starting from q_0, we can remove all states that are not reachable from q_0 in this DFA.

Next, for each accept state q, we run a DFS starting from q and check if another accept state $q' \neq q$ is reachable from q, or if there is a loop from q to itself. If any such paths or loops are found, then reject. Otherwise, accept.

Note that it is first required to remove all the (accept) states not reachable from q_0 as these states cannot lead to any string being in the language.

4. (a). This problem is in P. So it cannot be NP-Complete unless $P = NP$.

An algorithm for this problem is as follows. We enumerate all the subsets of the vertices of size $n-3$, and check whether any of these subset forms a vertex cover for G.

There are $\binom{n}{n-3} = \binom{n}{3} = O(n^3)$ subsets to enumerate.

To check whether a subset S is a vertex cover, we only need to check whether there exists an edge between two vertices not in S. There are only three edges to check. So this algorithm runs in polynomial time.

(b) Recall that the following two problems are equivalent:

① Decide whether G has a clique of size l;

② Decide whether the complement of G has a vertex cover of size $n-l$.

So, here we study the problem $(\frac{n}{3})$-Clique instead of $(\frac{2n}{3})$ - Vertex - Cover. (Let V.C. stand for Vertex Cover)

Recall Karp's reduction from SAT to CLIQUE.

Reading it carefully, we notice that for a 3 CNF formula ϕ, by the reduction, it is mapped to a graph G with $3m$ vertices (where m is the number of clauses in ϕ) such that ϕ is satisfiable iff G has a clique of size m. This implies that $3SAT \leq_P (\frac{n}{3})$-Clique!

So the problem $(\frac{n}{3})$-Clique is NP-Complete. (Its membership in NP is obvious. Why?)

Hence, the problem $(\frac{2n}{3})$-V.C. is also NP-Complete.

(c) We use the fact that $(\frac{2n}{3})$-V.C. is NP-Complete.

Now we show that $(\frac{2n}{3})$-V.C. $\leq_P$ $(\frac{n}{2})$-V.C.

Given a graph G with n vertices, let us modify it by simply adding ℓ isolated vertices, where ℓ is a parameter chosen later. Let G' be this modified graph with $n+\ell$ vertices. Obviously, G has a V.C.

$n' =$

of size k iff G' has a V.C. of size K.

Now $k = \frac{2n}{3}$, and we want K to be $\frac{n+l}{2} = \frac{n'}{2}$. So

we need to set $l = \frac{n}{3}$. Then, we have

$\langle G \rangle \in (\frac{2n}{3})$-V.C. iff $\langle G' \rangle \in (\frac{n}{2})$-V.C.

Also, the above reduction is obviously polynomial-time

computable. So $(\frac{2n}{3})$-V.C. $\leq_p$ $(\frac{n}{2})$-V.C.

Thus, $(\frac{n}{2})$-V.C. is also NP-Complete. (It is clearly

in NP. Why?)

5. Let $L \in P$ and let M be a polynomial-time decider for L. We will build a decider M' for L^* as follows.

For a string $w = w_1 w_2 \cdots w_n \in \Sigma^n$, let $w_{i,j} \triangleq w_i w_{i+1} \cdots w_j$ for any $i \leq j$. The decider builds a table where $table(i,j) = true$ if $w_{i,j} \in L^*$. We do this by considering all substrings of w starting with those of length 1 and ending with those of length n.

" On input $w = w_1 w_2 \cdots w_n$:

1. If $w = \epsilon$, then <u>Accept</u>; else
2. For $l := 1$ to n
3. For $i := 1$ to $n-(l-1)$
4. $j := i + l - 1$
5. Run M on $w_{i,j}$
6. If M accepts $w_{i,j}$, then $table(i,j) = True$
7. Else
8. For $k := i$ to $j-1$
9. If $table(i,k) = true$ and $table(k+1,j) = true$
10. then $table(i,j) = true$

11. If table$(1,n)$ = true, then Accept; else Reject.

Assume M has time complexity $O(n^k)$ for some $k \geq 0$. Then M' has time complexity $O(n^{k+3})$ which is still polynomial in n.