Problem 1: The Class P (35 points)

A. (25) Show that the class P is closed under union, intersection, concatenation and complement.

Assume $L_1 \; and \; L_2 \in P$ and $M_1 \; and \; M_2$ are their TMs. For $s \in \Sigma^*$

We create a DTM M' that accepts input s and

- Run $M_1(s)$ and $M_2(s)$
  - If M_1 and M_2 are both $O(n)$, so polynomial time, then M' is $O(n_1 + n_2)$, which is also polynomial time.
- Accepts when either M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class P are closed under union since they all decide in polynomial time.

We create a DTM M' that accepts input s and

- Run $M_1(s)$ and $M_2(s)$
  - If M_1 and M_2 are both $O(n)$, so polynomial time, then M' is $O(n_1 + n_2)$, which is also polynomial time.
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class P are closed under intersection since they all decide in polynomial time.

We create a DTM M' that accepts input s and

- For each two substrings of s where $s = s_1 s_2$
- Run $M_1(s_1)$ and $M_2(s_2)$
  - If M_1 and M_2 are both $O(n)$, so polynomial time, then M' is $O(2n)$, which is also polynomial time.
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class P are closed under concatenation since they all decide in polynomial time.

We create a DTM M' that accepts input s and

- Assume s' is the reverse of s, or its complement.

- Run $M_1(s)$ and $M_2(s')$
  - If M_1 and M_2 are both $O(n)$, so polynomial time, then M' is $O(2n)$, which is still polynomial time.
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class P are closed under complement since they all decide in polynomial time.

B. (10) Show that EQ_{DFA} is in the class P.

We create TM M that accepts input s and TM M

- Create M_1 that on any input, accepts.
- Create M_2 that on any input, run M on s, if it accepts, accept.
- Run $M_1(s)$ and $M_2(s)$
  - M_1 is constant since it accepts on any input and M_2 is equivalent to M
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then EQ_{DFA} is in class P since it decides in polynomial time

C. (10) ALL_{DFA} = { <*D*> | *D* is a DFA with L(*D*) = Σ* }. Show that ALL_{DFA} is in the class P.

Mimic ALL_{DFA} using TM M_{DFA} where

- We receive input s
- We run the DFA using breadth first search.
  - If any accept states are found accept.
  - If any non-accept states are found or we've searched the entire tree, reject.

Moving through a graph's nodes via breadth first search is in polynomial time. Then ALL_{DFA} is in class P since it decides in polynomial time.

Problem 2: The Class NP (25 points)

A. (10) Show that the class NP is closed under union and concatenation.

Assume $L_1 \; and \; L_2 \in NP$ and $M_1 \; and \; M_2$ are their TMs. For $s \in \Sigma^*$

We create a NDTM M' that accepts input s and

- For input s
- Run $M_1(s)$ and $M_2(s)$
    - If M_1 and M_2 are both nondeterministic, they'll randomly choose the next character and the total time they'll take to run through the whole string is the number of times equal to string s.
    - If M_1 and M_2 are both $O(n^k)$, then M' is $O(n^{2k})$, which is still polynomial time.
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class NP are closed under union since they all decide in polynomial time.

Assume $L_1 \; and \; L_2 \in NP$ and $M_1 \; and \; M_2$ are their TMs. For $s \in \Sigma^*$

We create a NDTM M' that accepts input s and

- For each two substrings of s where $s = s_1 s_2$ and $k = |s|$
- Run $M_1(s_1)$ and $M_2(s_2)$
    - If M_1 and M_2 are both nondeterministic, they'll randomly choose the next character and the total time they'll take to run through their respective substrings is the number of times equal to half the length of the original string s.
    - If M_1 and M_2 are both $O(n^{k/2})$, then M' is $O(n^k)$, which is still polynomial time.
- Accepts when both M_1 and M_2 accept.
- Rejects otherwise.

Then all languages of class NP are closed under concatenation since they all decide in polynomial time.

B. (15) We say that graphs *G* and *H* are **isomorphic** if the nodes of *G* may be reordered so that it is identical to *H*. Let *ISO* = { <*G*, *H*> | *G* and *H* are isomorphic graphs }. Show that *ISO* is in the class NP.

Problem 3: NP-Completeness (25 points)

A subset *S* of the nodes of a graph *G* is a **dominating set** if every other node of *G* is adjacent to some node in *S*. Consider the language:

DOMINATING-SET = { <*G*, *k*> | *G* has a dominating set with *k* nodes }

Show that *DOMINATING-SET* is NP-complete by reduction from *VERTEX-COVER*. You can find *VERTEX-COVER*, and the proof that *VERTEX-COVER* is NP-complete, in Chapter 7 of your textbook.

Problem 4: NP-not-so-Completeness (30 points)

You can find the *SUBSET-SUM* problem in our slides, and the proof of its NP-completeness in chapter 7. Let *UNARY-SSUM* be *SUBSET-SUM* except with all numbers represented in unary.

A. (15) Why is *UNARY-SSUM* **not** NP-complete when *SUBSET-SUM* is?
B. (15) Show that *UNARY-SSUM* is in the class P.

$V =$ on input $< X, t >$:

- Reject if C is not a subset of X.
- Accept if T_b is true
  - Assume $t = \{t_0, (t_0 + t_1), (t_0 + t_1 + t_2) \ldots t_t\}$
    - Since our numbers are unary
  - Create array Boolean array T of length t and set T_0 true
  - For each value in X, where $t_j + X_i \leq t$ set $T_{t_j + X_i}$ to true if T_{t_j} is true.
  - Since our numbers are in unary we are iterating over all the elements of our array T for each element of X.
- Reject otherwise

This algorithm ends up resulting in $O(X * t)$ or the length of our set multiplied by our target number (so the length of our Boolean array)

Problem 5: A Contrasting Path (35 points)

A simple path in a graph is a path that contains no repeated vertices, and (therefore) no cycles. Let *G* be an undirected graph and consider the languages:

SPATH = { <*G*, *a*, *b*, *k*> | *G* has a simple path of length ≤ *k* from *a* to *b* }

LPATH = { <*G*, *a*, *b*, *k*> | *G* has a simple path of length ≥ *k* from *a* to *b* }

A. (10) Show that *SPATH* is in class P.

$V =$ on input $< G, a, b, k >$:

- Reject if a or b is not a node of G or if $k < 1$
- Assume an array T of length k used to track visited nodes.
  - Traverse every node in the tree starting at node a.

        - Push current node i into the array.
        - If node i is equal to node b then **accept**.
        - If the node is already in the array or the array is full, go back a node and try a different path
    - Otherwise, **reject**

This algorithm ends up resulting in $O(k * n)$ or the number of nodes in our graph multiplied by our max path length

B. (25) Show that *LPATH* is NP-complete.

$LPATH \in NP$ by the following

$M =$ on input $< G, a, b, k >$
Non-deterministically push an unchosen node into an array T of length k,

- if the node has already been traversed or T is full, go back and pick a different unchosen node
- if T forms a path to b, accept.
- Otherwise, if we've traversed all possible nodes, reject

$V =$ on input $< G, a, b, k >$:
Push each node into array T of length k in the graph starting after node a

- if the node has already been traversed or T is full, go back and pick a different unchosen node.
- if T forms a path to b, accept.
- Otherwise, if we've traversed all possible nodes, reject

So $HAMPATH \leq_P LPATH$ because

$M_H =$ On input $< G, a, b >$

- If k is the number of nodes in G and a Hamilton path exists then
- Output $< G, a, b, k >$ and accept.
- Reject

Therefore

$M' =$ On input $< G, a, b >$

- Run M_H using $< G, a, b >$
- Run M or V using $< G, a, b, k >$
- If M and V accept then accept, otherwise reject.

If M_H accepts graph $< G, a, b >$ then G has a Hamilton path of length k and $< G, a, b, k >$ will be accepted by M or V since by definition of a Hamilton path, it will contain a simple path within it. If M or V accept then the graph has a simple path that spans the entire graph since k will be the same length as the graph and thus also be a Hamilton path.