

1. (Closure properties of \mathcal{P})(a) Proof that the class \mathcal{P} is closed under union:

Let $L_1, L_2 \in \mathcal{P}$ and let $L = L_1 \cup L_2$. We need to show that $L \in \mathcal{P}$. Since $L_1, L_2 \in \mathcal{P}$, let M_1 and M_2 be the single-tape deterministic TMs that decide L_1 and L_2 , respectively, where both run in polynomial time $O(n^k)$ for some $k \in \mathbb{R}^+$. Then create a multitape deterministic TM M that decides L as follows:

$M :=$ “On input w where w is a string:

1. Copy w from input tape 1 to tapes 2 and 3.
2. Run M_1 on tape 2. If M_1 accepts, *accept*.
3. Otherwise, run M_2 on tape 3. If M_2 accepts, *accept*. Otherwise, *reject*.”

Line 1 takes at most $O(n^2)$ time (where $n = |w|$) to complete. Since line 2 and line 3 each run $O(n^k)$ -time TMs, the whole multitape machine M runs in $O((n^k)^2) = O(n^{2k})$ time by Theorem 7.8 on page 254 of the text. Hence, M is a polynomial-time decider for L , which implies that $L \in \mathcal{P}$.

(b) Proof that the class \mathcal{P} is closed under intersection:

This is the same as (a) except M needs to be modified as follows:

$M :=$ “On input w where w is a string:

1. Copy w from input tape 1 to tapes 2 and 3.
2. Run M_1 on tape 2. If M_1 rejects, *reject*.
3. Otherwise, run M_2 on tape 3. If M_2 accepts, *accept*. Otherwise, *reject*.”

(c) Proof that the class \mathcal{P} is closed under complement:

Let $L \in \mathcal{P}$ and denote $\bar{L} = \Sigma^* \setminus L$. We need to show that $\bar{L} \in \mathcal{P}$. Since $L \in \mathcal{P}$, let M be the single-tape deterministic TM that decides L , where it runs in polynomial time $O(n^k)$ for some $k \in \mathbb{R}^+$. Then create a single-tape deterministic Turing machine M' that decides \bar{L} as follows:

$M' :=$ “On input w where w is a string:

1. Run M on input w . If M accepts, *reject*.
2. Otherwise, if M rejects, *accept*.”

Line 1 takes at most $O(n^k)$ time to complete since M is a $O(n^k)$ -time TM, so M' is also an $O(n^k)$ -time TM that decides \bar{L} in polynomial time, which implies that $\bar{L} \in \mathcal{P}$.

(Note that it is not necessary to make M' into a multitape deterministic TM since M' does not need to emulate two single-tape TMs as was the case (a) and (b).)

(d) Proof that the class \mathcal{P} is closed under concatenation:

Let $L_1, L_2 \in \mathcal{P}$ and let $L = L_1 \circ L_2$. We need to show that $L \in \mathcal{P}$. Since $L_1, L_2 \in \mathcal{P}$, let M_1 and M_2 be the single-tape deterministic TMs that decide L_1 and L_2 , respectively, where both run in polynomial time $O(n^k)$ for some $k \in \mathbb{R}^+$. Then create a multitape deterministic Turing machine M that decides L as follows:

$M :=$ “On input w where w is a string:

1. Consider all $(n + 1)$ possible ways w can be divided into the substrings x and y so that $w = xy$. Copy w from input tape 1 to tapes 2 and 3 so that x is on tape 2 and y is tape 3 starting with $x = w$ and $y = \varepsilon$.

- a. Run M_1 on tape 2. If M_1 rejects, consider the next pairing of x and y .
- b. Otherwise, run M_2 on tape 3. If M_2 accepts, *accept*. Otherwise, consider the next pairing of x and y .

2. If no pairing of x and y was accepted by both M_1 and M_2 , then *reject*.”

For a string w of length n , there are at most $n+1$ ways in which a string w can be split into two strings x and y so that $w = xy$. Hence, there are $O(n)$ calls to M_1 , which decides whether $x \in L_1$ in $O(n^k)$ steps, for a total of $O(n^{k+1})$ steps running M_1 altogether. Likewise, there are $O(n)$ calls to M_2 , which decides whether $y \in L_2$ in $O(n^k)$ for a total of $O(n^{k+1})$ steps running M_2 altogether.

Since line 1 emulates two single-tape TMs that each run for $O(n^{k+1})$ steps over the $O(n)$ calls, the whole multitape TM M requires no more than $O((n^{k+1})^2) = O(n^{2k+2})$ steps by Theorem 7.8 to emulate both TMs over these $O(n)$ calls. Hence, M runs in $O(n^{2k+2})$ time deciding whether $w \in L$, which makes it a polynomial-time decider for L , which implies that $L \in \mathcal{P}$.

(e) Proof that the class \mathcal{P} is closed under Kleene star:

Let $L \in \mathcal{P}$. We need to show that $L^* \in \mathcal{P}$. Since $L \in \mathcal{P}$, let M be the single-tape deterministic TM that decides L , that runs in polynomial time $O(n^k)$ for some $k \in \mathbb{R}^+$.

Consider all the possible ways that a string $w = w_1w_2 \cdots w_n$ of length n can be divided into substrings x_1, x_2, \dots, x_k for some k is $1 \leq k \leq n$ such that $w = x_1 \circ x_2 \circ \cdots \circ x_k$ where $x_j \in L$ for all $1 \leq j \leq k$. This can be done via dynamic programming in $O(n^3)$ time by constructing an $n \times n$ binary table T in which its upper triangular portion contains entries for all possible $\sum_{i=0}^n i = n(n+1)/2$ substrings of w from ε to w itself.

Define $T[i, j] = 1$ if and only if $w_i \cdots w_j \in L^*$ for all $1 \leq i \leq j \leq n$. The boundary condition is given by $T[i, i] = 1$ if and only if $w_i \in L$. The optimal substructure is given by $T[i, j] = 1$ if and only if both $T[i, k] = 1$ and $T[k, j] = 1$ for $i \leq k \leq j$.

Then create a deterministic Turing machine M^* that decides L^* as follows:

$M^* :=$ “On input w where $w = w_1 \cdots w_n$ is a string of length n :

1. If $w = \varepsilon$, *accept*. [[**Handle** $|w| = 0$ **case**]]
2. For $i \leftarrow 1$ to n do
3. For $j \leftarrow i$ to n do
4. $T[i, j] \leftarrow 0$. [[**Initialize table**]]
5. For $i \leftarrow 1$ to n do
6. $T[i, i] \leftarrow 1$ if $w_i \in L$. [[**Boundary condition**]]
7. For $\ell \leftarrow 2$ to n do [[ℓ **is length of substrings** > 1]]
8. For $i \leftarrow 1$ to $n - \ell + 1$ do [[i **is start position**]]
9. $j \leftarrow i + \ell - 1$, [[j **is end position**]]
10. $T[i, j] \leftarrow 1$ if $w_i \cdots w_j \in L$ using M .
11. For $k \leftarrow i$ to $j - 1$ do [[**Optimal substructure of join at position** k]]
12. $T[i, j] \leftarrow T[i, k] \wedge T[k + 1, j]$.
13. If $T[1, n] = 1$, *accept*, otherwise *reject*.”

Each of the $O(n^3)$ stages of this algorithm takes polynomial time with the most steps required for line 10, namely $O(n^t)$ for some $t \in \mathbb{R}^+$, in which M^* calls M to decide if a given string is in L .

The overhead of using multiple tapes for M^* to emulate M at most squares the time required. Hence, the algorithm has at most $O(n^{6t})$ steps, so that M^* is a polynomial-time decider for L^* , which implies that $L^* \in \mathcal{P}$.

2. (Closure properties of \mathcal{NP})

(a) Proof that the class \mathcal{NP} is closed under union:

Let $L_1, L_2 \in \mathcal{NP}$ and let $L = L_1 \cup L_2$. We need to show that $L \in \mathcal{NP}$. Since $L_1, L_2 \in \mathcal{NP}$, let M_1 and M_2 be the nondeterministic TMs that decide L_1 and L_2 in polynomial time. Then create an NTM M that decides L in polynomial time as follows:

$M :=$ “On input w where w is a string:

1. Run M_1 on w . If M_1 accepts, *accept*.
2. Otherwise, run M_2 on w . If M_2 accepts, *accept*. Otherwise, *reject*.”

For both lines 1 and 2, M uses nondeterminism when the machines M_1 and M_2 each run nondeterministically. This allows M to accept w if and only if either machine accepts it. Since both lines use NTMs that run in polynomial time, M runs in polynomial time to decide that $L \in \mathcal{NP}$.

(b) Proof that the class \mathcal{NP} is closed under intersection:

This is the same as (a) except M needs to be modified as follows:

$M :=$ “On input w where w is a string:

1. Run M_1 on w . If M_1 reject, *reject*.
2. Otherwise, run M_2 on w . If M_2 accepts, *accept*. Otherwise, *reject*.”

(c) Proof that the class \mathcal{NP} is closed under concatenation:

Let $L_1, L_2 \in \mathcal{NP}$ and let $L = L_1 \circ L_2$. We need to show that $L \in \mathcal{NP}$. Since $L_1, L_2 \in \mathcal{NP}$, let M_1 and M_2 be the NTMs that decide L_1 and L_2 , respectively, where both run in polynomial time.

Then create an NTM M that decides L in polynomial time as follows:

$M :=$ “On input w where w is a string:

1. Nondeterministically choose one of the $(n + 1)$ possible ways w can be divided into the substrings x and y so that $w = xy$.
 - a. Run M_1 on x .
 - b. Run M_2 on y .
2. If both M_1 and M_2 accepted, then *accept*. Otherwise, *reject*.”

For both parts of line 1, M uses nondeterminism when the machines M_1 and M_2 each make nondeterministic choices. This allows M to accept w if and only if M_1 accepts x and M_2 accepts y for some nondeterministic division of w into xy . Since both NTMs run in polynomial time that are each call once, M runs in nondeterministic polynomial time to decide if $L \in \mathcal{NP}$.

3. (Graph isomorphism)

Given that $L := \{ \langle G, H \rangle : G, H \text{ are isomorphic graphs} \}$, we want to prove that $L \in \mathcal{NP}$. This can be using either a deterministic polynomial-time verifier for L or a NTM that decides L in polynomial time. We show both ways though either way is sufficient for this problem.

The certificate used for the verifier and what is non-deterministically chosen by the NTM is a relabeling of vertices from G to H . Here we use the notation that $G = (V_G, E_G)$ and $H = (V_H, E_H)$ where V_G and V_H are vertex sets and E_G and E_H are edge sets.

(a) Deterministic polynomial-time verifier for L :

The following deterministic TM is a verifier \mathcal{V}_L for L :

$\mathcal{V}_L :=$ “On input $\langle\langle G, H \rangle, c\rangle$ where $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are graphs and $c : V_G \rightarrow V_H$ is the proposed relabeling of vertices of G to H :

1. Test if $|V_G| = |V_H|$ and $|E_G| = |E_H|$.
2. Test if c is proper relabeling, i.e a bijection from V_G to V_H
3. Test if $(c(u), c(v)) \in E_H$ for every distinct edge $(u, v) \in E_G$.
4. Test if $(c(u), c(v)) \notin E_H$ for every distinct pair of non-adjacent vertices $(u, v) \notin E_G$.
5. If all four tests pass, c must be a valid relabeling of G to match H so *accept*; otherwise, *reject*.”

Line 1 clearly takes $O(n)$ time (where n is the size of the input $\langle\langle G, H \rangle, c\rangle$).

Line 2 takes $O(|V_G| + |V_H|) = O(n)$ time.

Lines 3 and 4 takes at most $O(|V_G|^2 + |V_H|^2) = O(n^2)$ time each.

So \mathcal{V}_L runs in polynomial time $O(n^2)$, which means that \mathcal{V}_L is a deterministic polynomial-time verifier for L . This shows that $L \in NP$.

(b) Non-deterministic polynomial-time decider for L :

The following non-deterministic TM is a decider N_L for L :

$N_L :=$ “On input $\langle G, H \rangle$ where $G = (V_G, E_G)$, $H = (V_H, E_H)$ are graphs:

1. Test if $|V_G| = |V_H|$ and $|E_G| = |E_H|$.
2. Non-deterministically pick a relabeling $c : V_G \rightarrow V_H$ of vertices.
3. Test if $(c(u), c(v)) \in E_H$ for every distinct edge $(u, v) \in E_G$.
4. Test if $(c(u), c(v)) \notin E_H$ for every distinct pair of vertices $(u, v) \notin E_G$.
5. If all three tests pass, c must be a valid relabeling of G to match H so *accept*; otherwise, *reject*.”

Line 1 takes $O(n)$ time (where n is the size of the input $\langle G, H \rangle$), step 2 takes $O(|V_G| + |V_H|) = O(n)$ time, and steps 3 and 4 take $O(|V_G|^2 + |V_H|^2) = O(n^2)$ time.

So N_L runs in polynomial time $O(n^2)$, which means that N_L is a non-deterministic polynomial-time decider for L . This also shows that $L \in NP$.

4. (Powers of permutation)

Here a permutation of $N = \{1, 2, \dots, k\}$ is a bijective function $f : N \rightarrow N$. When f is composed with itself i times, it is denoted $f^{(i)}$ where $f^{(0)}$ represents the identity function on N . Given that $L := \{\langle f, g, i \rangle : f \text{ and } g \text{ are permutations with } f^{(i)} = g\}$, we want to prove that $L \in \mathcal{P}$ by giving a decider M that is polynomial in the length input in which $|\langle f, g, i \rangle| = O(k \log k + \log i)$.

Using the fact that $f^{(2j)} = f^{(j)} \circ f^{(j)}$ (where \circ denotes the operation of composition, not concatenation), this determination can be made in $\log i$ phases by creating the following deterministic TM M that decides L :

$M^* :=$ “On input w where $w = w_1 \dots w_n$ is a string of length n :

1. Verify f and g are valid permutations on $\{1, \dots, k\}$. If not, *reject*.
2. $h \leftarrow \{1, \dots, k\}$. [[Initialized to identity permutation]]
3. While $i > 0$ do
4. $h \leftarrow h \circ f$ if i is odd, [[Apply current f if low-order bit set of the current i]]
5. $f \leftarrow f \circ f$, [[Double the power f]]
6. $i \leftarrow \lfloor \frac{i}{2} \rfloor$. [[Halve i]]
7. If $g = h$, *accept*, otherwise *reject*.

Line 1 can be done easily in $O((k \log k)^2)$ time by comparing that both f and g each list the numbers in N exactly once iterating through the list of length $(k \log k)$ a total of k times, examining the $O(\log k)$ bits of each number at most $O(\log^2 k)$ times.

Line 2 takes $O(k \log k)$ time.

Lines 4 and 5 can each be done in $O(k \log k)$ time by reading the first permutation in $O(k \log k)$ time to create a new permutation by moving the $O(\log k)$ bits of each number of the second permutation to its position in the new permutation, which takes $O(k \log k)$ time.

Line 6 can be done in $O(\log i)$ time by chopping of the last bit of the bit representation of i .

Lines 3 through 7 run at most $\log i$ times, which gives a total running time for this algorithm of $O((\log i + k \log k)^2)$.

Hence, M is a polynomial-time decider for L , which implies that $L \in \mathcal{P}$.

5. (Two-way infinite tapes)

Here we show how two different ways of simulating a two-way infinite tape TM M using the cells $\{\dots, -2, -1, 0, 1, 2, \dots\}$ can give different running times.

(a) Straightforward simulation of M with a standard single tape TM M'

Here we simulate M by representing its two-way infinite tape as one-way infinite tapes on a *one-way* infinite multitape TM M' .

Map the non-negative tape cells of M to tape 1 of M' with the mapping $f : \{0, 1, 2, \dots\} \rightarrow \{1, 2, \dots\}$ where $f(x) = x + 1$. Next map the negative tape cells of M to tape 2 of M' with the mapping $f : \{-1, -2, \dots\} \rightarrow \{1, 2, \dots\}$ where $f(x) = -x$. Cell 0 of both tapes 1 and 2 have '#' symbol, which is added to tape alphabet, in order to detect when either head is trying to move left from its cell 1.

Then M' can emulate the operation of M by reversing the direction the head moves on tape 2. At any one time only one tape 1 or tape 2 of M' is active during the simulation of M , but not both. If one tape is active and the tape head moves to the left of cell 1, then the other tape becomes active with the tape head starting at cell 1 on that tape. If M takes $f(n)$ time to run, then M' takes $O(f(n))$ time since only constant time overhead is applied for each simulated movement of M on M' .

Then we can apply Theorem 7.8 on page 254 of the text to see that if the standard *one-way* infinite multitape TM M' runs in $O(f(n))$ time simulating the *two-way* infinite *single-tape* TM M , then there exists a standard *one-way* infinite *single-tape* TM M'' that runs in $O(f^2(n))$ time.

(b) Faster simulation of M with a standard single tape TM M''

Here we can simulate M onto M'' directly by using the following mappings:

Map the non-negative tape cells of M to the odd cells of M'' with the mapping $f : \{0, 1, 2, \dots\} \rightarrow \{1, 3, 5, \dots\}$ where $f(x) = 2x + 1$. Next map the negative tape cells of M to tape 2 of M'' with the mapping $f : \{-1, -2, \dots\} \rightarrow \{2, 4, 6, \dots\}$ where $f(x) = -2x$. Again cell 0 is marked with '#' symbol.

Then we have the transfer function of M'' move two movements to the left or two the right for each single movement of M in which direction is reversed when moving between even cells. If the tapes head attempts to move left from cell 1 to cell 0, then it moves to cell 2, and visa versa.

Finally, the tape alphabet of M'' has to be doubled from M so that it can mark intermediate cells when moving two movements to the left or right.

Doing all this only gives $O(1)$ overhead in M'' per movement in M so that M'' runs in $O(f(n))$ time.