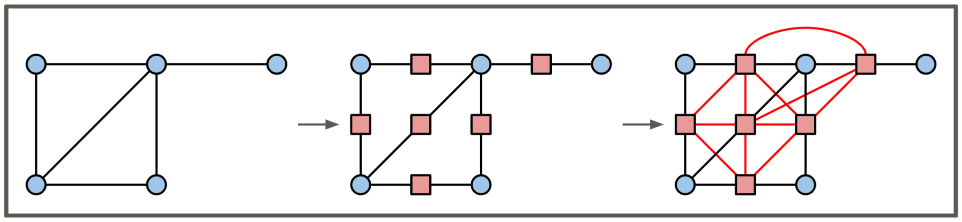
1. A *strongly independent* set is a subset of vertices S in a graph G such that for any two vertices in S, there is no path of length less than or equal to two in G. Prove that *Strongly Independent Set* is NP-hard.

Solution: To show that strongly independent set is NP-hard we do a reduction from independent set.

For any graph G we construct a new graph H where we add vertices that correspond to each edge, these edge vertices are connected to the vertices that the original edges were connected to and to any edge vertex where the original edges shared a vertex.



$\rightarrow$ Let G have an independent set X, then X is a strongly independent set in H. If u, v in G are 2 length away, then the path in H is u connects to the edge vertex of u, which connects to the edge vertex of v which connects to v, so u, v are 3 length away in H. Therefore if G has an independent set of size k, then H has a strongly independent set of size k.

$\leftarrow$ Let H have a strongly independent set Y. Then let w be an edge vertex in Y where the original edge in G connects to vertices a, b. The path from any vertex z (not a) to w is less than or equal to the path from z to a (or b). This is because a connects only to the edge vertices that represent the edges a connects to in G, but these edge vertices that connect to a are all connected to one another. So any path to a has to go through one of these edge vertices. If this edge vertex is w then the path to w is 1 shorter then the path to a. If this edge vertex is not w then this path could go to w instead of a next so the path to w is the same as the path to a. This mean for every edge vertex in Y we can swap it out for a vertex that the edge connected to in G and preserve strong independence. Let Y' be the strongly independent set obtained from swapping out all of the edge vertices. Y' is an independent set in G because if all of the paths from u to v in Y' are greater than 2 in H then u and v cannot be adjacent in G. Therefore if H has a strongly independent set of size k, then G has an independent set of size k.

We have proved both directions in the reduction and constructing the new graph is polynomial time. Therefore because independent set is NP-hard, strongly independent set must also be NP-hard.

■

2. Are the following problems in P, NP, co-NP, NP-Hard, NP-complete? Either way, prove it.

   (a) A *kite* is a graph on an even number of vertices, say 2n, in which n of the vertices form a clique and the remaining n vertices are connected in a tail that consists of a path joined to one of the vertices of the clique. Given a graph and a goal g, the max kite problem asks for a sub-graph that is a kite and contains 2g nodes. What complexity classes does $kite$ belong in?

Solution: The Kite problem is NP-Complete. To prove this, we will have to prove that the problem belongs to NP class and then we prove that it is NP-Hard.

To prove that the problem is NP:

- We will verify that g vertices form a Clique. For all pairs of vertices (x, y) such that $(x, y) \in E$ and this would be $O(n^2)$ where n is the number of vertices of graph.
- Verify that the remaining k vertices are part of the tail attached to a vertex on the clique. This would take $O(n^2)$ time.

This part is for reference. Now we need to prove that the Kite problem is NP Hard. Let us take the Clique Decision problem where given a graph G, we find the max sized clique. We can reduce the Boolean Satisfiability problem to this. Consider an expression $F = (x_1 \vee x_2) \wedge (x_1' \vee x_2') \wedge (x_1 \vee x_3)$. Let us create a graph with vertices constructed as $< x_1, 1 >, < x_2, 1 >, < x_1', 2 >, < x_2', 2 >, < x_1, 3 >, < x_3, 3 >$ where the second term in each vertex denotes the clause number. We construct the edges such that:

- No two vertices belonging to the same clause are connected.
- No variable is connected to its complement.

This forms a clique of size 3. Thus, if there are k clauses, we will get k sized clique. Hence the Clique Decision problem is NP-Hard, since we reduced the satisfiability problem to Clique Decision problem.

This is just a proof for why Clique Decision is a NP Hard problem. This is just for reference.

Now we can reduce the CLique Decision problem to the Kite problem. Given an input graph G = (V, E). Let us convert the graph from Clique to Kite.

Given a graph H = (V_H, E_H). We will construct a graph G by adding a tail of size g to all vertices in H.

$\rightarrow$ Suppose H has a Clique of size g. We know that there is a tail of size g that is connected to one of the vertices of the clique in H. Thus G has a Kite of size 2g.

$\leftarrow$ Suppose G has a Kite of size 2g. Since we know that there are g vertices forming a tail in G and there are $|V_H|$ tails, there should be a clique of size g in the remaining graph which is H. Thus H has a Clique of size g.

Hence the Kite problem is NP-Hard and thus NP-Complete.

■

(b) A *4kite* is exactly the same problem, but this time $g = 4$. What complexity classes does $4kite$ belong in?

Solution: The 4kite problem belongs to the class P. We can compute, for each vertex of the graph, the kite of size 8 that contains that vertex and has its tail in a specified subset of the vertices. This can be done by considering all possible ways to extend the tail of the kite and selecting the one that leads to 8 sized kite. We can go through all sets of 8 vertices in the graph and check whether it forms a Kite.

We are choosing 8 vertices and all the edges among them from the graph and seeing if they form a kite in constant time. Choosing 8 out of n vertices nC8 would lead to a runtime of O(n^8).

■

3. Let $T = \left\{\langle M\rangle | M \text{ is a TM that accepts } w^R \text{ whenever it accepts } w\right\}$. Show that T is undecidable.

Solution: For the sake of argument, suppose there is an algorithm DECIDEL_4 that correctly decides the language T. Then we can solve the halting problem as follows:

```
DECIDEHALT(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        if x=10
            run M on input w
        return TRUE

  if DECIDEL_4(⟨M′⟩)
      return TRUE
  else
      return FALSE
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.
Then M' accepts *every* input string x.
In particular, M' accepts w^R and w.
So DECIDEL_4 accepts the encoding $\langle M'\rangle$.
So DECIDEHALT correctly accepts the encoding $\langle M, w\rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.
Then M' accepts 01 but not 10.
So DECIDEL_4 rejects the encoding $\langle M'\rangle$.
So DECIDEHALT correctly rejects the encoding $\langle M, w\rangle$.

In both cases, DECIDEHALT is correct. But that's impossible, because HALT is undecidable. We conclude that the algorithm DECIDEL_4 does not exist. ■

4. Are the following problems decidable or undecidable? If the language is decidable, explain why, and if it's not, prove it.

(a) $L_{5a} = \left\{\langle M\rangle | M \text{ TM accepts string } 1011 \text{ in } |1011|^6 \text{steps}\right\}$

Solution: The language L_{5a} is decidable and we describe the algorithm DECIDEL_{5a} as follows:

```
DECIDE$L_{5a}$($\langle M, w\rangle$):
    run 1011 on $M$ for $|1011|^6$ steps
    if M reaches accept state
        return TRUE
    else
        return FALSE
```

Since we can come up with a decider DECIDEL_{5a} that can decide correctly for L_{5a} wether a TM M belongs to the language or not by simulating M for a finite amount of time, we prove that L_{5a} is decidable. ■

(b) $L_{5b} = \left\{\langle M\rangle | M \text{ TM does not accept any strings in } |w|^6 \text{steps}\right\}$

Solution: For the sake of argument, suppose there is an algorithm DECIDEL_{5b} that correctly decides the language L_{5b}. Then we can solve the halting problem as follows:

```
DECIDEHALT($\langle M, w\rangle$):
    Encode the following Turing machine $M'$:
        $M'(x)$:
            run $M$ on input $w$
            return TRUE
    if DECIDE$L_{5b}$($\langle M'\rangle$)
        return FALSE
    else
        return TRUE
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.
Then M' accept any input x in finite steps. That is, for some long x it would accept x in $|x|^6$ steps.
So DECIDEL_{5b} rejects the encoding $\langle M'\rangle$.
So DECIDEHALT correctly accepts the encoding $\langle M, w\rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.
Then M' diverges on *every* input string x.
In particular, M' does not accept any string in $|x|^6$ steps.
So DECIDEL_{5b} accepts the encoding $\langle M'\rangle$.
So DECIDEHALT correctly rejects the encoding $\langle M, w\rangle$.

In both cases, DECIDEHALT is correct. But that's impossible, because HALT is undecidable. We conclude that the algorithm DECIDEL_{5b} does not exist. ■