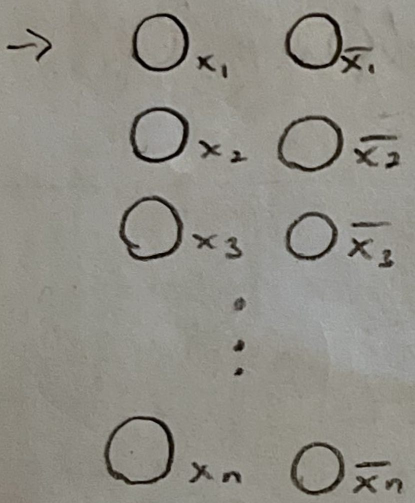
CS 334 | Pblm # 10

- Name: [illegible] & [illegible]
- Pledge: "I plege my honor I've abided by the Stevens Honor System"

#1. 1. We construct a graph with 2n nodes, s.t each pair of nodes represents a literal & its negation, for each literal in our inputed formula.

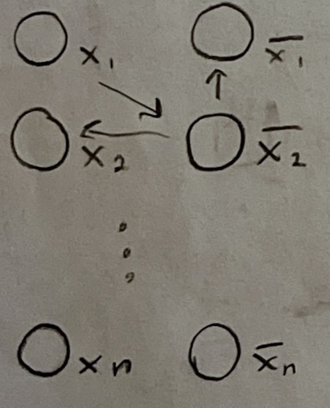


2. Then, for every clause we construct edges as the following,

- If the clause is of the form $(x_i \vee x_j)$, add the directed edges $(\neg x_i, x_j)$ and $(\neg x_j, x_i)$
- If the clause is of the form $(\neg x_i \vee x_j)$, add directed edges (x_i, x_j) and $(\neg x_j, \neg x_i)$.

That is we encode for each clause its equivalent implication $(\neg x_i \rightarrow x_j)$ and its negation.

3.



(Algo starts)
Run DFS on the first node x_1 in the graph.
We cross out every node we traverse.
If we end up traversing a cycle that contains a node & its compliment, we reject.
(We'd know this by returning to a marked node which we can mark uniquely for each x)

Else, we accept.

// Note: we can accept since by showing there are no cycle's between a node & its compliment, we show there exists no contradiction in our implications.
→ There must exist some truth assignment to satisfy $\emptyset$.

#2. High level algorithm:

```
A: on input $\phi$, some satisfiable expression & $x_i$
    1. Replace all instances of $x_i$ with true.
    2. Ask genie about modified expression $\phi'$.
    3. If satisfiable, run A on $\phi'$ & $x_{i+1}$. Save correct truth assignment for $x_i$.
    4. Else, change all instances of what were $x_i$'s
       (now trues) to false. Run from step 2.
    0. Check if we've finished on all $x_{n+1}$ literals.
```

We run A on the original input ϕ & x_1. When we finish we're left with all correct truth assignments for $x_1 \to x_n$. DFS $\in \Theta(n)$ ✓

#2ii. n

#2iii. The algorithm A continuosly modifies ϕ so ϕ' has $x_i = T$ initially. This essentially simulates having ϕ s.t x_i is = True. If this truth assignment causes ϕ' to be unsatisfiable, this implies by setting x_i to be true, ϕ has no viable solution. Thus x_i must be false. We continue with this logic until all x_i are determined.

#2iv. Similar approach, construct algorith H. # Assume G has hamiltonian cycle.

```
H: on input G<V, E>
    1. Ask genie #2 if the graph has a hamiltonian cycle.
    2. If yes, remove an edge (in order) from the graph.
       Repeat from 1.
    3. If no, add the previously removed edge back to the graph.
       It was part of the Hamiltonian Cycle. Repeat from 1.
```

We do this until we've run through every edge in our graph. We end up removing all edges that are not part of the hamiltonian cycle → leaving us with a graph only containing the edges of the hamiltonian cycle in G. Since we can have at most $(\frac{n(n-1)}{2})$ edges, our algorithm $\in P$