Recall that for a class of languager $\mathcal{C}$, we had $A \subseteq \Sigma^*$ is $\mathcal{C}$-HARD if for every $B \in \mathcal{C}$, $B \leq_m A$. Then if $A \in \mathcal{C}$ as well, A is $\mathcal{C}$-COMPLETE.

There is a snag, however. There's a second requirement we should place on completeness. For example, if our $\mathcal{C} = NP$, then certainly any reduction from all $B \in NP$ to A will suffice to show A is hard for NP. The worse the required reduction, the harder A must be. (Actually this isn't true, b/c a reduction could solve the problem and ask some trivial y/n to a simple TM) But for A to be complete for NP we want to be able to solve every problem in NP using A. This means we need a poly time reduction from everything to A (this can be nondeterministic poly time in theorey but in practice never is). We denote this by $\leq_m^{P}$ (or sometimes $\leq_p$). Let's formalize this.

This resource bound can be other time/space choices for other classes

A function $f : \Sigma^* \to \Sigma^*$ is a <u>polynomial time computable function</u> if some polynomial time TM M exists that, on input $w \in \Sigma^*$, halts with just $f(w)$ on its tape.

A language A is <u>polynomial time mapping reduceable</u> to a language B, written $A \leq_m^P B$, if there is a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that $\forall w \in \Sigma^*$,

$$w \in A \iff f(w) \in B.$$

The function f is called the <u>polynomial time reduction</u> of A to B.

Why does this matter? Well, let's look at a useful theorem.

Thm) If $A \leq_m^p B$ and $B \in P$, then $A \in P$

Pf) Suppose $A \leq_p B$ and $B \in P$.
Then there is a polytime reduction f from A to B and a poly time TM M_B that decides B. Consider the following TM M_A.

```
M_A = "On input w,
    1) Compute f(w)
    2) Run M_B(f(w))
    3) Accept if M_B does and reject if not"
```

Clearly, $L(M_A) = A$ and M_A decides A in poly time. □

The same arguement gives us a theorem about NP.

Thm) If $A \leq_m^p B$ and $B \in NP$, then $A \in NP$.

We can get negative results with resource bounds too.
Before we had $A \leq_m B$ and $A \notin RE \Rightarrow B \notin RE$

Cor) If $A \leq_m^p B$ and $A \notin P$, then $B \notin P$.

Pf) If $B \in P$, then there is a polytime decider D for B. There's also a polytime reduction f from A to B. So $D(f)$ decides A in polytime, which is nonsense, so no such D exists. □

Cor) If $A \leq_m^p B$ and $A \notin NP$, then $B \notin NP$.

Pf) Identical to the prior proof but with NP. □

You can also get results in the opposite direction.

Thm) If $A \le_m^P B$ and $A \in$ NP-HARD, then $B \in$ NP-HARD.

Pf) Since $A \in$ NP-HARD, $\forall C \in NP$, we have $C \le_m^P A$. But $A \le_m^P B$, so $\forall C \in NP$, it must be the case that $C \le_m^P B$. Thus $B \in$ NP-HARD.

Cor) If $A \le_m^P B$ and $B \notin$ NP-HARD, then $A \notin$ NP-HARD.

Pf) Since $B \notin$ NP-HARD, $\exists C \in NP$ such that $C \not\le_m^P B$. If $A \in$ NP-HARD then $C \le_m^P A \le_m^P B$, so $C \le_m^P B$, which is nonsense, so $A \notin$ NP-HARD. □

Remember that these reductions can be read as

$A \le_m^P B$ = "A is no harder than B"

or

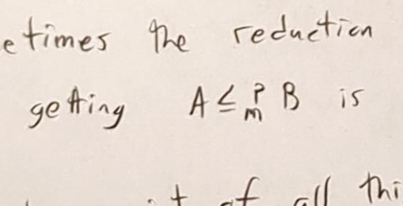
$A \le_m^P B$ = "B is at least as hard as A".

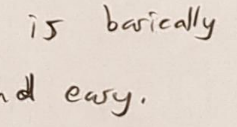
To show a language A is NP-COMPLETE, there are two ways to do so conveniently.

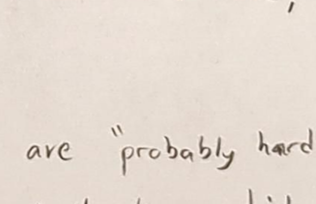
- Show $A \in NP$ and $B \le_m^P A$ for some $B \in$ NP-HARD.
- Show for some $B \in$ NP-COMPETE that $A \le_m^P B$ and $B \le_m^P A$.

Sometimes the reduction $B \le_m^P A$ is basically the same in reverse, so getting $A \le_m^P B$ is quick and easy.

The key point of all this is that NP problems are "probably hard", but NPC problems are "almost certainly hard". We don't know which of the following cases are true, so we don't know how hard these problems actually are.







If we show any $B \in NPC$ and $B \in P$, then we have $P = NP$. Similarly, if we could show any $B \in NP$ and $B \notin P$, then we would know $P \cap NPC = \emptyset$.

Obviously, we need a first NPC problem to do anything useful with these definitions. (technically knowing $A \notin NP$ and $A \in NP\text{-HARD}$ is also weful, but let's move on). The problem we pick is SAT.

$$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$$

Proving $SAT \in NPC$ is hard. The thearm is known as the Cook-Levin Theorem. We will prove this if time allows, but let's first just accept it as true and look at a bunch of NPC problems.

3SAT

A literal of a Boolean formula is a variable or its negation.

A clause is a finite disjunction of literals (i.e. $x_1 \vee \overline{x}_2 \vee x_4$).

A Boolean function in conjunctive normal form (cnf) is a finite conjunction of clauses (i.e. $(x_1 \vee x_2) \wedge (\overline{x}_3 \vee x_4)$). This is also called a cnf-formula.

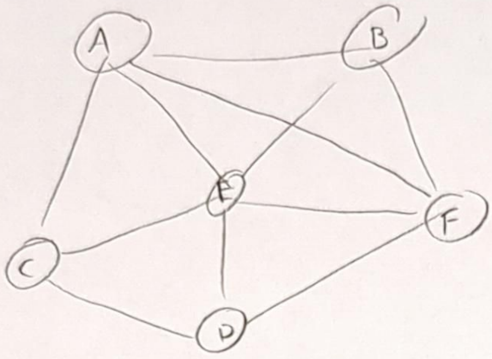
A 3cnf-formula is a cnf-formula such that each clause has exactly 3 literals.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$$

That $3SAT \in NPC$ follows from a modification of the Cook-Levin Theorem as it's earier than a reduction $SAT \le_m^P 3SAT$. It's often the case that 3SAT is earier to work with.

Given an undirected graph $G=(V,E)$, a clique is a $V' \subseteq V$ such that $\forall u,v \in V'$ (with $u \neq v$), $\{u,v\} \in E$. A k-clique is a clique of size k.

Ex)



$\{A, B, E, F\}$ is a 4-clique.

We define the decision problem for clique as

$$\text{CLIQUE} = \{ \langle G,k \rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}.$$

Thm) CLIQUE $\in$ NPC

Pf) We show CLIQUE $\in$ NP by giving a verifier V.

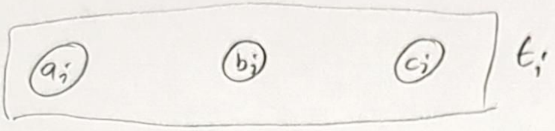
```
V = "On input $\langle G,k,c \rangle$,
  1) Check that $c = V' \subseteq V$ and reject if not
  2) If $|V'| \neq k$, reject,
  3) For each $u,v \in V'$ with $u \neq v$,
       a) If $(u,v) \notin E$, reject
  4) Accept"
```

We now show CLIQUE $\in$ NP-HARD by showing $3SAT \leq_m^P$ CLIQUE.

Let $\phi = (a_1 \vee b_1 \vee c_1) \wedge \dots \wedge (a_k \vee b_k \vee c_k)$ be a 3cnf formula with k clauses.

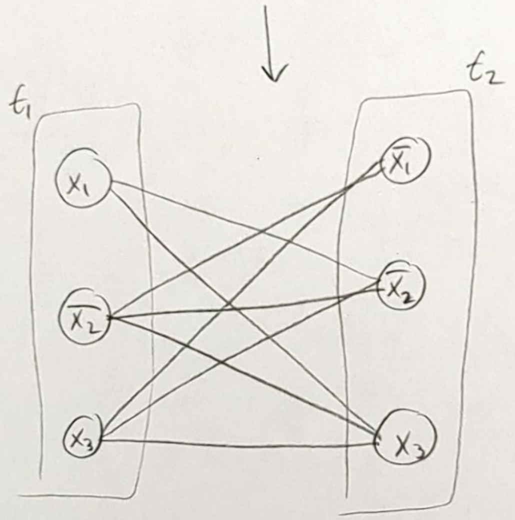
We'll construct a graph $G_\phi = (V,E)$ out of ϕ.

The vertices of G_ϕ are placed into k groups of 3, $t_1, \dots, t_k$. Each t_i corresponds to the i^{th} clause of ϕ, and each vertex of t_i corresponds to it's literal.



We now connect every distinct pair of vertices in V except that with the same group or with contradictory values.

$$\phi(x_1, x_2, x_3) = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$$



Any 2-clique (ie. a connected pair of vertices) will satisfy ϕ.

We claim: $f(\langle\phi\rangle) = \langle G_\phi, k\rangle$ is a reduction of 3SAT to CLIQUE.

Suppose ϕ has a satisfying assignment. To build a k-clique, we must pick one vertex from each t_i since the t_i's have no internal edges and there are precisely k of them. In fact, from t_i, we pick whichever of a_i, b_i, or c_i is true in the satisfying assignment of ϕ (if more than one is, pick one at random). None of these choices can correspond to a logical contradiction, so by construction, there is an edge between each of them. As such, we have a k-clique.

In the reverse direction, suppose we have a k-clique. Again, there are k t_i's with no internal edges, so we must have one vertex from each. By construction, these correspond to non-conflicting assignments to ϕ's variables. If not every variable has been assigned, just make the remainders all true. Thus ϕ is satisfiable.

□

Given an undirected graph $G=(V,E)$, an <u>independent set</u> is a $V' \subseteq V$ such that $\forall u,v \in V'$, $\{u,v\} \notin E$. A <u>k-independent-set</u> is an independent set of size k.

The decision problem for this is

$$IS = \{\langle G,k\rangle \mid G \text{ is an undirected graph with a } k\text{-independent-set}\}.$$

<u>Thm)</u> $IS \in NPC$

<u>pf)</u> We'll prove $IS \leq_m^p CLIQUE$ and $CLIQUE \leq_m^p IS$.

In either case, the reduction is the same.

$$f(\langle G=(V,E),k\rangle) = \langle (V,\bar{E}),k\rangle \qquad (\bar{E} = \{\{u,v\} \mid \{u,v\} \notin E,\ u,v \in V, \text{ and } u \neq v\})$$

Notice that $f(f(\langle G,k\rangle)) = \langle G,k\rangle$, so it suffices to show that (V,E) contains a k-clique iff $(V,\bar{E})$ contains a k-IS.

Suppose G contains a k-clique V'. Then $\forall u,v \in V'$ $(u \neq v)$ we have $\{u,v\} \in E \iff \forall u,v \in V'$ $(u\neq v)$, $\{u,v\} \notin \bar{E} \iff (V,\bar{E})$ contains a k-IS V'.

□

If we wanted to show the reverse reduction, then we do the following.

Suppose G contains a k-IS V'. Then $\forall u,v \in V'$ $(u \neq v)$, $\{u,v\} \notin E$ $\iff \forall u,v \in V'$ $(u \neq v)$, $\{u,v\} \in \bar{E} \iff (V,\bar{E})$ contains a k-clique V'.

□

We have 3SAT and a couple graph problems. Let's pick up a numerical problem next.

The <u>subset sum problem</u> is as follows. Given a finite collection of numbers S and a target value t, determine if there is an $S' \subseteq S$ such that $\sum_{x \in S'} x = t$.

The language for this problem is (set or multiset)

$$\text{SUBSET-SUM} = \{\langle S, t\rangle \mid S \text{ is a finite collection of numbers with an } S' \subseteq S \text{ such that } \sum_{x \in S'} x = t\}$$

Thm) SUBSET-SUM $\in$ NPC

We first give a verifier V for SUBSET-SUM.

```
V = "On input ⟨⟨S,t⟩, c⟩,
    1) Check that $c \subseteq S$
    2) Check that $\sum_{x \in c} x = t$
    3) If both tests pass, accept; reject otherwise"
```

Obviously, V runs in poly time and verifies SUBSET-SUM.

We now show $3SAT \leq_m^p$ SUBSET-SUM.

Let ϕ be a 3cnf-formula with variables $x_1, \ldots, x_\ell$ and clauses $c_1, \ldots, c_k$. We map x_i to a number y_i and $\overline{x}_i$ to a number z_i. We'll pick y_i, z_i, and t such that we can only pick one of y_i and z_i. Here's how we do that. The i^{th} digit of y_i and z_i are both 1, as is the i^{th} digit of t. Every other number we pick will have the i^{th} digit be 0 so that the only possible way to get 1 in the i^{th} digit is to pick exactly one of y_i and z_i.

We also want digit $l+j$ of y_i to be 1 if x_i satisfies clause c_j. Similarly, digit $l+j$ of z_i is 1 if $\overline{x_i}$ satisfies clause c_j. Both are 0 otherwise. In total, both y_i and z_i should have $l+k$ digits.

Further, digit $l+j$ of t should be 3 for each j. This is because a clause can have up to 3 +1's from a y_i or z_i.

We also add a g_i and h_i that have digit $l+i$ be 1 and every other digit be 0. This allows us to fill in up to 2 unsatisfied literals of a clause. We give a full example of this construction.

$$\phi(x_1,x_2,x_3,x_4) = \underbrace{(x_1 \vee x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee x_3 \vee x_4)}_{c_2} \wedge \underbrace{(x_2 \vee \overline{x_3} \vee x_4)}_{c_3}$$

		1	2	3	4	1	2	3
x_1	y_1	1	0	0	0	1	0	0
$\overline{x_1}$	z_1	1	0	0	0	0	1	0
x_2	y_2	0	1	0	0	1	0	1
$\overline{x_2}$	z_2	0	1	0	0	0	0	0
x_3	y_3	0	0	1	0	1	1	0
$\overline{x_3}$	z_3	0	0	1	0	0	0	1
x_4	y_4	0	0	0	1	0	1	1
$\overline{x_4}$	z_4	0	0	0	1	0	0	0
	g_1	0	0	0	0	1	0	0
	h_1	0	0	0	0	1	0	0
	g_2	0	0	0	0	0	1	0
	h_2	0	0	0	0	0	1	0
	g_3	0	0	0	0	0	0	1
	h_3	0	0	0	0	0	0	1
	t	1	1	1	1	3	3	3
			l				k	

Pick x_1, x_3, x_4 (and x_2)

or

$y_1, y_2, y_3, y_4, g_2, g_3$

```
  1000 100   y1
  0100 101   y2
  0010 110   y3
+ 0001 011   y4
  1111 322
  0000 010   g2
+ 0000 001   g3
  1111 333 = t
```

We'll formally argue this reduction works now. Suppose ϕ is satisfiable. Then there is some assignment of $x_1, \dots, x_\ell$ that satisfies ϕ. Then selecting the corresponding y's and z's gives us that the first ℓ digits are all 1 and the last K digits are all at least 1 since each clause is satisfied but no more than 3 by construction. We can fill in any of these last k digits that are not 3 with g's and h's.

Now suppose there is a selection of numbers that adds up to t. By construction, we must select eactly 1 of each y_i or z_i, and for each j, we must have at least one i for which a selection of y_i or z_i has it's $\ell+j^{th}$ digit as 1. This correspends to a satisfying assignment, so ϕ is satisfiable.

Lastly, the reduction must be done in poly time. The table has size $2(k+\ell)^2$, and each entry can be determined in linear time at worst. So the runtime is $O(n^3)$.

□

We now consider the Hamiltonian path problem. A Hamiltonian path in a directed graph $G=(V,E)$ is a path that visits each vertex exactly once. We define the relevant language as

$$\text{HAM-PATH} = \{\langle G,s,t\rangle \mid G \text{ is an undirected graph with a Hamiltonian path from } s \text{ to } t\}$$

We will show HAM-PATH is NPC via a reduction from 3SAT.

Thm) HAM-PATH ∈ NPC

Pf) We first give a polytime verifier for HAM-PATH to show it's in NP.

```
V = " On input ⟨⟨G,s,t⟩,c⟩,
    1) Check that c is a path p
    2) check that p starts with s and ends at t
    3) check that p visits each u∈V exactly once
    4) If any check fails, reject
    5) Accept"
```

We now reduce $3SAT \leq^p_m$ HAM-PATH to get that HAM-PATH is in NP-HARD (and thus NPC).

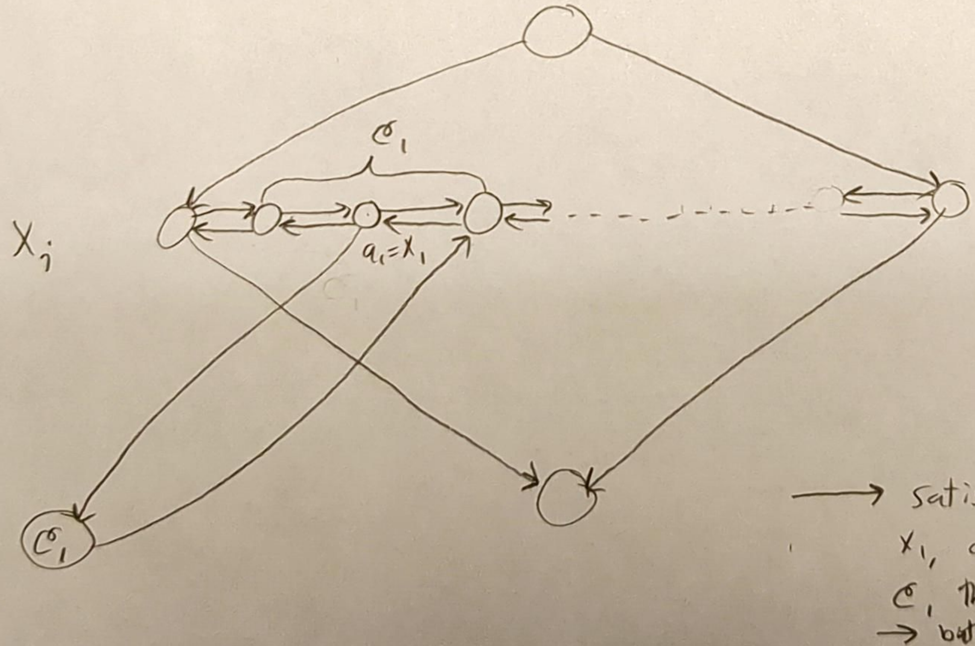
We'll create a gadget that mimics a variable assignment and attacch it to clause nodes, indicating the clause is satisfied. The trick is in the directionality.

Given a 3cnf formula of l variables,

$$\phi = \underbrace{(a_1 \vee b_1 \vee c_1)}_{C_1} \wedge \dots \wedge \underbrace{(a_k \vee b_k \vee c_k)}_{C_k},$$

we make a vertex for each C_i and a gadget for each x_i

$C_1 = (x_i \vee ? \vee ?)$

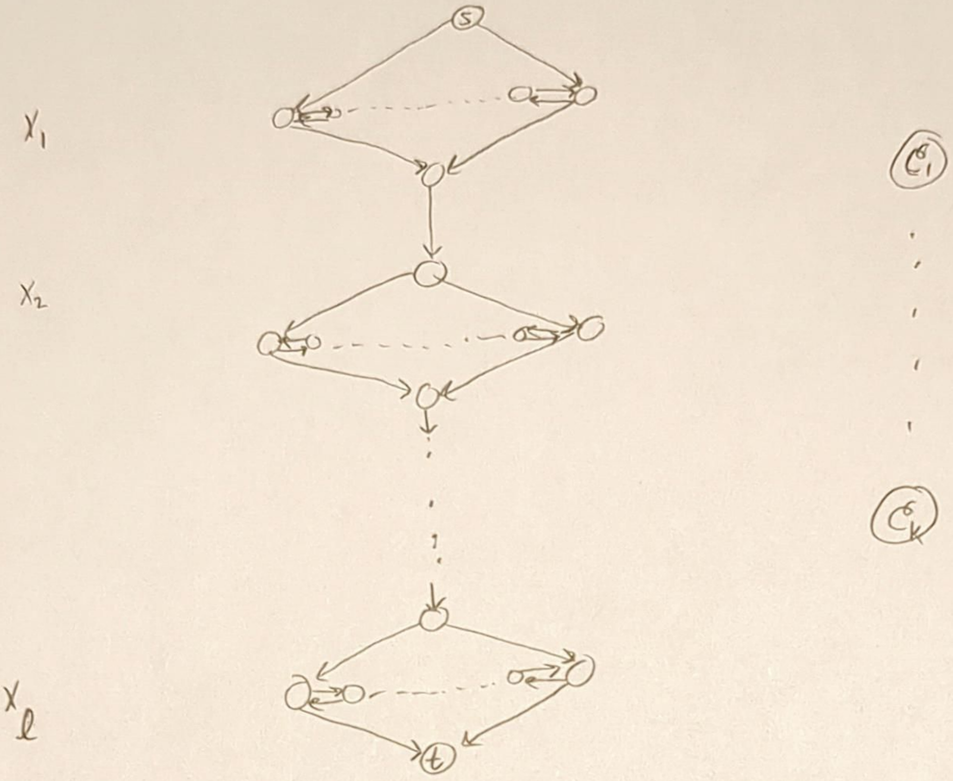


→ Assign x_1 T

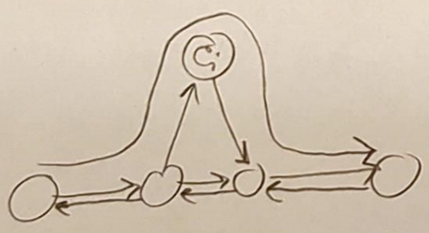
← Assign x_1 F

→ satisfies C_1, which contains x_1, and we can include C_1 the vertex on a Hampath → but not ←

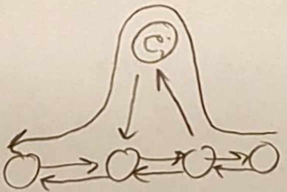
The entire construction looks like this



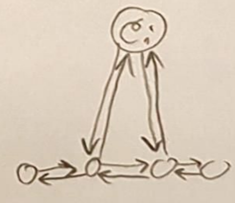
To see why this construction works, consider a clause C_i $C_i = a_i \vee b_i \vee c_i$. Then if C_i is satisfied, then at least one of a_i, b_i, and c_i must be a true literal. Then in The corresponding X_j for that literal we can include the middle vertices exactly once by going exclusively left or exclusively right. Right means X_j = True and left means x_j = False ($\overline{X_j}$ = True). The C_i vertex can be detoured to without revisiting a vertex only when the variable is assigned correctly.



X_j must be true



xj must be false



X_j can be true or false

$C_i = X_j \vee \overline{X_j} \vee ?$

In the opposite direction, if there's a Ham path, it must detor to each C_j. Where it does so corresponds to a variable assignment, and such a path can <u>only</u> go one direction through that variable, so we can't generate a contradictory assignment.

Both of these arguements generalize, so the construction works as a (poly time) reduction.

$\square$

The <u>undirected Hamiltonian Path problem</u> is the same except now you have an undirected graph.

$$\text{UHAM-PATH} = \{ \langle G,s,t \rangle \mid G \text{ is a graph with a Ham-path from } s \text{ to } t \}.$$

<u>Cor)</u> UHAM-PATH $\in$ NPC

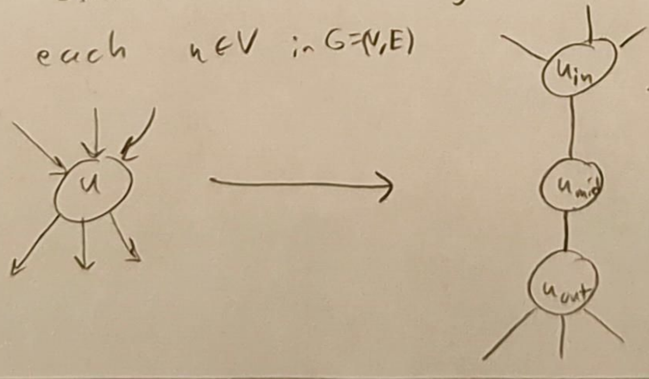
<u>pf)</u> We can transform an undirected graph $G=(V,E)$ into a directed graph $G' = (V,E')$ by defining $E' = \{ (u,v) \in V^2 \mid \{u,v\} \in E \}$.

Suppose $p = v_1 \dots v_{|V|}$ is a UHam-path in G. Then $\forall i, \{v_i, v_{i+1}\} \in E$, hence $\forall i, (v_i, v_{i+1}) \in E'$. Thus p is a Ham-path in G'.

Suppose $p = v_1 \dots v_{|V|}$ is a Ham-path in G'. Then $\forall i, (v_i, v_{i+1}) \in E'$, hence $\forall i, \{v_i, v_{i+1}\} \in E$. Thus p is a UHam-path in G.

This shows UHAM-PATH $\leq^P_m$ HAM-PATH, so UHAMPATH $\in$ NP.

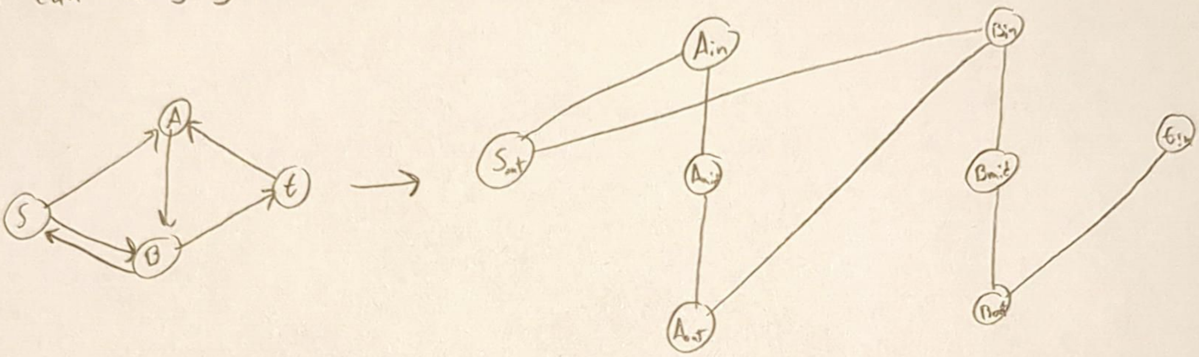
To get the reverse reduction to complete the proof, we need to simulate directionality. Except for s and t, we transform each $u \in V$ in $G=(V,E)$



That is we create 3 vertices u_{in}, u_{mid}, and u_{out} for each $u \in V \setminus \{s,t\}$. There is an (undirected) edge from v_{out} $(v \in V)$ to u_{in} if $(v,u) \in E$, an edge between u_{in} and u_{mid}, the same for u_{mid} and u_{out}, and an edge from u_{out} to v_{in} $(v \in V)$ if $(u,v) \in E$. s only gets s_{out} and t only gets t_{in}.

To get a UHAM-PATH, you can only go through the u vertices one way, and since ins are only connected to outs (and vice versa), you can only go one direction.

Ex)



Notice the edge (t,A) is lost but that it's useless anyway.
The same is true of edge (B,s).

Let this construction be $f(\langle G,s,t\rangle) = \langle (V',E'), s_{out}, t_{in}\rangle$.

Suppose G has an s-t Ham-path $s u_1 \dots u_k t$. Then clearly

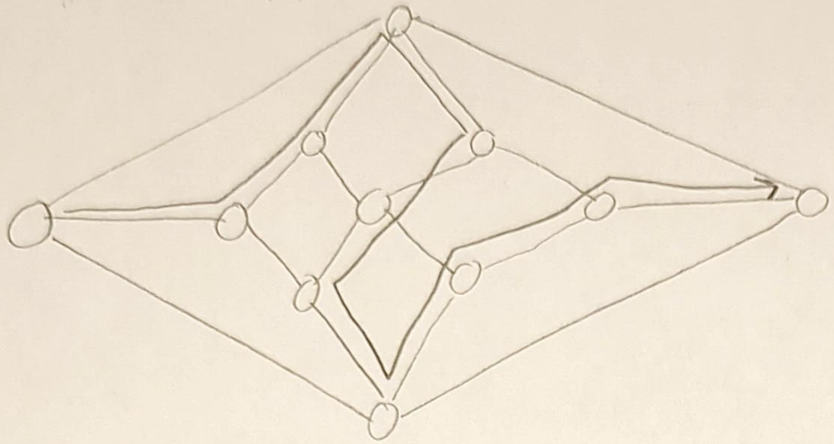
$s_{out}\ u_{1,in}\ u_{1,mid}\ u_{1,out}, \dots, u_{k,in}\ u_{k,mid}\ u_{k,out}\ t_{in}$ is a UHam-path in G'.

Now suppose we have an s_{out} - t_{in} Ham path in G'. We obrerved earlier that an out must go to an in which must then go to the corresponding out via the mid) between since otherwise the mid is lost. This repeats from s_{out} to t_{in}, which coresspands to an s-t Ham-path in G.

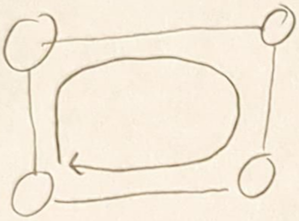
□

A **Hamiltonian cycle** is a cycle that visits each vertex exactly once,

Ex)



Contains a Ham-path but not a Ham-cycle



A Hamiltonian cycle.

The language associated is

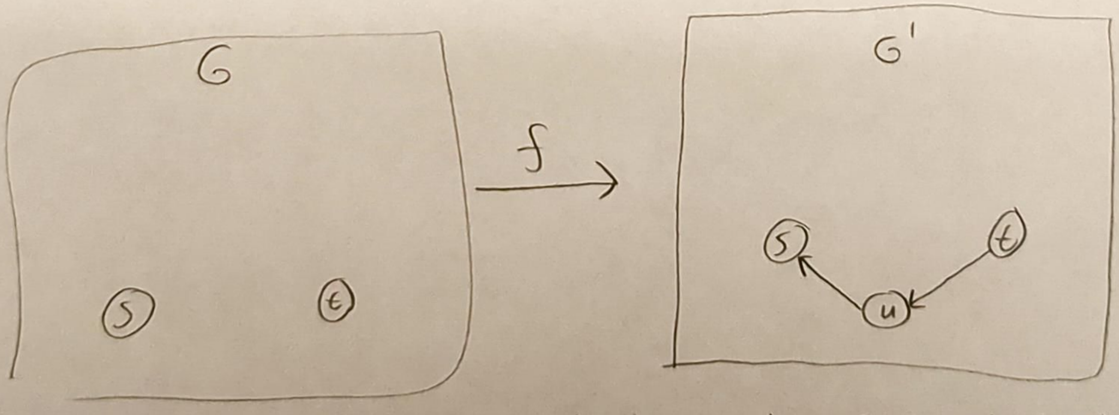
HAM-CYCLE = {⟨G⟩ | G is a directed graph containing a Hamiltonian cycle}.

The undirected version can be similarly defined.

Thm) HAM-CYCLE ∈ NPC.

Pf) A Ham cycle is a Ham-path plus an edge, so obviously its in PV, hence it's in NP.

We can reduce HAM-PATH $\le_m^P$ HAM-CYCLE easily.



Just add the vertex u with the edges (t,u) and (u,s).

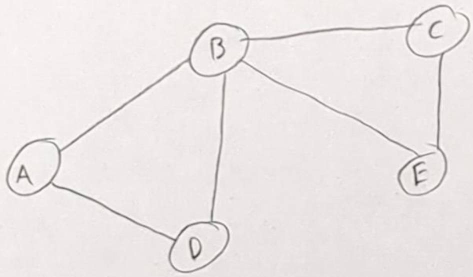
If there is an s-t Ham-path in G, specified by edges $e_1 \dots e_k$, then clearly $e_1 \dots e_k\ (t,u)\ (u,s)$ is a Ham-cycle in G'.

Now suppose $e_1 \dots e_{k+2}$ is a Ham-cycle in G'. Then it must be the case that $\exists j$ such that $e_j = (t,u)$ and $e_{i+1} = (u,s)$. WLOG, assume $j = K+1$. Then $e_1 \dots e_k$ is an s-t Ham-path in G. □

Next on our list of problems is vertex cover. This is a minimization problem that we will reduce the maximization problem Clique to. Beyond that it will introduce the notion of cover problems.

Given a graph $G=(V,E)$, a <u>vertex cover</u> of G is a $V' \subseteq V$ such that for every edge $(a,b) \in E$, $a \in V' \vee b \in V'$.

<u>Ex)</u>



{A, B, C} is a vertex cover of this graph. It is also minimal.

The corresponding language is

$$VC = \{\langle G,k \rangle \mid G \text{ is a graph with a vertex cover of size } k\}.$$

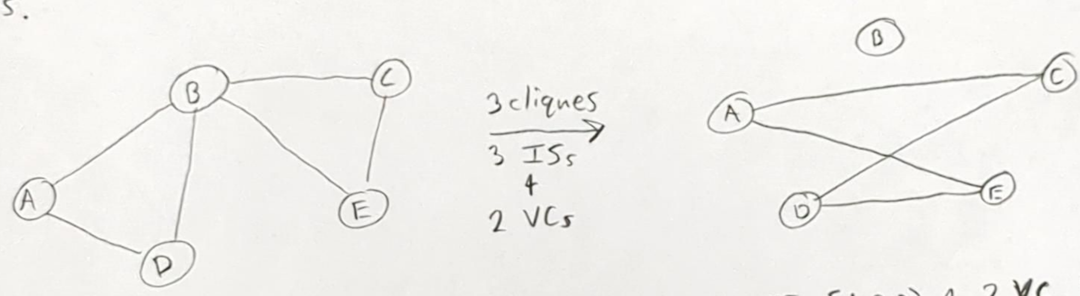
<u>Thm)</u> VC ∈ NP-COMPLETE

<u>Pf)</u> We will do the usual thing and show VC ∈ NP by providing a poly time verifier. Afterwards, we will show VC ∈ NP-HARD by showing CLIQUE $\leq_m^P$ VC.

```
V = "On input $\langle\langle G,k\rangle, c\rangle$,
    1) Check that $c = V' \subseteq V$.
    2) check that $|V'| = k$
    3) Check that $\forall (a,b) \in E$, $a \in V' \vee b \in V'$
    4) If any check fails, reject.
    5) Accept"
```

The reduction from CLIQUE $\leq_m^P$ VC is fairly simple. Let $G = (V,E)$ be a graph. We construct the edge set compliment $\overline{E} = \{\{u,v\} \mid \{u,v\} \notin E\}$ as we did with the reduction to IS.

We then claim that $f(\langle (V,E), k\rangle) = \langle (V,\overline{E}), |V|-k\rangle$ is a reduction (and is obviously poly time). The intuition is that $E \to \overline{E}$ turns cliques into independent sets, and we need vertices from outside the ISs.



3 Clique $\{A, B, D\}$ $\longrightarrow$ 3 IS $\{A,B,D\}$ & 2 VC $\{C,E\}$.

Suppose $G = (V,E)$ has a k-clique V'. Then we know V' is a k-IS in $\overline{G} = (V,\overline{E})$. By definition, $\forall u,v \in V'$, $(u,v) \notin \overline{E}$. In other words, if $(u,v) \in E$, then either $u \notin V'$ or $v \notin V'$. Let $V'' = V \setminus V'$. Then this means $(u,v) \in E \Rightarrow u \in V''$ or $v \in V''$. But this is the definition of a vertex cover, so V'' is a $(|V|-k)$-VC.

Now suppose $G = (V,E)$ and $\overline{G} = (V,\overline{E})$ and $\overline{G}$ has a $(|V|-k)$-VC $V' \subseteq V$. Then by definition, $\forall u,v \notin V'$, $(u,v) \notin \overline{E}$. But then if $V'' = V \setminus V'$, V'' is a $|V| - (|V|-k) = k$-IS, which we know is a k-clique in G.

Thus we have $\text{CLIQUE} \leq_m^p \text{VC}$, so $\text{VC} \in \text{NP-HARD}$.

Since $\text{VC} \in \text{NP}$ as well, we have $\text{VC} \in \text{NP-COMPLETE}$. $\square$