How to Prove B ∈ NPC:

1. Prove that B ∈ NP.
   You can do this with an NTM or certificate.
2. Choose problem A, which is known to be NPC.
3. Describe a polytime reduction of A to B, $A \leq_p B$.
   - Show how to transform any instance of A into an equivalent instance of B.
   - Argue that the transformation is polytime.

Exact Cover (XCOV):

- Instance: ⟨U, C⟩ where U is the universe set of elements and C is collection (set) of subsets of U.
- Question: Is there C' ⊆ C s.t.
  a. C' covers every element in U. (We call C' a **cover**.)
     I.e. The union of all subsets in C' is U.
  b. ∀ A, B ∈ C', s.t. A≠B, A∩B = Ø (We call C' an **exact cover**.)
     I.e. All of the sets in C' are disjoint.
     I.e. There is no element in C' that is in 2 different sets in C'.
     I.e. The intersection of any two subsets in C' should be empty. That is, each element of U should be contained in at most one subset of C'.
- **Theorem 10.1:** XCOV ∈ NPC.

  Proof:

  a. **Proof that XCOV ∈ NP:**
     Use a NTM to non-deterministically choose a subset of C and verify that it covers all the elements and no two sets in the chosen subset of C overlap.
  b. **Show that CNF-SAT $\leq_p$ XCOV:**
     Given a CNF formula F = C1 ∧ … ∧ Cm, where each Cj is a clause, with variables X1, …, Xn, we will construct U and C s.t. F is satisfiable iff (U, C) has an exact cover.
     Let $Cj = l_j^1 \lor l_j^2 \lor \dots l_j^{kj}$, $1 \leq j \leq m$ and l_j^t is a literal.

     For example, suppose F = (¬x1 ∨ ¬x2 ∨ x3) ∧ (¬x1 ∨ x2 ∨ ¬x3) ∧ (¬x1 ∨ x3) ∧ (x1 ∨ x2 ∨ x3).
     We have 3 variables, x1, x2 and x3.
     We have 4 clauses.

     Every variable gets an element in the universe.
     Every clause also gets its own element.
     Every literal inside the clause also gets its own element l_j^t.
     We define $U = \{xi \mid 1 \leq i \leq n\} \cup \{Cj \mid 1 \leq j \leq m\} \cup \{l_j^t \mid 1 \leq j \leq m, 1 \leq t \leq kj\}$.
     $\{xi \mid 1 \leq i \leq n\}$ represents the set of variables
     $\{Cj \mid 1 \leq j \leq m\}$ represents the set of clauses.
     $\{l_j^t \mid 1 \leq j \leq m, 1 \leq t \leq kj\}$ represents the set literals.

     Here are the sets in C:

     1. For each xi, $1 \leq i \leq n$:
        a. $Pi = \{xi\} \cup \{l_j^t \mid l_j^t = xi, 1 \leq j \leq m, 1 \leq t \leq kj\}$
           Pi is the union of each variable and all the literals that are the same as the variable.

P1 = {x1, l_4^1} in our example. l_4^1 is the only literal that is the same as x1.

b. Ni = {xi} U {l_j^t |l_j^t = ¬xi , 1 ≤ j ≤ m, 1 ≤ t ≤ kj}
Ni is the union of each variable and all the literals that are the not of the variable.
N1 = {x1, l_1^1, l_2^1, l_3^1} in our example. l_1^1, l_2^1, l_3^1 are literals that are the not of x1.

2. For every clause Cj 1 ≤ i ≤ m:
S_j^t = {Cj, l_j^t} 1 ≤ t ≤ kj.
Each literal inside a clause is grouped with that clause.
E.g. S_1^1 = {C1, l_1^1}
3. For each l_j^t, 1 ≤ j ≤ m, 1 ≤ t ≤ kj:
L_j^t = {l_j^t}.
Each literal gets its own set.

Here's the intuition for coming up with this idea:

- First, to cover each of the clauses, Cj, you must choose only one S_j^t for each Cj. The idea of choosing the S_j^t is that l_j^t is a literal that will satisfy Cj.
E.g. Suppose that, from our example above, we want to use ¬x1 to make C1 true. Then, we choose S_1^1.
- Next, to cover the variables, if l_j^t is true, choose Ni and if l_j^t is false, then choose Pi.
E.g. Suppose that, from our example above, we want to use ¬x1 to make C1 true. Then, that means that x1 is false. We choose P1 instead of N1 because by choosing P1, we are preventing ourselves from using x1 to make another clause true later on. Since an exact cover forbids the same element in multiple subsets, by choosing P1 now, we are eliminating x1.

  Another way of looking at this is the fact that we need to cover each variable xi. Each variable xi is only in Pi and Ni. Suppose that l_j^t, which corresponds to xi, is true. To cover xi, we can only use Pi or Ni. If we use Pi, then l_j^t will be in 2 different sets, Pi and S_j^t, which is not allowed. Hence, we must choose Ni. The same reason applies for why we choose Pi if l_j^t is false.
- Lastly, to get the literals that we did not choose earlier, we select then from L_j^t.

c. **Claim: F is satisfiable iff (U, C) has an exact cover.**
Proof:
(=>)
Let τ be a truth assignment satisfying F.
Then let C' be defined as the following:
1. For each xi 1 ≤ i ≤ n:
   a. If τ(xi) = True, then put Ni in C'.
   b. If τ(xi) = False, then put Pi in C'.

2. For each clause Cj $1 \le j \le m$ let $l_j^{\tau j}$ be a literal such that $\tau(l_j^{\tau j})$ = True:
    a. If there are more than one literal that satisfies Cj then pick one arbitrarily.
    b. Then put $S_j^{\tau j}$ in C'.
3. For each l_j^t not covered by 1 or 2:
    a. Add L_j^t to C'.

Claim: C' is an exact cover.

a. Proof that C' covers all elements:
   This is clear as
   (1) includes all variables,
   (2) includes all the clauses and
   (3) includes all the leftover literals.
b. Proof that no two sets in C' intersect:
   The only possibility of intersection is between $S_j^{\tau j}$ and Pi or Ni.
   Suppose $l_j^{tj} \in S_j^{tj} \cap$ Ni.
   Since $l_j^{tj} \in$ Ni, $l_j^{tj} = \neg$xi.
   However, since $l_j^{tj} \in S_j^{tj}$, l_j^{tj} = True.
   This is a contradiction.
   If $\tau(\neg$xi$)$ = 1, then τ(xi) = 0 and Ni $\notin$ C'.

Hence, C' is an exact cover.

(<=)

Let C' be an exact cover of (U, C).

Define

$$\boxed{\tau(x_i) = \begin{cases} 1 & \text{if } N_i \in \mathcal{C}' \\ 0, & \text{if } P_i \in \mathcal{C}' \end{cases}}$$

These sets must contain at least one of them because these are the only sets that contain xi. It cannot contain both because then it would not be an exact cover.

Claim: τ satises F.

To prove this, it suffices to show that it satisfies every clause, Cj.

Proof:

- Let S_j^{tj} be the unique set in C' that contains Cj.
- S_j^{tj} = {Cj, l_j^{tj}}
- Suppose l_j^t = xi.
    - Then, C' does not contain Pi.
    - This means that C' contains Ni.
    - This means that τ(xi) = 1.
    - This means that τ satisfies Cj.

- Suppose $l_j^t = \neg xi$.
  - Then, C' does not contain Ni.
  - This means that C' contains Pi.
  - This means that τ(¬xi) = 0.
  - This means that τ(xi) = 1.
  - This means that τ satisfies Cj.

d. Argument for polynomial time:

The construction of ⟨U, C⟩ from ⟨F⟩ is polytime w.r.t |⟨F⟩|.
We just have to argue that |⟨U, C⟩| isn't too big.

|⟨F⟩| = n + $\sum_{j=1}^{m} kj$

(n is the number of variables.)

($\sum_{j=1}^{m} kj$ is the total number of literals in all the clauses.
There are m clauses and each clause has kj literals.)

|⟨U, C⟩|

- |⟨U⟩| = n + m + $\sum_{j=1}^{m} kj$, where m is the number of clauses.
- The number of all the sets in C =
  $\sum_{j=1}^{m} kj$ +2 $\sum_{j=1}^{m} kj$ +2n+ $\sum_{j=1}^{m} kj$

  The first $\sum_{j=1}^{m} kj$ is for L_j^t.

  The 2 $\sum_{j=1}^{m} kj$ is for S_j^t.

  The 2n is for the xi in Pi and Ni.

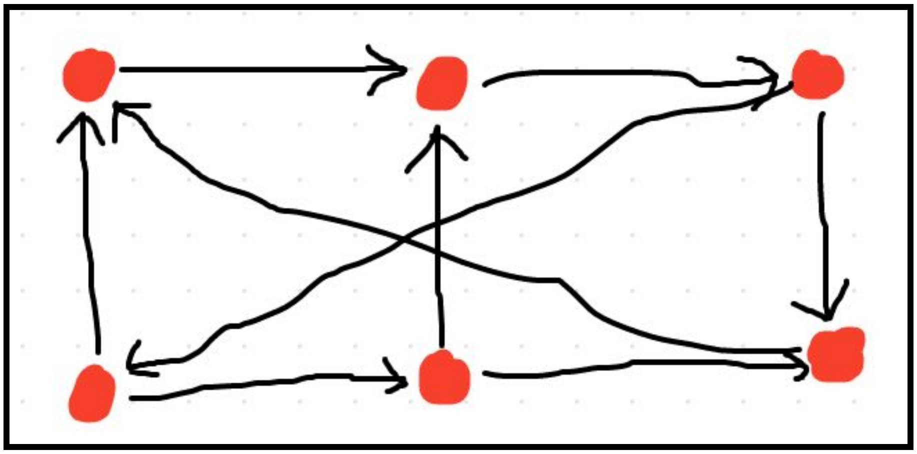
  The final 2 $\sum_{j=1}^{m} kj$ is for each literal in Pi or Ni.

  = 2n + 4 $\sum_{j=1}^{m} kj$

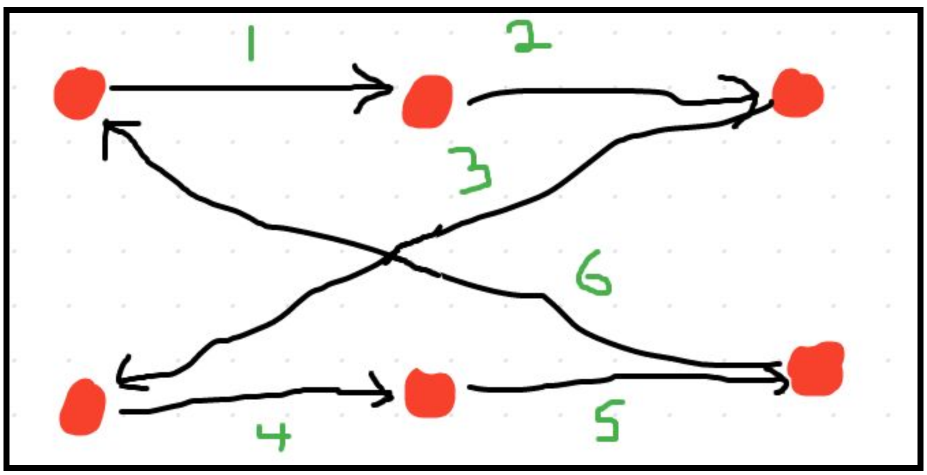
  Hence, this is polynomial in |⟨F⟩|.

Directed Hamiltonian Cycle (DHC):

- Instance: ⟨G⟩, G = (V, E) is a directed graph.
- Question: Does G have a HC? A HC is a path that goes through all nodes exactly once. E.g. The graph below has a HC.



This is the HC:



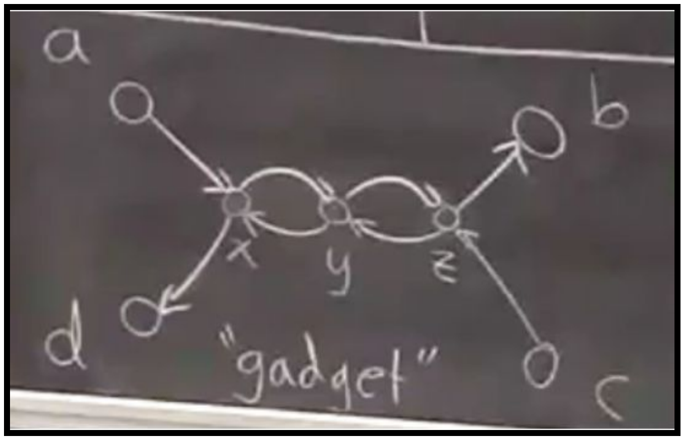
- **Theorem 10.2:** DHC ∈ NPC

Proof:

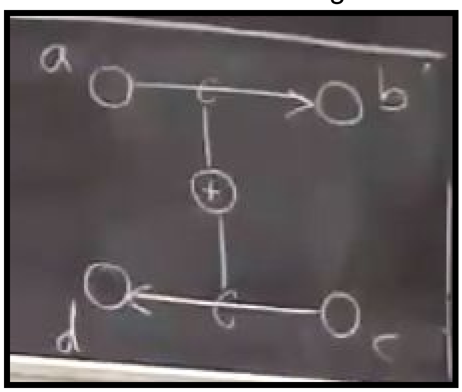
a. **Proof that DHC ∈ NP:**
A NTM guesses a path and a deterministic TM goes through the path and confirms it.
b. **Show that XCOV $\leq_p$ DHC:**
We will use **gadgets** to show this.
Gadgets are a sub construction that are embedded in the overall construction.
We will define a gadget for this proof as follows:
    - We have 2 pairs of nodes (A,B) and (C,D) that connect to three nodes x, y and z.
    - Other nodes can be connected to A, B, C or D, but there are no other nodes that are connected to x, y or z.
    - If we have a HC in the larger graph the gadget is part of, it must contain one of either 2 paths:
        1. A → x → y → z → B We call this path A → B.

2. C → z → y → x → D We call this path C → D.

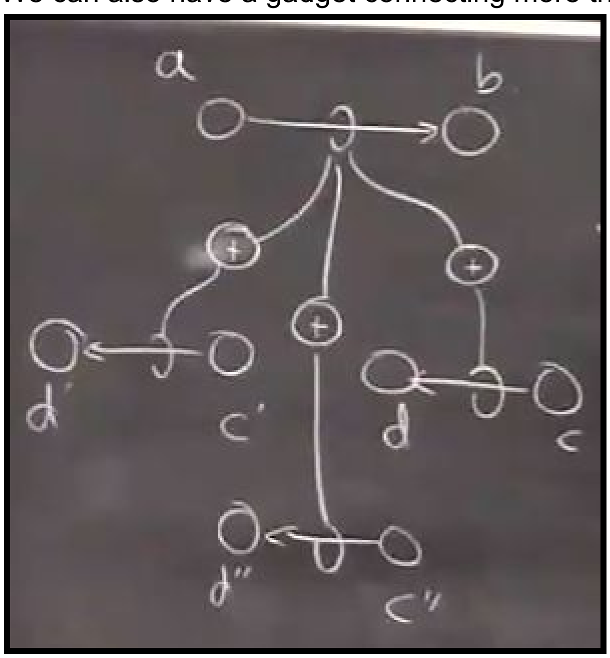
- This creates the equivalent of an xor in paths.
- Here is a picture of the gadget.



We will use the below diagram as a shorthand for the above diagram.

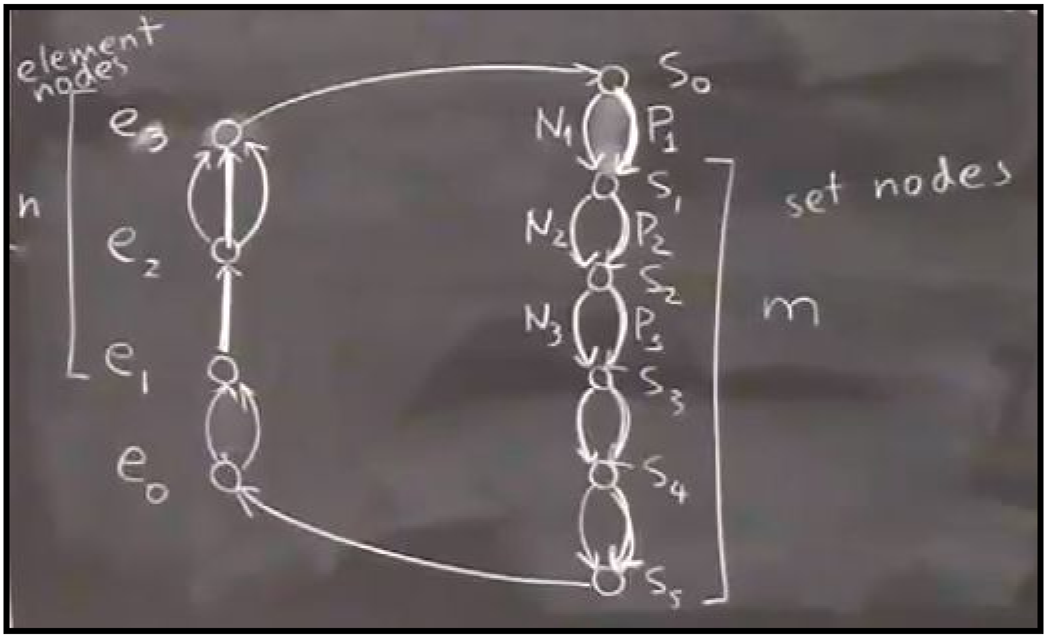


We can also have a gadget connecting more than 2 paths.



Given (U, C), which is an instance of XCOV, construct a directed graph G = (V, E) such that (U, C) has exact cover iff G has HC.
Let U = {u1, u2, ..., un} and C = {A1, ..., Am} such that Aj ⊆ U.
Construct G with n + m + 2 nodes.
n is the number of elements in U.
m is the number of sets in C.

Here is an example/informal construction of how to construct G.



The edge from e0 to e1 corresponds to u1.
The edge from e1 to e2 corresponds to u2.
The edge from e2 to e3 corresponds to u3.

The edge from s0 to s1 corresponds to A1.
The edge from s1 to s2 corresponds to A2.
The edge from s2 to s3 corresponds to A3.
The edge from s3 to s4 corresponds to A4.
The edge from s4 to s5 corresponds to A5.

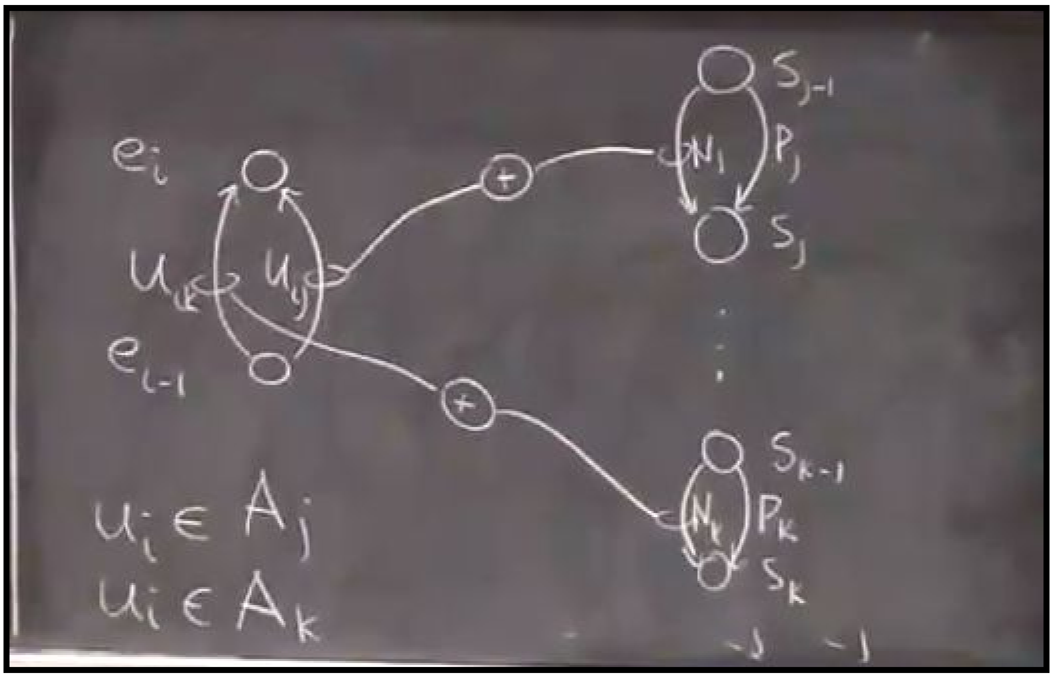
We construct set nodes, si, with an extra s0 such that there are 2 edges from si−1 to si.
We will call one edge Pi and the other edge Ni.
The Ni edges are part of a gadget.
If our HC path goes through a Pi, I will put the corresponding Ai in the exact cover. Otherwise, I won't.

We construct element nodes, ei, with an extra e0 such that there are multiple edges from ei−1 to ei.
Each element has an edge for each set it is in.
E.g. u1 is in 2 sets, so that's why there are 2 edges from e0 to e1.
E.g. u2 is in 1 set, so that's why there is 1 edge from e1 to e2.

E.g. u3 is in 3 sets, so that's why there are 3 edges from e2 to e3.
To find which set(s) each ui is in, we will couple each of the edges in ei with the corresponding Ni of the appropriate set via a gadget.
Suppose element ui ∈ Aj and ui ∈ Ak.

- We call the edges uij and uik and connect uij to Ni with a gadget and uik to Nk with a gadget.
- The number of gadgets for each si is the number of elements in the set.

Here is a picture of the above:



We have an edge from en to S0.
We have an edge from sn to e0.

Here is the formal construction of how to construct G:
G = (V, E)
V = {ei : 0 ≤ i ≤ n} U {sj : 0 ≤ j ≤ m} U [additional gadget nodes]
E = {uij : ui ∈ Aj} U {Nj,Pj : 1 ≤ j ≤ m} U {(em, s0),(sm, e0)} U [edges for the gadgets]

c. **Claim: (U, C) has exact cover iff G has a HC.**
Proof:
(=>)
Suppose C' is an exact cover of (U, C).
Then, choose edges as follows:

1. For every Aj ∈ C', choose Pj.
   For every Aj ∉ C', choose Nj.
   This connects the set nodes.
2. For each ui ∈ U choose a unique uij such that ui ∈ Aj and Aj ∈ C'.
   This connects the element nodes.
   This must exist because C' is an exact cover.
3. Choose edges (en, s0),(sn, e0).

These form a HC because we've connected the set nodes, the element nodes and the connecting edges.

(<=)
Suppose G has a HC H.
Let C' = {Aj |H uses Pj}
C' is an exact cover.

1. Every ui ∈ U is covered by some Aj ∈ C'.
   H must go from ei−1 to ei using uij s.t. ui ∈ Aj.
   H does not use Nj because of the coupling by the gadget.
   Therefore, H must use Pj.
   Therefore, Aj ∈ C0.
   Therefore, ui is covered by C'.
2. Sets in C' are disjoint.
   Suppose for contradiction, Aj, Ak ∈ C' such that j ≠ k, Aj ∩ Ak ≠ ∅.
   Let ui ∈ Aj ∩ Ak.
   H uses both Pj and Pk, because Aj, Ak ∈ C'.
   H does not use Nj or Nk, because it would visit the same nodes twice.
   H uses uij and uik, because of gadgets.
   H uses two edges from ei−1 to ei.
   Therefore, H is not HC. This is a contradiction.

d. **Argument for polynomial time:**

$|\langle U, C\rangle| = n + \sum_{j=1}^{m} |Aj|$

n is the number of elements in U.

$\sum_{j=1}^{m} |Aj|$ is the sum of the sizes of the various sets in C.

|G| = # of nodes + # of edges
\# of nodes = n + m + 2

\# of edges = $2m + 2 + \sum_{j=1}^{m} |Aj|$

2m → Every set has 2 edges.
2 → There's an edge from en to s0 and e0 to sn.
$\sum_{j=1}^{m} |Aj|$ → Recall that each ei-1 to ei has x edges, where x is the number of sets ui is in.
I.e. e0 to e1 has 2 edges if u1 is in 2 sets.

|G| is polynomial w.r.t |⟨U, C⟩|.
Hence, construction of ⟨G⟩ from ⟨U, C⟩ is polytime w.r.t |⟨U, C⟩|.

Undirected Hamiltonian Cycle (UHC)

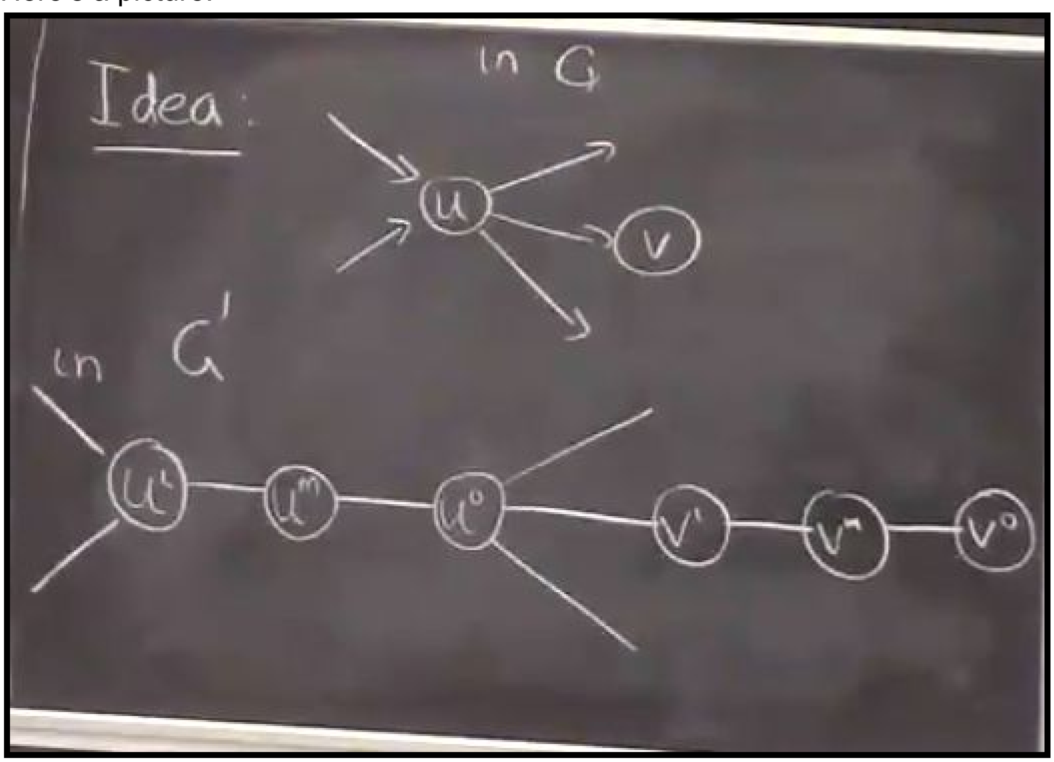
- Instance: ⟨G⟩, where G = (V, E) is undirected graph.
- Question: Does G have a HC?
- **Theorem 10.3:** UHC ∈ NPC.

Proof:

a. **UHC ∈ NP**
b. **Show that DHC $\le_P$ UHC:**
Given a directed graph G = (V, E), construct an undirected graph G' = (V', E') such that G has a HC iff G' has a HC.

Here's the idea:

- If G has a node "u" that's connected to a node "v" we construct 6 extra nodes: ui, um, uo and vi, vm, vo.
ui stands for "u in".
um stands for "u middle".
uo stands for "u out".
Similar naming patterns for vi, vm and vo.
- ui connects to um which connects to uo.
- vi connects to vm which connects to vo.
- uo connects to vi.
- Here's a picture:



c. **Claim: G has a HC iff G' has a HC.**

Proof:
(=>)
Suppose G has a HC.
Just use the appropriate triple of nodes that replaces the original.

(<=)
Suppose G' has an HC.
Start it from um for some u.
Without loss of generality assume the next node is uo. If its ui just reverse the direction.
After uo the next node is vi for some v.
The next node must be vm. This is because if vi doesn't go to vm now, in order to visit vm later, it must revisit some node.
By the same reasoning, the next node must be vo.
Hence, there must be this pattern of xi -- xm -- xo.
This easily translates to the HC from G' to G as we just choose the above pattern for the appropriate nodes.

d. **Argument for polynomial time:**
For each node, x, in G, there are 3 nodes, xi, xm and xo, in G'.
Hence, |⟨G'⟩| is polynomial w.r.t to |⟨G⟩|.

TSP:

- Instance: ⟨G, wt, b⟩, where G = (V,E), wt is a weight function and b is a budget.
- Question: Is there a tour, HC, of G of wt ≤ b?
- **Theorem 10.4:** TSP ∈ NPC