SubSet Sum (SUBSUM):

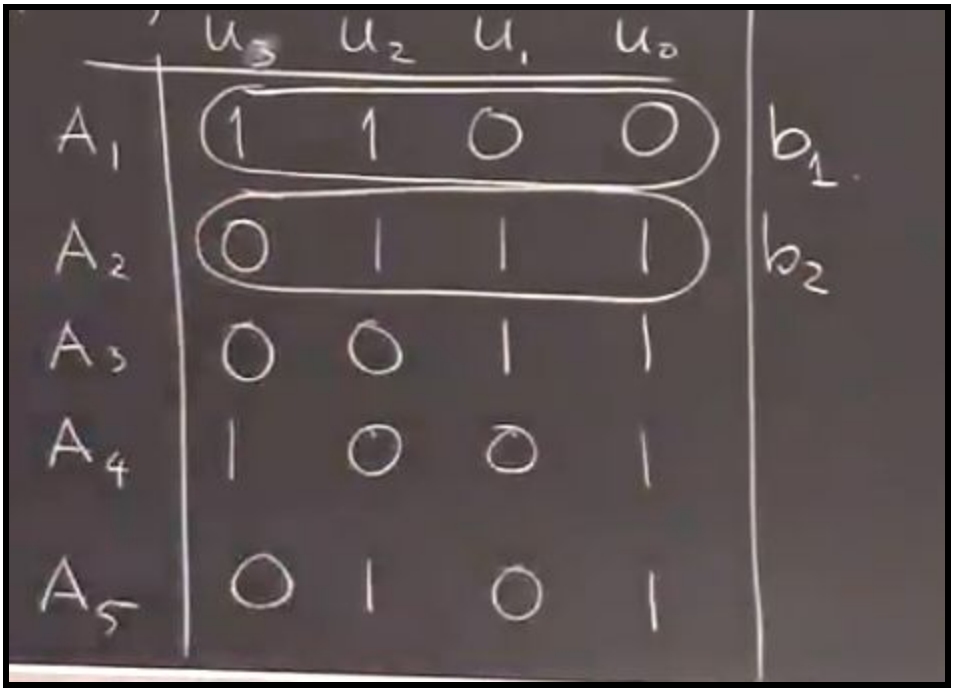
- Instance: $\langle A,k \rangle$, where A is a list of positive integers, I.e. $A = a_1, a_2, \ldots, a_n$ s.t. $a_i \in Z^+$, and k is a positive integer, $k \in Z^+$.
- Question: Is there a subsequence of A that adds up to k?
  I.e. of subsequence of A: $a_{i1}, a_{i2}, \ldots, a_{im}$ s.t. $1 \leq i1 \leq i2 \leq \ldots \leq im \leq m$.
  E.g. If A = 3, 2, 17, 11, 3, 9 and k = 15, then the answer is yes. (3+3+9 = 15) However, if k = 24, then the answer is no.
- **Theorem 11.1:** SUBSUM $\in$ NPC

  Proof:

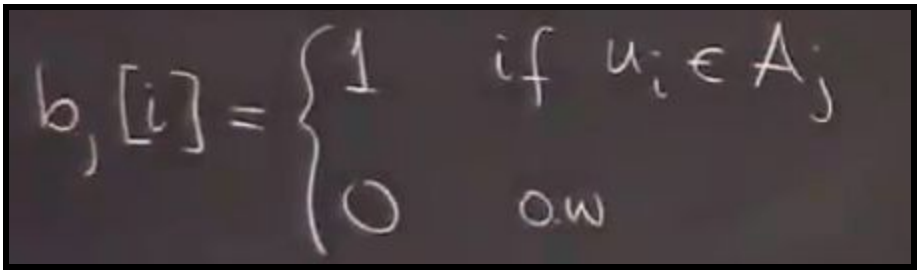
  a. **Proof that SUBSUM $\in$ NP:**
     The certificate here is the subsequence of A. The verifier simply checks that it is a subsequence of A and that the subsequence adds up to k.
  b. **Show that XCOV $\leq_p$ SUBSUM:**
     Given $\langle U,C \rangle$, which is an instance of XCOV, construct in polytime $\langle A,k \rangle$, which is an instance of SUBSUM, s.t. $\langle U,C \rangle \in$ XCOV iff $\langle A,k \rangle \in$ SUBSUM.
     Let $U = \{u_0, u_1, \ldots, u_{n-1}\}$.
     Let $C = \{A_1, A_2, \ldots, A_m\}$ s.t. $A_j \subseteq U$.

     Intuition: We will represent every element in set A_j as a binary number by viewing it in binary.
     For example:
     $U = \{u_0, u_1, u_2, u_3\}$
     $A_1 = \{u_2, u_3\}$
     $A_2 = \{u_0, u_1, u_2\}$
     $A_3 = \{u_0, u_1\}$
     $A_4 = \{u_0, u_3\}$
     $A_5 = \{u_0, u_2\}$
     We will create a matrix where every row corresponds to a set and every column corresponds to an element.
     If an element, y, is in a set, x, we put a 1 in (x,y).
     If an element, y, is not in a set, x, we put a 0 in (x,y).

This is how our matrix will look like:



Notice how A_1 has elements u_2 and u_3, but not u_0 or u_1.
Hence, the cell where u_3 and A_1 intersect has a 1.
Similarly, the cell where u_2 and A_1 intersect has a 1.
And the cell where u_1 and A_1 intersect and where u_0 and A_1 has a 0.
Each row of binary bits will be labelled b_i. For example, the first row is labelled b_1.
The below picture just references the fact that if an element is in a set, we put a 1 in the cell where they intersect, and if an element is not in a set, we put a 0 in the cell where they intersect.



Since we are looking for an exact cover, we are looking for sets s.t. when we add up their binary representations, we get a binary number s.t. all bits are 1'sr.
For example, take A_1 and A_3 from our example.
A_1 = 1100
A_3 = 0011
  1100
+0011
=1111 ← All the bits are 1

However, there's an issue with this, and the issue is carrying over 1's.
For example, take A_2, A_3, and A_5 from our example
A_2 = 0111
A_3 = 0011
A_5 = 0101

```
 0111
 0011
+0101
=1111
```

A_2, A_3, and A_5 aren't an exact cover because none of them contains U_4 and there are duplicates. However, when you add up their binary representations, you get 1111.
We can solve this by viewing b_i's in base (m + 1) because the only way to get a carry over is to sum m + 1 sets, but we only have m sets, so it's impossible to have a carry over.

c. **Claim: (U, C) has an exact cover iff there is a subsequence of b_1, ..., b_m, viewed as base (m + 1) numbers, that add up to 111...1 in base (m + 1).**

Note: 111...1 in base (m + 1) means that $k = \sum_{i=0}^{n-1} (m+1)^i = \frac{(m+1)^n - 1}{m}$.

Note: A subsequence of b_1, ..., b_m, viewed as base (m + 1) numbers means that $b_i = \sum_{j=0}^{n-1} (b_i[j])*(m+1)^j$.

Proof:
Suppose $\{A_{j1}, ..., A_{jk}\}$ is an exact cover of C.
$\leftrightarrow \forall i, 0 \le i \le n-1, \exists$ unique t, $1 \le t \le k$, s.t. $u_i \in A_{jt}$.
$\leftrightarrow \forall i, 0 \le i \le n-1, \exists$ unique t, $1 \le t \le k$, s.t. $b_{jt}[i] = 1$.
$\leftrightarrow$ the sum of b_{j1}, ..., b_{jk} = 111…1 (as base (m + 1) numbers).

This is polynomial as we just need to construct a (m × n) matrix.

Partition (PART):

- Instance: ⟨A⟩, where A is a list of positive integers, I.e. A = $a_1, a_2, \ldots, a_n$ s.t. $a_i \in Z^+$.
- Question: Is there a subsequence of A that adds up to half of the total sum of A?

  I.e. Is there a subsequence of A s.t. their sum = $\frac{\sum_{i=1}^{n} ai}{2}$?
- **Theorem 11.2:** PART ∈ NPC

  Proof:

  a. **Proof that PART ∈ NP:**
  The certificate is a sequence of numbers and the verifier verifies that this sequence is a subsequence of A and that their sum is equal to half the sum of the elements in A.

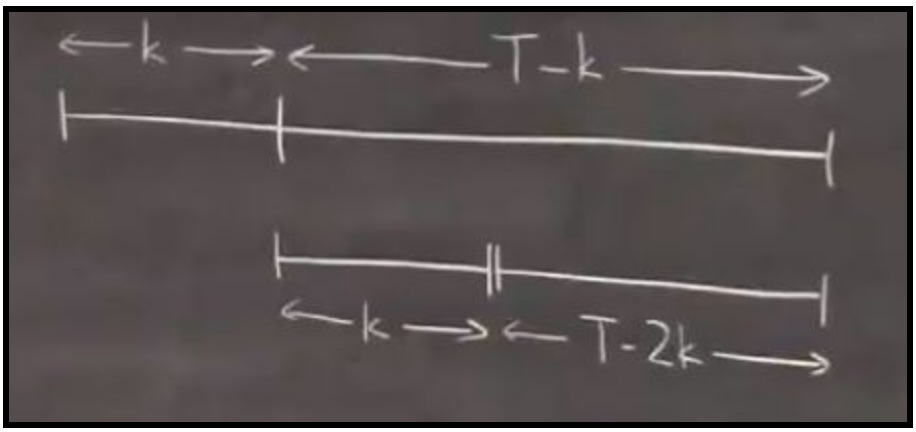
  b. **Show that SUBSUM ≤$_p$ PART:**
  Given ⟨A, k⟩, which is an instance of SUBSUM, construct in polynomial time ⟨A'⟩ s.t. ⟨A, k⟩ ∈ SUBSUM iff ⟨A'⟩ ∈ PART.

  Intuition:

  Let T = $\sum_{i=1}^{n} a_i$.

  We can add some of the elements in A and get k.
  If we take out k from our original sequence we're left with T−k. We can use that as our first half.
  If we take k from (T-k), we're left with T-2k.
  (T-k) and k add up to A.
  Now, if we add (T-2k), we can partition A into 2 halves:
  1. T-k
  2. k and (T-2k)

  Here's a picture.



  However, this picture is misleading, as it's possible that T − 2k is negative but our sequence must use a positive number. In that case, use 2k − T.
  Case 1. T − 2k < 0 → T − k ≤ k
  Here we use 2k − T.

Case 2. T − 2k = 0 → k = T/2
Then we are done, this is exactly what we are looking for.

c. Claim: ⟨A, k⟩ ∈ SUBSUM iff ⟨A'⟩ ∈ PART.

$$\text{From } A = a_1, ..., a_n \text{ and } k$$

$$A' = \begin{cases} A & \text{if } T = \sum_{i=1}^{n} a_i = 2k \\ a_1, ..., a_n, T - 2k & \text{if } T - 2k > 0 \\ a_1, ..., a_n, 2k - T & \text{if } T - k < 0 \end{cases}$$

The construction of A' from A, k is polynomial time.

Linear Programming (LP):

- An optimization Problem.
- Max/min a linear function subject to linear inequality constraints.
- For example to minimize the cost of a diet, we could just buy nothing but then we would starve. Here, the constraint is that we have sufficient nutrients in our diet.
- For example, we want to maximize profit from productive activities such that resource constraints are not exceeded.
- Example: Minimize $c_1x_1 + \ldots + c_nx_n$ where x_i are variables and c_i are constants in Z s.t.
  $a_{11}x_1 + \ldots + a_{1n}x_1 \geq b_1$
  .
  .
  .
  $a_{m1}x_1 + \ldots + a_{mn}x_m \geq b_m$
  This can be written simply as $A\bar{x} \leq \bar{b}$.
- There is a polynomial time algorithm for LP if $\bar{x} \in Q^n$.
  However, if we insist that the solution, $\bar{x}$, be integers, then it is NPC.
- The decision version of the linear programming questions works as follows:
  "Given A and $\bar{b}$, is there a vector $\bar{x}$ s.t. $A\bar{x} \leq \bar{b}$?"

Integer Linear Programming (ILP):

- Instance: ⟨A, B⟩ where A = m × n matrix of Z and b = m-vector of Z.
- Question: Is there an $\bar{x}$ s.t. $A\bar{x} \leq \bar{b}$ [$\bar{x}$ is a n-vector]

Zero or One Equations (ZOE):

- Instance: ⟨A⟩ where A = 0/1 m × n matrix.
- Question: Is there an $\vec{x}$ s.t. $A\vec{x} \leq \begin{pmatrix} 1 \\ . \\ . \\ 1 \end{pmatrix}$? [$\vec{x}$ is a n x 1 vector]
- **Theorem 11.3:** ZOE ∈ NPC

Proof:

a. **Proof that ZOE ∈ NP:**
The certificate is a 0/1 vector of length n and the verifier verifies that the dot product of any row in matrix A with $\vec{x}$ equals to 1.

b. **Show that XCOV ≤$_p$ ZOE:**
Given ⟨U, C⟩, which is an instance of XCOV, construct in polynomial time ⟨A⟩, which is an instance of ZOE, s.t. ⟨U, C⟩ ∈ XCOV iff ⟨A⟩ ∈ ZOE.
Let U = {u_1, … ,u_n}.
Let C = {A_1, … ,A_m} $A_j \subseteq$ U.

$$A\vec{x} = \begin{pmatrix} a_{11} & \dots & \dots & a_{1m} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & \dots & a_{nm} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_m \end{pmatrix}$$

$$a_{ij} = \begin{cases} 1 & \text{if } u_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

Columns represent sets.
Rows represent elements.
1 means that the set has the element.
Each x_i in $\vec{x}$ is either 0 or 1.
x_i = 1 means that we are selecting that set for our exact cover.
x_i = 0 means that we are not selecting that set for our exact cover.
Each $\vec{x} \in \{0, 1\}^m$ defines a subset of C.
$C_{\vec{x}} = \{A_{ji} \mid x_j = 1\}$.
Consider the dot product of (The i-th row of A) and $\vec{x}$, denoted as (The i-th row of A)*$\vec{x}$.
(The i-th row of A)*$\vec{x}$ = $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{im}x_m$

$$a_{ij}x_j = \begin{cases} 0 & \text{if we chose } a_{ij} \text{ but its not in } A_j \text{ or if we don't choose } a_{ij} \\ 1 & \text{if we choose } a_{ij} \text{ and its in } A_j \end{cases}$$

(The i-th row of A)* $\vec{x}$ is the number of times u, appears in sets in $C_{\vec{x}}$.

We want A$\vec{x}$ to be $\begin{pmatrix} 1 \\ . \\ . \\ 1 \end{pmatrix}$ of size n × 1.

c. Claim: ⟨U, C⟩ ∈ XCOV iff ⟨A⟩ ∈ ZOE

If ⟨U, C⟩ has an exact cover C', then $\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_m \end{pmatrix}$ where

$$x_j = \begin{cases} 1 & \text{if } A_j \in C' \\ 0 & otherwise \end{cases}$$ satisfies $$A\vec{x} = \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}$$.

Conversely, if $\vec{x}$ satisfies $$A\vec{x} = \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}$$ then $C_{\vec{x}} = \{A_j \mid x_j = 1\}$ is an exact cover.

Therefore, ZOE ∈ NPC.

Zero One Linear Programming (ZOLP):

- Instance: ⟨A, b⟩ where A is a m × n matrix and b is a m-vector of Z.
- Question: Is there an $\bar{x}$ s.t. $A\bar{x} \leq \bar{b}$? [$\bar{x}$ is a n-vector ∈ $\{0, 1\}^n$]
- **Note:** To turn the equality $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = 1$, we simply do $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq 1$ and $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq 1$. However, since we want everything to be in the form of $A\bar{x} \leq \bar{b}$, we multiple the second inequality by (-1). Hence, we get:
  1. $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq 1$
  2. $-a_{11}x_1 - a_{12}x_2 - \ldots - a_{1n}x_n \leq -1$

  These 2 inequalities, together, are equivalent to $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = 1$.
- **Theorem 11.4:** ZOLP ∈ NPC

Proof:

a. **Proof that ZOLP ∈ NP**
The certificate is the vector. The verifier just multiplies the matrix with the vector and makes sure that the dot product of the i^{th} row of the matrix with the i^{th} element of the vector is less than or equal to b_i.

b. **Show that ZOE ≤$_P$ ZOLP**
Given ⟨A⟩, which is an instance of ZOE, construct in polynomial time ⟨A'⟩, which is an instance of ZOLP s.t. there exists an $\bar{x}$ where $A\bar{x} = \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}$ iff there exists an $\bar{y}$ where $A'\bar{y} \leq \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}$.

As shown above, $A' = \begin{pmatrix} A \\ -A \end{pmatrix}$.

Note: Since A is a m x n matrix and (-A) is also a m x n matrix, A' is a (2m) x n matrix and b is a (2m) vector where the first half of it is 1's and the second half of it is (-1)'s.

- **Corollary 11.5:** ILP ∈ NPC
  Note: ZOLP $\leq_P$ ILP
  Proof:
  a. **Show that VC $\leq_P$ ZOLP:**
     Given ⟨G, k⟩, which is an instance of VC, construct in polynomial time, ⟨A, $\bar{b}$⟩, which is an instance of ZOLP s.t. G has VC of size k iff there exists a $\bar{x}$ s.t. $A\bar{x} \leq \bar{b}$.
     Note: G = (V,E) is an undirected graph.
     Let V = {u_1, u_2, …, u_n}.
     We will introduce variables x_i, ... , x_n ∈ {0, 1}.
     Each $\bar{x}$ = (x_i, …, x_n) corresponds to a V' ⊆ V.
     We do this by having u_i ∈ V' iff x_i = 1.
     $x_1 + x_2 + \ldots + x_n \leq k$ (|V'| ≤ k)
     To show that V' covers all of the edges, we do $x_i + x_j \geq 1$ ∀{u_i, u_j} ∈ E.
     However, since we want ≤, we will turn $x_i + x_j \geq 1$ into $-x_i - x_j \leq -1$.

$$A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & -1 & -1 & 0 \end{pmatrix}$$

     A has all 1's on the first row.
     Every row afterwards has all 0's except for -1's in the i-th and j-th column.

$$A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} \leq \begin{pmatrix} k \\ -1 \\ \vdots \\ -1 \end{pmatrix}$$

3DM:

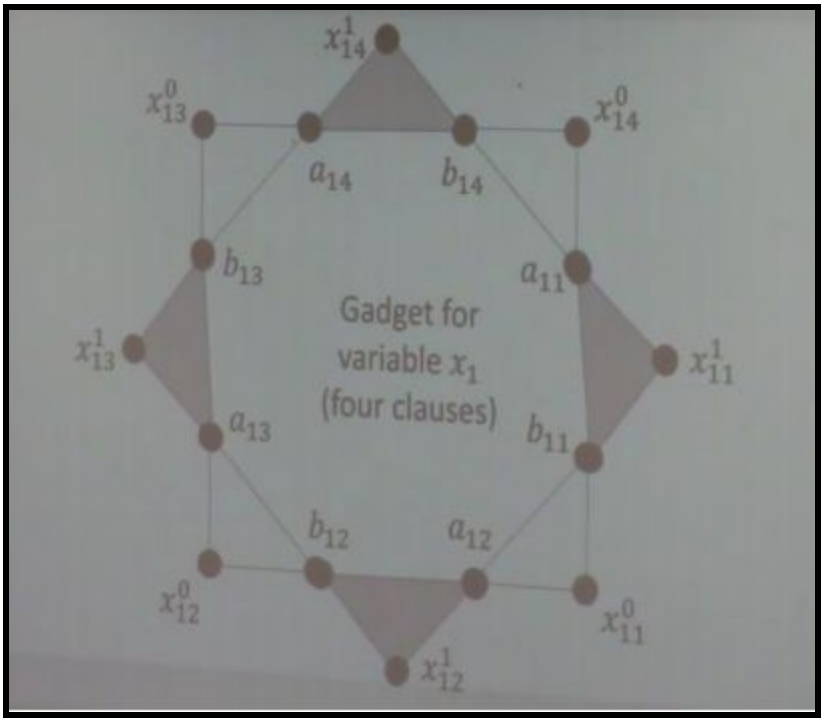
- Suppose we have a coding competition with the following rule:
  - Each team must have exactly 3 people which consists of one 2nd year student, one 3rd year student and one 4th year student.
- Suppose there are n 2nd year students, n 3rd year students and n 4th year students.
- We know that certain triples are compatible, meaning that the people in that triple get along, and certain triples are not compatible, meaning that the people in that triple don't get along.
- Can we make n teams where every team is disjoint and every team consists of compatible teams from one person in each year?
- This is an example of a problem known as **3 dimensional matching**.

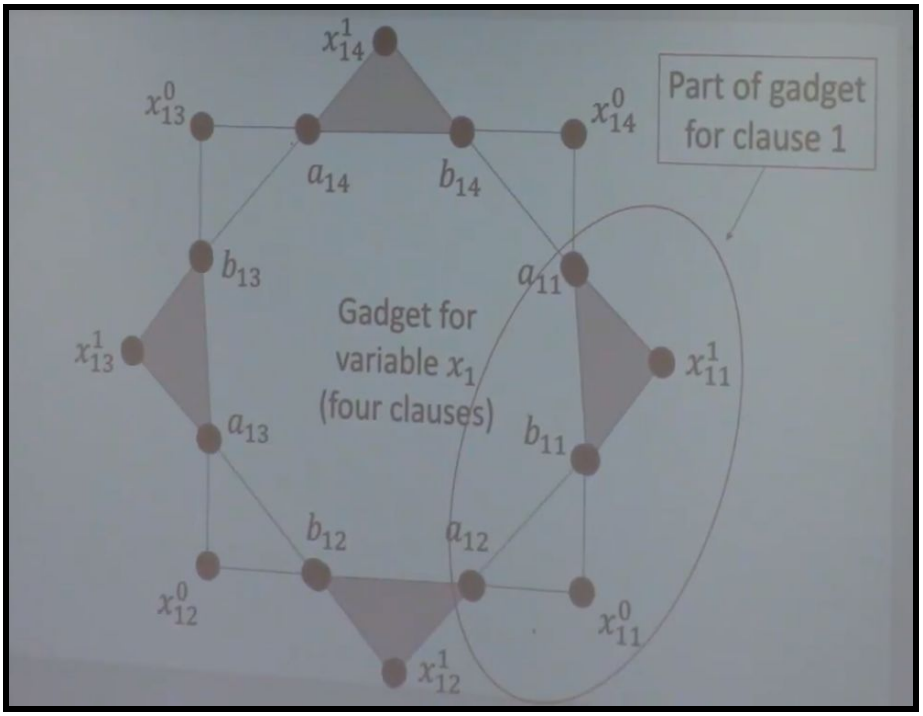
- Instance: ⟨A, B, C, M⟩ where A, b, C are disjoint sets of n elements each and $M \subseteq A \times B \times C$ (M is the compatible triples.)
- Question: Is there $M' \subseteq M$ s.t. $|M'| = n$ and $\forall$ disjoint sets (a, b, c), (a', b', c') $\in M'$, $a \neq a'$,
  $b \neq b', c \neq c'$?
  I.e. Is there a subset of M, M', s.t. there n triples/teams and the teams must consist of distinct people. **Note:** M' is called a matching.
- **Theorem 11.6:** 3DM $\in$ NPC

Proof:

a. **3DM $\in$ NP**
b. **Show that 3SAT $\leq_p$ 3DM**
   Given a 3-CNF formula F, construct in polynomial time (A, B, C, M) s.t.
   F is satisfiable iff $M' \subseteq M$ is a matching.
   F's variables are $x_1, \ldots, x_n$.
   $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
   $Cj = l_j^1 \vee l_j^2 \vee l_j^3$

   Group I triples:
   - For every variable xi we will construct a gadget like shown below.





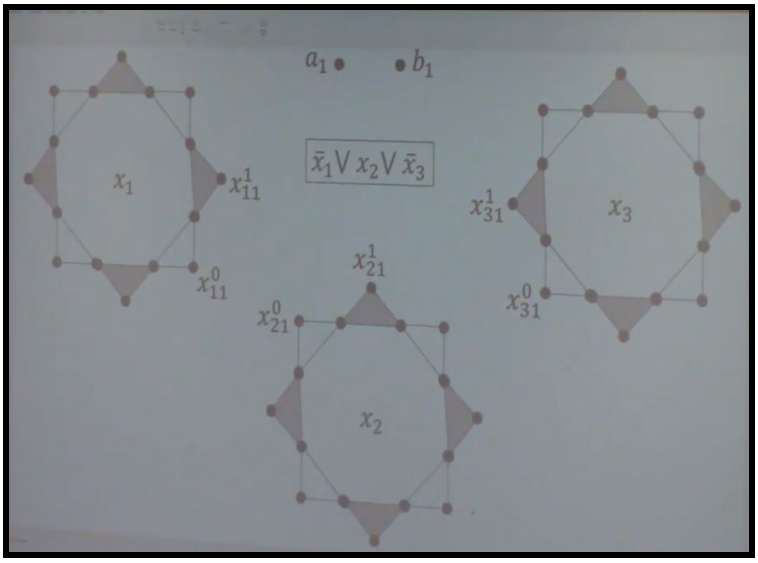
This is an example of a gadget with 4 clauses.

- For every variable and every clause, we will have 2 triples, which will be interconnected in the above pattern.
- We label x_{ij}^a for each item where a means positive or negative. If a = 0, then x_{ij}^0 is negative and if a = 1, then x_{ij}^1 is positive. Furthermore, i means that this is the i^{th} variable and j means that this is the j^{th} clause.
- The elements aij and bij they are constructed as follows:
  - a_{ij} is connected to x_{ij}^1 and b_{ij}.
  - b_{ij} is connected to x_{ij}^0 and a_{ij+1}.
- These interconnections are done in such a way that if we select the shaded triple we can not select the unshaded triple that follows.
  - If we select the triple containing x_{11}^1 then we cannot also select the triple containing x_{11}^0 because they share b_{11}.
  - If we choose one shaded triple, they need to be all shaded.
  - If we choose one clear triple, they need to be all clear.
  - This is what we want, since x_{11} can only be one truth value.
- Here's how we create the triples more precisely:
  ∀i, j where $1 \le i \le n$ and $1 \le j \le m$, we have $(a_{ij}, b_{ij}, x_{ij}^1)$ and $(a_{ij\oplus 1}, b_{ij}, x_{ij}^0)$, where $j \oplus 1 = \begin{cases} j+1 & \text{if } j < m \\ 1 & \text{if } j = m \end{cases}$.
- We have 2mn of these triples.

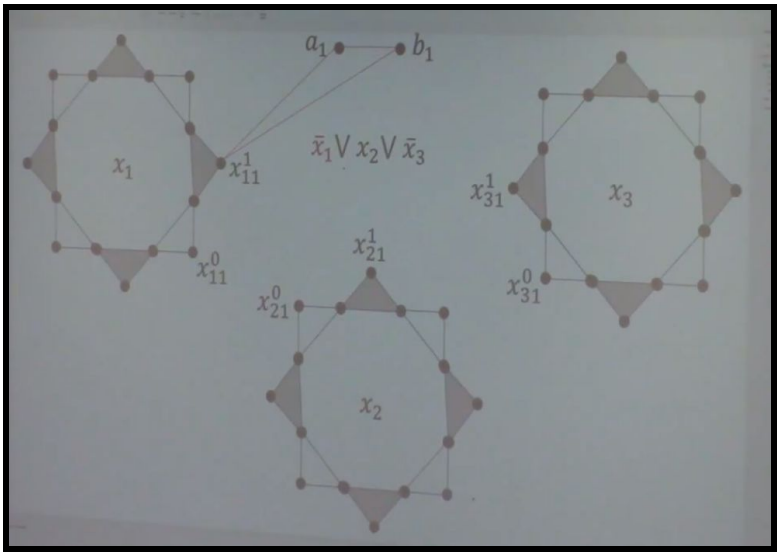
Group 2 triples:

- Let $C_j = l_j^1 \vee l_j^2 \vee l_j^3$.
- We will make one triple per literal, 3 for the entire clause.
- Take l_j^t, where t is 1, 2, or 3. There are 2 possibilities for it.
    1. l_j^t is a positive literal.
       I.e. $l_j^t = x_i$
       Hence, we create a triple (a_j, b_j, x_{ij}^0) where $a_j \in A$ and $b_j \in B$.
    2. l_j^t is a negative literal.
    3. I.e. $l_j^t = \neg x_i$
       Hence, we get (a_j, b_j, x_{ij}^1) where $a_j \in A$ and $b_j \in B$.

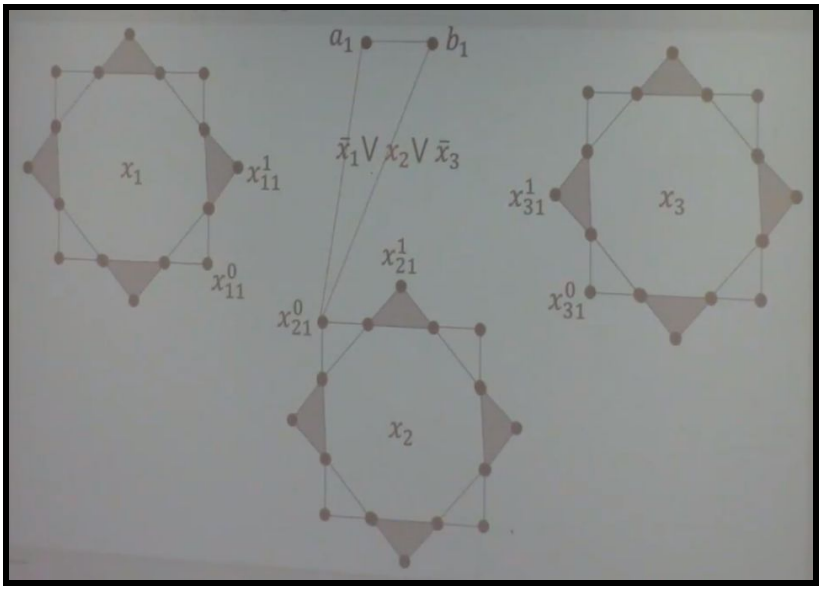
    We take the x_{ij} that's opposite to our x_i so that we take the variable that was not selected in our previous gadget.
    Here's a picture.



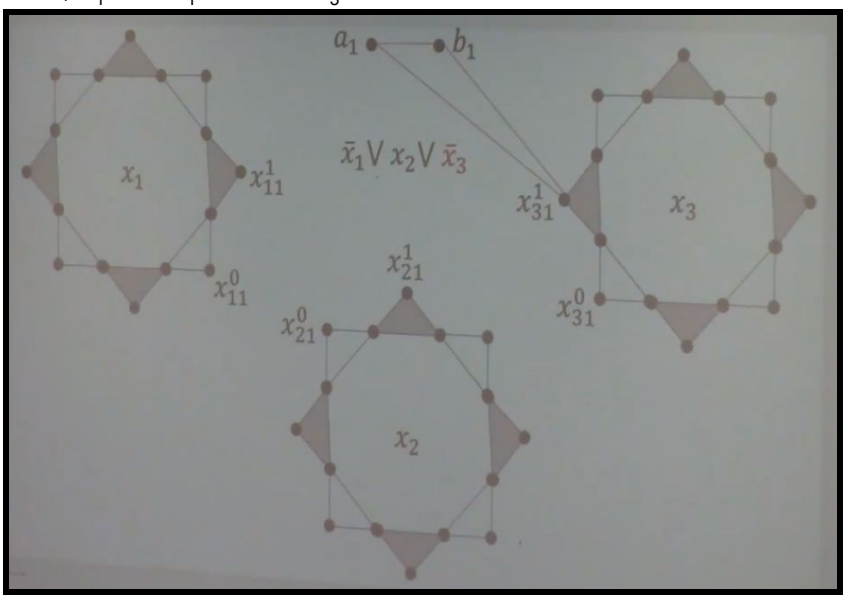
Here, a_1 and b_1 are in $\neg x_1$.



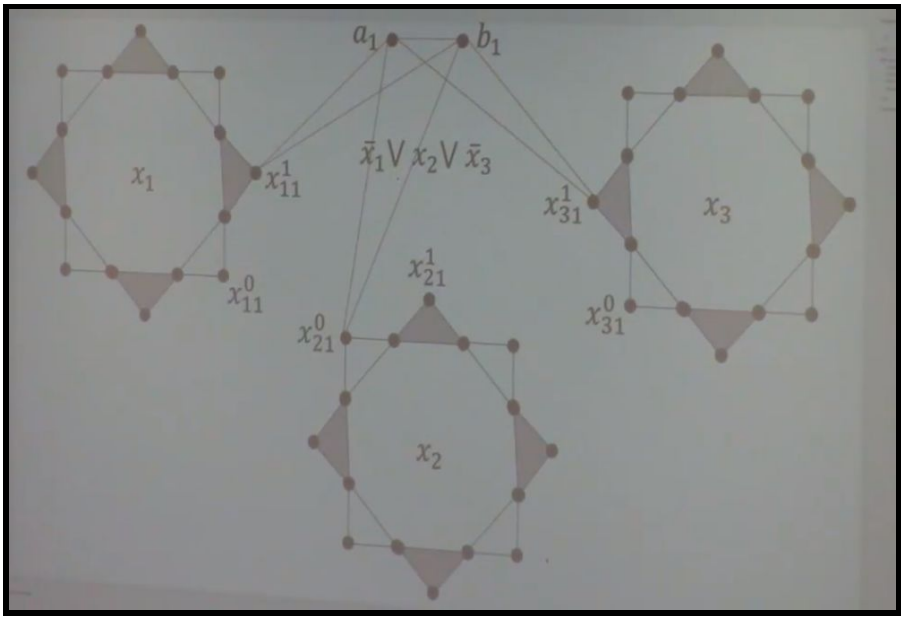
Here, a_1 and b_1 are in x_2.



Here, a_1 and b_1 are in $\neg x_3$.



The whole thing looks like this



- There are 3m such triples here. There are m clauses and each clause has 3 triples.

Group 3 Triples:

- Recall the 2mn triples from group 1.
  x_{ij}^0 and x_{ij}^1 are the tips of the triangles in group 1.
  Half of the 2mn triples (mn triples) will be covered by choosing whether or not to use x_{ij}^0 or x_{ij}^1.
  Another m triples will be covered because we have chosen certain triples, one per clause, to indicate which literal of that clause will make that whole clause true.
  2mn - mn - m = m(n-1). This means that there are m(n-1) triples that have not been covered yet. We cover them here.
- We will introduce new variables ($\sim a_k$, $\sim b_k$, x_{ij}^0) and ($\sim a_k$, $\sim b_k$, x_{ij}^1), where $1 \leq k \leq m(n-1)$. In total, we have $2m^2n(n-1)$ triples.

In total, we define:

- $A = \{a_{ij} \mid 1 \leq i \leq n\} \cup \{a_j \mid 1 \leq j \leq m\} \cup \{\sim a_k \mid 1 \leq k \leq m(n-1)\}$
- $B = \{b_{ij} \mid 1 \leq i \leq n\} \cup \{b_j \mid 1 \leq j \leq m\} \cup \{\sim b_k \mid 1 \leq k \leq m(n-1)\}$
- $C = \{c_{ij} \mid 1 \leq i \leq n\} \cup \{c_j \mid 1 \leq j \leq m\} \cup \{\sim c_k \mid 1 \leq k \leq m(n-1)\}$
- Now we need to confirm that A, B, C are all of the same size.
  $mn + m + m(n-1) = 2mn = |A| = |B| = |C|$
- # of triples:
  Group I = 2mn
  Group II = 3m
  Group III = $2m^2n(n-1) = O(m^2n^2)$.
  In total, we have $O(m^2n^2)$.

d. F is satisfiable iff (A, B, C, M) has a matching.

Sketch of Proof:
(=>)
Let τ satisfy F.
Pick triples in M' as follows:
If τ(xi) = 1, then pick the triple $(-, -, x_{ij}^{1})$ from Group 1.
If τ(xi) = 0 then pick the triple $(-, -, x_{ij}^{0})$ from Group 1.

For each C_j pick l_j^{tj} s.t. $\tau(l_j^{tj}) = 1$.
If $\tau(l_j^{tj}) = x_i$ then pick the triple $(-, -, x_{ij}^{0})$ from Group 2.
If $\tau(l_j^{tj}) = \neg x_i$ then pick the triple $(-, -, x_{ij}^{1})$ from Group 2.

For each x_{ij}^{0} or x_{ij}^{1}, not yet covered, choose the triple $(\sim a_k, \sim b_k, x_{ij}^{0/1})$ from Group 3 to cover it.

(<=)
Left it as exercise and gave the following hints:

1. Let M' be a matching.
2. $$\text{Define } \tau(x_i) = \begin{cases} 1 & \text{if } M' \text{ includes } (-, -, x_{ij}^{1}) \\ 0 & \text{if } M' \text{ includes } (-, -, x_{ij}^{0}) \end{cases}$$

<u>3XCOV Exact cover by 3-sets [sets of size 3]:</u>

- **Theorem 11.7:** 3XCOV ∈ NPC

Proof:
Left as exercise with the hint: show that $3DM \leq_p 3XCOV$.