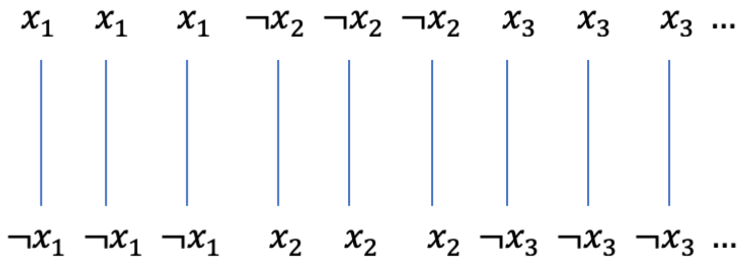
Collaborators: yc2454 (Yalu Cai)

To prove MAX-CUT is NP-complete, we need to prove it's both NP and NP-hard. It's NP because it's easy to verify in polynomial time whether a potential solution is correct because we just need to see whether the cut goes through k edges that have one endpoint in S and the other in $\bar{S}$.
To prove it's NP-hard, we will reduce from a $\neq$-SAT problem ϕ as per the hint. This reduction is polynomial time because we're constructing $O(n)$ nodes and $O(l^2n)$ edges.

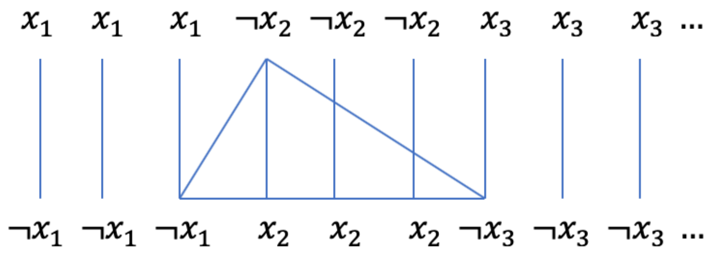
Claim: There is a cut of size $n(3l)^2 + 2l$ on the transformed graph if and only if ϕ is $\neq$-satisfiable.
Proof:

Claim: If ϕ is $\neq$-satisfiable, then there exists a cut of size $n(3l)^2 + 2l$ on the transformed graph.
Proof: Given the assignment, visualize all literals that are set to true to be lined up horizontally on the upper side of the page and all their negation nodes to be lined up horizontally on the lower side of the page as such:



x_1 x_1 x_1 $\neg x_2$ $\neg x_2$ $\neg x_2$ x_3 x_3 x_3 ...
$\neg x_1$ $\neg x_1$ $\neg x_1$ x_2 x_2 x_2 $\neg x_3$ $\neg x_3$ $\neg x_3$ ...

Put all top nodes in S and all bottom nodes in $\bar{S}$. There are thus $n(3l)^2$ edges between all the x_i's and $\neg x_i$'s. The satisfying $\neq$-assignment makes it so that one or more but less than three literals in each clause is true. The resulting triangle thus has nodes on both sides, as such:



x_1 x_1 x_1 $\neg x_2$ $\neg x_2$ $\neg x_2$ x_3 x_3 x_3 ...
$\neg x_1$ $\neg x_1$ $\neg x_1$ x_2 x_2 x_2 $\neg x_3$ $\neg x_3$ $\neg x_3$ ...

A cut for a satisfying $\neq$-assignment thus has the size $n(3l)^2 + 2l$. The $n(3l)^2$ is because of the edges between x_i's and $\neg x_i$'s, and the $2l$ is because each satisfied clause's triangle would get cut twice.

Claim: If there exists a cut of size $n(3l)^2 + 2l$, then ϕ is $\neq$-satisfiable.
Proof: Realize that the $n(3l)^2$ are all from the edges between x_i and $\neg x_i$ being cut. The remaining $2l$ edges cut are from the edges of clause triangles. Assign all literal nodes in S to true and all literal nodes in $\bar{S}$ to false (or vice-versa; it doesn't matter because we proved in Q2a that the negation of a valid $\neq$-assignment is still a valid $\neq$-assignment).