BILL AND NATHAN, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!!
**Def of LAPX**

**Def** Let $A$ be a min problem. $A \in \text{LAPX}$ if $\exists c$ and alg $M$: $M(x) \leq (c \log |x|)\text{OPT}(x)$. 
Def of LAPX

Def Let $A$ be a min problem. $A \in \text{LAPX}$ if $\exists$ $c$ and alg $M$: $M(x) \leq (c \log |x|) \text{OPT}(x)$.

We want to show some problems cannot be approximated any better than LAPX.
Def of LAPX

**Def** Let $A$ be a min problem. $A \in \text{LAPX}$ if $\exists$ $c$ and alg $M$: $M(x) \leq (c \log |x|) \text{OPT}(x)$.

We want to show some problems cannot be approximated any better than LAPX.

Need to define what we mean.
Def of LPTAS and LAPX

**Def** A is in **LPTAS** if there is an alg that, on input $(x, \epsilon)$ outputs $x$ such that $M(x) \leq (\epsilon \log |x|)OPT(x)$.
Def of LPTAS and LAPX

Def $A$ is in **LPTAS** if there is an alg that, on input $(x, \epsilon)$ outputs $x$ such that $M(x) \leq (\epsilon \log |x|)\text{OPT}(x)$.

Recall

When we wanted to show some problems did not have a PTAS we first needed one problem that did not have a PTAS: MAX3SAT. We then used reductions.
Def A is in LPTAS if there is an alg that, on input $(x, \epsilon)$ outputs $x$ such that $M(x) \leq (\epsilon \log |x|)\text{OPT}(x)$.

Recall
When we wanted to show some problems did not have a PTAS we first needed one problem that did not have a PTAS: MAX3SAT. We then used reductions.

Now We want to show some problems do not have LPTAS. We first need one problem that does not have an LPTAS: SETCOVER.
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$’s that cover $\{1, \ldots, n\}$.
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$'s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \text{OPTIMAL}$. 

2. Dinur and Steurer in 2013 showed that, assuming $P \neq \text{NP}$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for **Set Cover** (when $m \sim n^{0.8}$).

   (1) Bound is surprisingly tight. Not $\Theta(\log n)$, Actually $\ln n$.
   (2) Dinur and Steurer showed SETCOVER is not LPTAS (earlier results did that).
   (2) Assuming this result we obtain other problems not in LPTAS.

(4) We define \text{LAPX-}complete with this in mind.
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$’s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \text{OPTIMAL}$.

2. Dinur and Steurer in 2013 showed that, assuming $P \neq NP$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for **Set Cover** (When $m \sim n^{0.8}$.)
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$’s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \text{OPTIMAL}$.

2. Dinur and Steurer in 2013 showed that, assuming $P \neq NP$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for **Set Cover** (When $m \sim n^{0.8}$.)

(1) Bound is surprisingly tight. Not $\Theta(\log n)$, Actually $\ln n$. 
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$’s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \text{OPTIMAL}$.

2. Dinur and Steurer in 2013 showed that, assuming $P \neq NP$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for **Set Cover** (When $m \sim n^{0.8}$.)

(1) Bound is surprisingly tight. Not $\Theta(\log n)$, Actually $\ln n$.
(2) Dinur and Steurer showed SETCOVER is not LPTAS (earlier results did that).
Approximating Set Cover

Set Cover Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$'s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for Set Cover that will return $(\ln n) \times \text{OPTIMAL}$.

2. Dinur and Steurer in 2013 showed that, assuming $P \neq NP$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for Set Cover (When $m \sim n^{0.8}$.)

(1) Bound is surprisingly tight. Not $\Theta(\log n)$, Actually $\ln n$.
(2) Dinur and Steurer showed SETCOVER is not LPTAS (earlier results did that).
(2) Assuming this result we obtain other problems not in LPTAS.
Approximating Set Cover

**Set Cover** Given $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ find the least number of sets $S_i$’s that cover $\{1, \ldots, n\}$.

1. Chvatal in 1979 showed that there is a poly time approx algorithm for **Set Cover** that will return $(\ln n) \times \text{OPTIMAL}$.

2. Dinur and Steurer in 2013 showed that, assuming $P \neq NP$, for all $\epsilon$ there is no $(1 - \epsilon) \ln n \times \text{OPTIMAL}$ approx alg for **Set Cover** (When $m \sim n^{0.8}$).

1. Bound is surprisingly tight. Not $\Theta(\log n)$, Actually $\ln n$.
2. Dinur and Steurer showed SETCOVER is not LPTAS (earlier results did that).
3. Assuming this result we obtain other problems not in LPTAS.
4. We define LAPX-complete with this in mind.
Def of LAPX-Complete

**Def** A is LAPX-complete if

1. A is in LAPX.
2. SETCOVER ≤ A with an APR (Approximation-Preserving-Reduction).
Def of LAPX-Complete

**Def** A is LAPX-complete if

1. A is in LAPX.
Def of LAPX-Complete

Def $A$ is LAPX-complete if

1. $A$ is in LAPX.
2. $\text{SETCOVER} \leq A$ with an APR (Approximation-Preserving-Reduction).
DOM is LAPX-Complete
DOM is in LAPX

A greedy algorithm where you always take the vertex of max degree yields a $(\ln \Delta + 2)$-approximation.
DOM is in LAPX

DOM is in LAPX A greedy algorithm where you always take the vertex of max degree yields a $(\ln \Delta + 2)$-approximation.

Next slide we show we show $\text{SETCOVER} \leq \text{DOM}$. 
SETCOVER $\leq$ DOM

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.

2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$'s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.

Let $U = \{1, \ldots, n\}$ and $S = \{S_1, \ldots, S_m\}$.

Let $D$ be a dominating set for $G$.

If there are any $U$-vertices in $D$ then replace them by the $S$-vertex they connect to. Can assume that every dominating set consists only of $S$-vertices.

We map a dominating set to the $S$-sets that its $S$-vertices correspond to. The size of the dominating set is exactly the size of a covering.
SETCOVER $\leq$ DOM

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$. 
SETCOVER $\leq$ DOM

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.
2. Form a graph $G = (V, E)$ as follows:
1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.
2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$. 
1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.
2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$'s is an edge.
SETCOVER ≤ DOM

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.

2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$'s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.
1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.

2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$’s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.

Let $U = \{1, \ldots, n\}$ and $S = \{S_1, \ldots, S_m\}$.
SETCOVER $\leq$ DOM

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.
2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$'s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.
Let $U = \{1, \ldots, n\}$ and $S = \{S_1, \ldots, S_m\}$. Let $D$ be a dominating set for $G$. 
SETCOVER \leq \text{DOM}

1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.

2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$'s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.

Let $U = \{1, \ldots, n\}$ and $S = \{S_1, \ldots, S_m\}$.

Let $D$ be a dominating set for $G$.

If there are any $U$-vertices in $D$ then replace them by the $S$-vertex they connect to. Can assume that every dominating set consists only of $S$-vertices.
1. Input $n$ and $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$. Let $U = \{1, \ldots, n\}$.

2. Form a graph $G = (V, E)$ as follows:
   (1) $V = \{1, \ldots, n\} \cup \{S_1, \ldots, S_m\}$.
   (2) Between every two $S_i$’s is an edge.
   (3) If $i \in S_j$ then have edge $(i, S_j)$.

Let $U = \{1, \ldots, n\}$ and $S = \{S_1, \ldots, S_m\}$.
Let $D$ be a dominating set for $G$.

If there are any $U$-vertices in $D$ then replace them by the $S$-vertex they connect to. Can assume that every dominating set consists only of $S$-vertices.

We map a dominating set to the $S$-sets that its $S$-vertices correspond to. The size of the dominating set is exactly the size of a covering.
List of LAPX-Complete Problems
About the Reductions

We will list several problems that are LAPX-complete.
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions.
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions. For all of the problem we present the reduction was from SETCOVER.
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions. For all of the problem we present the reduction was from SETCOVER.

A contrast:
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions. For all of the problem we present the reduction was from SETCOVER.

A contrast:
- For NPC we often use problems other than SAT.
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions. For all of the problem we present the reduction was from SETCOVER.

A contrast:

▶ For NPC we often use problems other than SAT.
▶ For APX-complete we often use problems other than MAX3SAT.
About the Reductions

We will list several problems that are LAPX-complete. We omit both the algorithms and the reductions. For all of the problem we present the reduction was from SETCOVER.

A contrast:

- For NPC we often use problems other than SAT.
- For APX-complete we often use problems other than MAX3SAT.
- For LAPX-complete we seem to only use SETCOVER. This may be because there are far fewer LAPX problems.
Def The **Motion Planning Problem** is as follows.

**Input** A graph $G = (V, E)$ with the vertex set split into two (possibly overlapping) sets $V_1, V_2$ of the same size. The set $V_1$ are called *tokens* and each one has a *robot* on it. A *move* is when a robot goes on a path with no other robots on it. Note that a robot may go quite far in one move.
The **Motion Planning Problem** is as follows.

**Input** A graph $G = (V, E)$ with the vertex set split into two (possibly overlapping) sets $V_1, V_2$ of the same size. The set $V_1$ are called *tokens* and each one has a *robot* on it. A **move** is when a robot goes on a path with no other robots on it. Note that a robot may go quite far in one move.

**Question** We want a final configuration where all the robots are on the vertices in $V_2$ (only one robot can fit on a vertex). We want to do this in the minimum number of moves. What is that min?
Node Weighted Steiner Tree

**Input** Graph $G = (V, E)$ with weights on its nodes, a terminal nodes $T \subseteq V$, and a node $r \in V$. 

*Question* We want a set of nodes $S$ such that

1. The graph induced by $T \cup S$ connects all terminal nodes to $r$,
2. the sum of the weights in $S$ is minimal over all such $S$. 

Input  Graph $G = (V, E)$ with weights on its nodes, a terminal nodes $T \subseteq V$, and a node $r \in V$.

Question  We want a set of nodes $S$ of such that

1. The graph induced by $T \cup S$ connects all terminal nodes to $r$,
2. the sum of the weights in $S$ is minimal over all such $S$. 

**Node Weighted Steiner Tree**

**Input** Graph $G = (V, E)$ with weights on its nodes, a terminal nodes $T \subseteq V$, and a node $r \in V$.

**Question** We want a set of nodes $S$ of such that

1. The graph induced by $T \cup S$ connects all terminal nodes to $r$,
Node Weighted Steiner Tree

Input: Graph $G = (V, E)$ with weights on its nodes, a terminal nodes $T \subseteq V$, and a node $r \in V$.

Question: We want a set of nodes $S$ of such that

1. The graph induced by $T \cup S$ connects all terminal nodes to $r$,
2. the sum of the weights in $S$ is minimal over all such $S$. 
Group Steiner Tree

Input A graph $G = (V, E)$ with weights on its edges, sets $V_1, \ldots, V_k \subseteq V$, and a node $r \in V$. 

Group Steiner Tree

**Input** A graph $G = (V, E)$ with weights on its edges, sets $V_1, \ldots, V_k \subseteq V$, and a node $r \in V$.

**Question** We want a set of nodes $S$ of such that

1. Graph induced by $T \cup S$ connects some vertex of each $V_i$ to $r$,
2. sum of the weights in $T$ is minimal over all such $T$. 
Group Steiner Tree

**Input** A graph $G = (V, E)$ with weights on its edges, sets $V_1, \ldots, V_k \subseteq V$, and a node $r \in V$.

**Question** We want a set of nodes $S$ of such that

1. Graph induced by $T \cup S$ connects some vertex of each $V_i$ to $r$, 
Group Steiner Tree

**Input** A graph $G = (V, E)$ with weights on its edges, sets $V_1, \ldots, V_k \subseteq V$, and a node $r \in V$.

**Question** We want a set of nodes $S$ of such that

1. Graph induced by $T \cup S$ connects some vertex of each $V_i$ to $r$,
2. sum of the weights in $T$ is minimal over all such $T$. 