Lecture 11: Polynomial Hierarchy I

Arijit Bishnu

30.03.2010
Outline

1. Some New Problems and a New Class
2. Polynomial Hierarchy
Outline

1. Some New Problems and a New Class
2. Polynomial Hierarchy
A New Problem on the lines of INDSET

**INDSET**

\[ \text{INDSET} = \{ < G, k > \mid \text{Graph } G \text{ has an independent set of size } \geq k \} \]
A New Problem on the lines of INDSET

**INDSET**

\[
\text{INDSET} = \{ < G, k > \mid \text{Graph } G \text{ has an independent set of size } \geq k \}
\]

**EXACT INDSET**

\[
\text{EXACT INDSET} = \{ < G, k > \mid \text{Graph } G \text{ has the largest independent set of size exactly } k \}
\]
A New Problem on the lines of INDSET

INDSET

INDSET = \{ < G, k > \mid \text{Graph } G \text{ has an independent set of size } \geq k \} 

EXACT INDSET

EXACT INDSET = \{ < G, k > \mid \text{Graph } G \text{ has the largest independent set of size exactly } k \} 

Question
A New Problem on the lines of INDSET

**INDSET**

\[ \text{INDSET} = \{ < G, k > \mid \text{Graph } G \text{ has an independent set of size } \geq k \} \]

**EXACT INDSET**

\[ \text{EXACT INDSET} = \{ < G, k > \mid \text{Graph } G \text{ has the largest independent set of size exactly } k \} \]

**Question**

- INDSET admits a short certificate. But, what about EXACT INDSET?
A New Problem on the lines of INDSET

**INDSET**

INDSET = \{ < G, k > \mid \text{Graph } G \text{ has an independent set of size } \geq k \}

**EXACT INDSET**

EXACT INDSET = \{ < G, k > \mid \text{Graph } G \text{ has the largest independent set of size exactly } k \}

**Question**

- INDSET admits a short certificate. But, what about EXACT INDSET?
- \(< G, k > \in \text{EXACT INDSET}\) iff \(\exists\) an independent set of size \(k\) in \(G\) and all other independent sets have size at most \(k\).
A New Problem on the lines of VERTEX COVER

**VERTEX COVER**

VERTEX COVER = \{ < G, k > \mid \text{Graph G has a vertex cover of size } \leq k \}
A New Problem on the lines of VERTEX COVER

**VERTEX COVER**

\[ \text{VERTEX COVER} = \{ < G, k > \mid \text{Graph } G \text{ has a vertex cover of size } \leq k \} \]

**EXACT VERTEX COVER**

\[ \text{EXACT VERTEX COVER} = \{ < G, k > \mid \text{Graph } G \text{ has the minimum vertex cover of size exactly } k \} \]
A New Problem on the lines of VERTEX COVER

VERTEX COVER

\[ \text{VERTEX COVER} = \{ < G, k > \mid \text{Graph } G \text{ has a vertex cover of size } \leq k \} \]

EXACT VERTEX COVER

\[ \text{EXACT VERTEX COVER} = \{ < G, k > \mid \text{Graph } G \text{ has the minimum vertex cover of size exactly } k \} \]

Question
A New Problem on the lines of VERTEX COVER

<table>
<thead>
<tr>
<th>VERTEX COVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERTEX COVER = { &lt; G, k &gt; \mid \text{Graph } G \text{ has a vertex cover of size } \leq k }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXACT VERTEX COVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXACT VERTEX COVER = { &lt; G, k &gt; \mid \text{Graph } G \text{ has the minimum vertex cover of size exactly } k }</td>
</tr>
</tbody>
</table>

**Question**

- VERTEX COVER admits a **short certificate**. But, what about EXACT VERTEX COVER?
A New Problem on the lines of VERTEX COVER

**VERTEX COVER**

VERTEX COVER = \{ \langle G, k \rangle \mid \text{Graph } G \text{ has a vertex cover of size } \leq k \} 

**EXACT VERTEX COVER**

EXACT VERTEX COVER = \{ \langle G, k \rangle \mid \text{Graph } G \text{ has the minimum vertex cover of size exactly } k \} 

**Question**

- VERTEX COVER admits a short certificate. But, what about EXACT VERTEX COVER?
- \langle G, k \rangle \in \text{EXACT VERTEX COVER} \iff \exists \text{ a vertex cover of size } k \text{ in } G \text{ and all other vertex covers have size at least } k.
Two boolean formulas $\phi$ and $\psi$ are equivalent if they evaluate to the same value on all assignments to their variables.
A New Problem on Minimum Equivalent Boolean Formulas

- Two boolean formulas $\phi$ and $\psi$ are equivalent if they evaluate to the same value on all assignments to their variables.
- A **minimal boolean formula** is a boolean formula that has no shorter equivalent.
A New Problem on Minimum Equivalent Boolean Formulas

- Two boolean formulas $\phi$ and $\psi$ are equivalent if they evaluate to the same value on all assignments to their variables.
- A **minimal boolean formula** is a boolean formula that has no shorter equivalent.

\[
\text{MINEQFORMULA} = \{ \langle \phi \rangle \mid \phi \text{ is a minimal Boolean formula} \}
\]
A New Problem on Minimum Equivalent Boolean Formulas

- Two boolean formulas $\phi$ and $\psi$ are equivalent if they evaluate to the same value on all assignments to their variables.
- A **minimal boolean formula** is a boolean formula that has no shorter equivalent.

\[ \text{MINEQFORMULA} = \{ <\phi> \mid \phi \text{ is a minimal Boolean formula} \} \]

- Again, a short certificate eludes us.
Two boolean formulas $\phi$ and $\psi$ are equivalent if they evaluate to the same value on all assignments to their variables.

A **minimal boolean formula** is a boolean formula that has no shorter equivalent.

$$\text{MINEQFORMULA} = \{ <\phi> \mid \phi \text{ is a minimal Boolean formula} \}$$

Again, a short certificate eludes us.

So, the problems are not in NP. Can they be in coNP?
Recall the definitions of NP and coNP

Recall Definition of class NP

A language $L \subseteq \{0, 1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a poly-time TM $M$ s.t. for every $x \in \{0, 1\}^*$,

$$x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1$$
Recall the definitions of NP and coNP

Recall Definition of class NP

A language \( L \subseteq \{0, 1\}^* \) is in NP if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) and a poly-time TM \( M \) s.t. for every \( x \in \{0, 1\}^* \),

\[
 x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1
\]

Recall Definition of coNP

For every \( L \subseteq \{0, 1\}^* \), we say that \( L \in \text{coNP} \) if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) and a poly-time TM \( M \) s.t. for every \( x \in \{0, 1\}^* \),

\[
 x \in L \iff \forall u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1
\]
Recall the definitions of NP and coNP

Recall Definition of class NP

A language \( L \subseteq \{0, 1\}^* \) is in NP if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) and a poly-time TM \( M \) s.t. for every \( x \in \{0, 1\}^* \),

\[
x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1
\]

Recall Definition of coNP

For every \( L \subseteq \{0, 1\}^* \), we say that \( L \in \text{coNP} \) if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) and a poly-time TM \( M \) s.t. for every \( x \in \{0, 1\}^* \),

\[
x \in L \iff \forall u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1
\]

What about the three problems?

EXACT INDSET, EXACT VERTEX COVER, MINEQFORMULA seems to be neither in NP nor in coNP.
Some New Problems and a New Class

Alternating Turing Machines

Can we design a TM that can solve the above problems?

What power do we need to give the TM to solve the above problems?

It has to be something more than non-determinism.

Consider the tree corresponding to the non-deterministic computation. Each node corresponds to a configuration. We can think of the computation as each node computing the OR operation of its children and the NDTM accepts if any of its children throws an accepting configuration.
Alternating Turing Machines

Can we design a TM that can solve the above problems?
Can we design a TM that can solve the above problems?

What power do we need to give the TM to solve the above problems?
Alternating Turing Machines

- Can we design a TM that can solve the above problems?
- What power do we need to give the TM to solve the above problems?
- It has to be something more than non-determinism.
Can we design a TM that can solve the above problems?

What power do we need to give the TM to solve the above problems?

It has to be something more than non-determinism.

Consider the tree corresponding to the non-deterministic computation. Each node corresponds to a configuration.
## Alternating Turing Machines

- Can we design a TM that can solve the above problems?
- What power do we need to give the TM to solve the above problems?
- It has to be something more than non-determinism.
- Consider the tree corresponding to the non-deterministic computation. Each node corresponds to a configuration.
- We can think of the computation as each node computing the OR operation of its children and the NDTM accepts if any of its children throws an accepting configuration.
Alternating Turing Machines continued
Alternating Turing Machines continued

In an alternating computation, the nodes may compute the AND or OR operations.
Alternating Turing Machines continued

Alternating Turing Machines

- In an alternating computation, the nodes may compute the AND or OR operations.
- The above corresponds to an alternating acceptance mode where the TM accepts if all or any of its children accept.
The Alternating Turing Machine (ATM)

An alternating Turing Machine is a NDTM with its states, except for the accept and reject states, divided into universal and existential states.

In a run of an ATM on a string, each node of its non-deterministic computation tree is labeled with ∧ or ∨, depending on whether the corresponding configuration contains a universal or existential state.

Acceptance is determined by designating a node to be accepting if it is labeled with ∧ and all of its children are accepting or if it is labeled with ∨ and any of its children are accepting.

Does this new concept of ATM help in deciding the languages EXACT INDSET, EXACT VERTEX COVER, MINEQFORMULA?
Alternating Turing Machines continued

The Alternating Turing Machine (ATM)

- An alternating Turing Machine is a NDTM with its states, except for the accept and reject states, divided into universal and existential states.
Alternating Turing Machines continued

The Alternating Turing Machine (ATM)

- An alternating Turing Machine is a NDTM with its states, except for the accept and reject states, divided into universal and existential states.

- In a run of an ATM on a string, each node of its non-deterministic computation tree is labeled with $\land$ or $\lor$, depending on whether the corresponding configuration contains a universal or existential state.
### The Alternating Turing Machine (ATM)

- An alternating Turing Machine is a NDTM with its states, except for the accept and reject states, divided into **universal** and **existential** states.

- In a run of an ATM on a string, each node of its non-deterministic computation tree is labeled with ∧ or ∨, depending on whether the corresponding configuration contains a universal or existential state.

- Acceptance is determined by designating a node to be accepting if it is labeled with ∧ and all of its children are accepting or if it is labeled with ∨ and any of its children are accepting.
Alternating Turing Machines continued

The Alternating Turing Machine (ATM)

- An alternating Turing Machine is a NDTM with its states, except for the accept and reject states, divided into universal and existential states.

- In a run of an ATM on a string, each node of its non-deterministic computation tree is labeled with $\land$ or $\lor$, depending on whether the corresponding configuration contains a universal or existential state.

- Acceptance is determined by designating a node to be accepting if it is labeled with $\land$ and all of its children are accepting or if it is labeled with $\lor$ and any of its children are accepting.

- Does this new concept of ATM help in deciding the languages EXACT INDSET, EXACT VERTEX COVER, MINEQFORMULA?
A New Class

Definition

The class $\Sigma_p^2$ is the set of all languages $L$ for which $\exists$ a polynomial time TM $M$ and a polynomial $q$ such that

$$x \in L \iff \exists u \in \{0,1\}^{q(|x|)} \forall v \in \{0,1\}^{q(|x|)} \text{ s. t. } M(x,u,v) = 1$$

for every $x \in \{0,1\}^*$
A New Class

Definition

The class $\Sigma^p_2$ is the set of all languages $L$ for which $\exists$ a polynomial time TM $M$ and a polynomial $q$ such that

$x \in L \iff \exists u \in \{0, 1\}^{q(|x|)} \forall v \in \{0, 1\}^{q(|x|)} \text{ s. t. } M(x, u, v) = 1$

for every $x \in \{0, 1\}^*$

Relation between NP, coNP and $\Sigma^p_2$

$NP, coNP \subseteq \Sigma^p_2$. 

Some New Problems and a New Class

A New Class

Definition

The class $\Sigma^p_2$ is the set of all languages $L$ for which $\exists$ a polynomial time TM $M$ and a polynomial $q$ such that

$$x \in L \iff \exists u \in \{0, 1\}^{q(|x|)} \forall v \in \{0, 1\}^{q(|x|)} \text{ s. t. } M(x, u, v) = 1$$

for every $x \in \{0, 1\}^*$

Relation between NP, coNP and $\Sigma^p_2$

NP, coNP $\subseteq \Sigma^p_2$.

Examples

EXACT INDSET, EXACT VERTEX COVER, MINEQFORMULA $\in \Sigma^p_2$. 
EXACT INDSET is in $\Sigma^p_2$. Why?
EXACT INDSET is in $\Sigma_2^p$. Why?

A pair $< G, k >$ is in EXACT INDSET iff $\exists$ a size-$k$ subset $S$ of $G$'s vertices s.t. for every $S'$ that is a $(k + 1)$-sized subset, $S$ is an independent set in $G$ and $S'$ is not an independent set in $G$. 
Some New Problems and a New Class

Polynomial Hierarchy
Polynomial Hierarchy

The definition of polynomial hierarchy generalizes the definitions of $NP$, $coNP$, $\Sigma^p_2$. 
Polynomial Hierarchy

- The definition of polynomial hierarchy generalizes the definitions of NP, coNP, \( \Sigma^P_2 \).
- This class consists of every language that can be defined via a combination of a poly-time computable predicate and a constant number of \( \forall, \exists \) quantifiers.
Polynomial Hierarchy

- The definition of polynomial hierarchy generalizes the definitions of NP, coNP, $\Sigma_2^P$.
- This class consists of every language that can be defined via a combination of a poly-time computable predicate and a constant number of $\forall$, $\exists$ quantifiers.

**Definition: Polynomial Hierarchy**

For $i \geq 1$, a language $L$ is in $\Sigma_i^P$ if $\exists$ a poly-time TM $M$ and a polynomial $q$ such that

$$x \in L \iff \exists u_1 \in \{0, 1\}^{q(|x|)} \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$$

where $Q_i$ denotes $\forall$ or $\exists$ depending on whether $i$ is even or odd, respectively.

The polynomial hierarchy is the set $\text{PH} = \bigcup_i \Sigma_i^P$. 
Some Relations Involving PH

- $\Sigma^p_1 = \text{NP}$. 
Some New Problems and a New Class Polynomial Hierarchy

Some Relations Involving PH

- $\Sigma_1^p = \text{NP}$.
- For every $i$, define $\Pi_i^p = \text{co}\Sigma_i^p = \{\overline{L} \mid L \in \Sigma_i^p\}$. 
Some Relations Involving PH

- $\Sigma_1^p = \text{NP}$.
- For every $i$, define $\Pi_i^p = \text{co}\Sigma_i^p = \{\overline{L} \mid L \in \Sigma_i^p\}$.
- So, $\Pi_1^p = \text{coNP}$.
Some Relations Involving PH

- $\Sigma_1^p = \text{NP}$.
- For every $i$, define $\Pi_i^p = \text{co}\Sigma_i^p = \{\overline{L} \mid L \in \Sigma_i^p\}$.
- So, $\Pi_1^p = \text{coNP}$.
- For every $i$, $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$. 
Some Relations Involving PH

- $\Sigma_1^p = \text{NP}$. 
- For every $i$, define $\Pi_i^p = \text{co}\Sigma_i^p = \{ \overline{L} \mid L \in \Sigma_i^p \}$. 
- So, $\Pi_1^p = \text{coNP}$. 
- For every $i$, $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$. 
- Hence, $\text{PH} = \bigcup_{i > 0} \Pi_i^p$. 
Properties of the Polynomial Hierarchy

- As we believe, $P \neq NP$ and $NP \neq coNP$, we also tend to believe $\Sigma_i^p$ is strictly contained in $\Sigma_{i+1}^p$.
As we believe, \( P \neq NP \) and \( NP \neq coNP \), we also tend to believe \( \Sigma^P_i \) is strictly contained in \( \Sigma^P_{i+1} \).

This conjecture is stated as the polynomial hierarchy does not collapse.
Properties of the Polynomial Hierarchy

- As we believe, $P \neq NP$ and $NP \neq coNP$, we also tend to believe $\Sigma^P_i$ is strictly contained in $\Sigma^P_{i+1}$.
- This conjecture is stated as the polynomial hierarchy does not collapse.
- The polynomial hierarchy is said to collapse if there is some $i$ s.t. $\Sigma^P_i = \Sigma^P_{i+1}$.
Properties of the Polynomial Hierarchy

- As we believe, $P \neq NP$ and $NP \neq coNP$, we also tend to believe $\Sigma^P_i$ is strictly contained in $\Sigma^P_{i+1}$.
- This conjecture is stated as the polynomial hierarchy does not collapse.
- The polynomial hierarchy is said to collapse if there is some $i$ s.t. $\Sigma^P_i = \Sigma^P_{i+1}$

**Theorem**

If $P = NP$, then $PH = P$; that is the hierarchy collapses to $P$. 
The Proof

- Assuming $P = NP$, we proceed by induction to show $\Sigma_i^p, \Pi_i^p \subseteq P$. 
The Proof

- Assuming $P = NP$, we proceed by induction to show $\Sigma_i^p, \Pi_i^p \subseteq P$.
- The base case $i = 1$ is true as $\Sigma_1^p = NP$ and $\Pi_1^p = coNP$ and our assumption is $P = NP$. 
The Proof

- Assuming $P = NP$, we proceed by induction to show $\Sigma^p_i$, $\Pi^p_i \subseteq P$.
- The base case $i = 1$ is true as $\Sigma^p_1 = NP$ and $\Pi^p_1 = coNP$ and our assumption is $P = NP$.
- We assume it is true for $i - 1$ and prove $\Sigma^p_i \subseteq P$. 
The Proof

- Assuming $P = NP$, we proceed by induction to show $\Sigma_i^p, \Pi_i^p \subseteq P$.
- The base case $i = 1$ is true as $\Sigma_1^p = NP$ and $\Pi_1^p = coNP$ and our assumption is $P = NP$.
- We assume it is true for $i - 1$ and prove $\Sigma_i^p \subseteq P$.
- Let $L \in \Sigma_i^p$. So, $\exists$ a poly-time TM $M$ and a polynomial $q$ s.t.

$$x \in L \iff \exists u_1 \in \{0, 1\}^{q(|x|)} \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$$
The Proof

1. Define a new language $L'$ as:

$$< x, u_1 > \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$$

Clearly, $L' \in \Pi_{p_i-1}$, and so using our inductive assumption, $L' \in \Sigma$. So, a det. poly-time TM $M'$ decides $L'$.

Plug in this $M'$ in the equation pertaining to the definition of $\Sigma$ to get

$$x \in L' \iff \exists u_1 \in \{0, 1\}^{q(|x|)} M'(x, u_1) = 1$$

So, $L \in \Sigma$ and hence, it is in $\Sigma$. Since $\Pi_{p_i}$ consists of complements of languages in $\Sigma$, and $\Sigma$ is closed under complementation, we have $\Pi_{p_i} \subseteq \Sigma$. 
The Proof

- Define a new language $L'$ as:

  $< x, u_1 > \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots$

  $Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$

- Clearly, $L' \in \Pi^p_{i-1}$, and so using our inductive assumption, $L' \in P$. 

- So, a det. poly-time TM $M'$ decides $L'$.

Plug in this $M'$ in the equation pertaining to the definition of $\Sigma^p_i$ to get

$x \in L' \iff \exists u_1 \in \{0, 1\}^{q(|x|)} M'(x, u_1) = 1$

So, $L \in \text{NP}$ and hence, it is in $P$.

Since $\Pi^p_i$ consists of complements of languages in $\Sigma^p_i$ and $P$ is closed under complementation, we have $\Pi^p_i \subseteq P$. 


The Proof

- Define a new language $L'$ as:

\[ <x, u_1> \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1 \]

- Clearly, $L' \in \Pi_{i-1}^P$, and so using our inductive assumption, $L' \in P$.

- So, a det. poly-time TM $M'$ decides $L'$. 
The Proof

- Define a new language $L'$ as:

$$< x, u_1 > \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots$$

$$Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$$

- Clearly, $L' \in \Pi_{i-1}^P$, and so using our inductive assumption, $L' \in P$.
- So, a det. poly-time TM $M'$ decides $L'$.
- Plug in this $M'$ in the equation pertaining to the definition of $\Sigma_i^P$ to get

$$x \in L \iff \exists u_1 \in \{0, 1\}^{q(|x|)} M'(x, u_1) = 1$$
The Proof

- Define a new language $L'$ as:

  \[
  <x, u_1> \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots \quad Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1
  \]

- Clearly, $L' \in \Pi_{i-1}^P$, and so using our inductive assumption, $L' \in P$.
- So, a det. poly-time TM $M'$ decides $L'$.
- Plug in this $M'$ in the equation pertaining to the definition of $\Sigma_i^P$ to get

  \[x \in L \iff \exists u_1 \in \{0, 1\}^{q(|x|)} M'(x, u_1) = 1\]

- So, $L \in NP$ and hence, it is in $P$. 
The Proof

- Define a new language $L'$ as:

$$< x, u_1 > \in L' \iff \forall u_2 \in \{0, 1\}^{q(|x|)} \ldots Q_i u_i \in \{0, 1\}^{q(|x|)} M(x, u_1, u_2, \ldots, u_i) = 1$$

- Clearly, $L' \in \Pi_{i-1}^p$, and so using our inductive assumption, $L' \in P$.
- So, a det. poly-time TM $M'$ decides $L'$.
- Plug in this $M'$ in the equation pertaining to the definition of $\Sigma_i^p$ to get

$$x \in L \iff \exists u_1 \in \{0, 1\}^{q(|x|)} M'(x, u_1) = 1$$

- So, $L \in \text{NP}$ and hence, it is in $P$.
- Since $\Pi_i^p$ consists of complements of languages in $\Sigma_i^p$, and $P$ is closed under complementation, we have $\Pi_i^p \subseteq P$. 
Another Theorem about Collapsing of Polynomial Hierarchy

Theorem

For every $i \geq 1$, if $\Sigma_i^p = \Pi_i^p$, then $\text{PH} = \Sigma_i^p$; that is, the hierarchy collapses to the $i^{th}$ level.
Another Theorem about Collapsing of Polynomial Hierarchy

**Theorem**
For every $i \geq 1$, if $\Sigma_i^p = \Pi_i^p$, then $\text{PH} = \Sigma_i^p$; that is, the hierarchy collapses to the $i^{th}$ level.

**Proof**
Left as an exercise. Hints: Try as the previous proof.