BILL AND NATHAN, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

PCP

Review NP

Def $A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

We want to rewrite this and modify it.

Review NP

Def $A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

We want to rewrite this and modify it.

Intuition
All powerful Alice is trying to convince Poly-Bob that $x \in A$.

Review NP

Def $A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

We want to rewrite this and modify it.

Intuition
All powerful Alice is trying to convince Poly-Bob that $x \in A$.

1. If $x \in A$ then Alice can send Bob y and Bob can verify it. NOTE- he is **sure** that $x \in A$.

Review NP

Def $A \in \mathrm{NP}$ if there exists $B \in \mathrm{P}$ such that

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

We want to rewrite this and modify it.

Intuition
All powerful Alice is trying to convince Poly-Bob that $x \in A$.

1. If $x \in A$ then Alice can send Bob y and Bob can verify it. NOTE- he is **sure** that $x \in A$.
2. If $x \notin A$ then **whatever** y Alice sends Bob, Bob is NOT convinced. **Not even a little.**

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.
3. Bob is **never wrong**.

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.
3. Bob is **never wrong**.

Imagine if:

Modify NP

$$A = \{x : (\exists^p y)[(x,y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.
3. Bob is **never wrong**.

Imagine if:

1. Bob only got to read **some of** y.

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.
3. Bob is **never wrong**.

Imagine if:

1. Bob only got to read **some of** y.
2. Bob uses a **randized** algorithm.

Modify NP

$$A = \{x : (\exists^p y)[(x, y) \in B]\}.$$

Note that:

1. Bob gets to read the **entire** string y.
2. Bob uses a **deterministic** algorithm.
3. Bob is **never wrong**.

Imagine if:

1. Bob only got to read **some of** y.
2. Bob uses a **randized** algorithm.
3. Bob is **wrong a small fraction of the time**.

Examples

Alice wants to convince Bob that

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k \in 3\mathrm{SAT}$$

Examples

Alice wants to convince Bob that

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k \in \mathrm{3SAT}$$

1. Alice send Bob a truth assignment $\vec{b} \in \{0, 1\}^n$.

Examples

Alice wants to convince Bob that

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k \in \mathrm{3SAT}$$

1. Alice send Bob a truth assignment $\vec{b} \in \{0,1\}^n$.
2. Bob **randly** picks $\lg n$ clauses ($\leq 3 \lg n$ vars).
   Bob looks at partial truth assignment $\vec{p}$ on $\leq 3 \lg n$ vars.

Examples

Alice wants to convince Bob that

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k \in \mathrm{3SAT}$$

1. Alice send Bob a truth assignment $\vec{b} \in \{0,1\}^n$.
2. Bob **randly** picks $\lg n$ clauses ($\leq 3 \lg n$ vars).
   Bob looks at partial truth assignment $\vec{p}$ on $\leq 3 \lg n$ vars.
3. 3.1 If $\vec{p}$ satisfies **all** $\lg n$ clauses, Bob thinks that ϕ is prob SAT, and says YES. (He might be wrong.)

Examples

Alice wants to convince Bob that

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k \in \mathrm{3SAT}$$

1. Alice send Bob a truth assignment $\vec{b} \in \{0,1\}^n$.
2. Bob **randly** picks $\lg n$ clauses ($\leq 3 \lg n$ vars).
   Bob looks at partial truth assignment $\vec{p}$ on $\leq 3 \lg n$ vars.
3. 3.1 If $\vec{p}$ satisfies **all** $\lg n$ clauses, Bob thinks that ϕ is prob SAT, and says YES. (He might be wrong.)
   3.2 If there is ≥ 1 clause that $\vec{p}$ does NOT satisfy then Bob KNOWS ϕ is not satisfied, and says NO. (He is right.)

How Good is the 3SAT Protocol

If $\phi \in 3\mathrm{SAT}$ then the protocol works fine.

How Good is the 3SAT Protocol

If $\phi \in \mathrm{3SAT}$ then the protocol works fine.

If $\phi \notin \mathrm{3SAT}$ but there is a way to satisfy all but 1 clause then this will almost surely fool Bob :-(

How Good is the 3SAT Protocol

If $\phi \in \mathrm{3SAT}$ then the protocol works fine.

If $\phi \notin \mathrm{3SAT}$ but there is a way to satisfy all but 1 clause then this will almost surely fool Bob :-(

If $\phi \notin \mathrm{3SAT}$ but the max number of clauses that can be satisfied is $\leq$ some fraction of the clauses then this protocol works well.

How Good is the 3SAT Protocol

If $\phi \in 3\mathrm{SAT}$ then the protocol works fine.

If $\phi \notin 3\mathrm{SAT}$ but there is a way to satisfy all but 1 clause then this will almost surely fool Bob :-(

If $\phi \notin 3\mathrm{SAT}$ but the max number of clauses that can be satisfied is $\leq$ some fraction of the clauses then this protocol works well.

BREAKOUT ROOMS Get a similar protocol for 3COL.

3COL With Just a Few Bits

Alice wants to convince Bob G is 3-Colorable.

3COL With Just a Few Bits

Alice wants to convince Bob G is 3-Colorable.

1. Alice send Bob ρ, a 3-coloring of G.

3COL With Just a Few Bits

Alice wants to convince Bob G is 3-Colorable.

1. Alice send Bob ρ, a 3-coloring of G.
2. Bob **randly** picks $\lg n$ edges.
   He looks at ρ' which is ρ restricted to the endpoints of the edges.

3COL With Just a Few Bits

Alice wants to convince Bob G is 3-Colorable.

1. Alice send Bob ρ, a 3-coloring of G.
2. Bob **randly** picks $\lg n$ edges.
   He looks at ρ' which is ρ restricted to the endpoints of the edges.
3. 3.1 If ρ' is a proper 3-coloring of the subgraph it colors then Bob things G is prob 3-colorable, and says YES. (He might be wrong.)

3COL With Just a Few Bits

Alice wants to convince Bob G is 3-Colorable.

1. Alice send Bob ρ, a 3-coloring of G.
2. Bob **randly** picks $\lg n$ edges.
   He looks at ρ' which is ρ restricted to the endpoints of the edges.
3. 3.1 If ρ' is a proper 3-coloring of the subgraph it colors then Bob things G is prob 3-colorable, and says YES. (He might be wrong.)
   3.2 If ρ' is not a proper 3-coloring of the subgraph it colors then then Bob KNOWS G is not 3-colorable and says NO (He is right.)

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.
3. A tape called THE ANSWER TAPE.

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.
3. A tape called THE ANSWER TAPE.

We denote such a device by $M^{()}$.

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.
3. A tape called THE ANSWER TAPE.

We denote such a device by $M^{()}$.
Given a string y (called an oracle) and an input x we can compute $M^{y}(x)$ as follows:

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.
3. A tape called THE ANSWER TAPE.

We denote such a device by $M^{()}$.
Given a string y (called an oracle) and an input x we can compute $M^{y}(x)$ as follows:
Simulate the machine. When it goes into state QUERY then there is a number written on the query tape, say i. Then the ith bit of y magically appears on the answer tape.

Polynomial Oracle Turing Machines-Bit Access

Def A **Poly Oracle Turing Machine–bit access (henceforth POTM-BA)** is a poly time TM which has

1. A state QUERY.
2. A tape called the QUERY TAPE.
3. A tape called THE ANSWER TAPE.

We denote such a device by $M^{()}$.
Given a string y (called an oracle) and an input x we can compute $M^y(x)$ as follows:
Simulate the machine. When it goes into state QUERY then there is a number written on the query tape, say i. Then the ith bit of y magically appears on the answer tape.
We will not need this formality but it is good to know that our concepts can be made formal.

Randomized POTM-BA

We give two equivalent definitions of a **Randomized POTM-BA (RPOTM-BA)**. One is intuitive and the other is better for proofs.

Randomized POTM-BA

We give two equivalent definitions of a **Randomized POTM-BA (RPOTM-BA)**. One is intuitive and the other is better for proofs.

Def One A **RPOTM-BA** is a POTM-BA that is allowed to flip coins. We care about **prob $M^y(x)$ accepts**.

Randomized POTM-BA

We give two equivalent definitions of a **Randomized POTM-BA (RPOTM-BA)**. One is intuitive and the other is better for proofs.

Def One A **RPOTM-BA** is a POTM-BA that is allowed to flip coins. We care about **prob $M^y(x)$ accepts**.

Def Two (Coin flips are part of the input.) A **RPOTM-BA** is a POTM-BA that has 2 inputs x, τ. Given an oracle y we care about the fraction of τ's for which $M^y(x, \tau)$ accepts. We will refer to τ as a **string of coin flips.**

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \to (\exists^p y)[M^y(x) = 1]$

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \rightarrow (\exists^p y)[M^y(x) = 1]$
$x \notin A \rightarrow (\forall^p y)[M^y(x) \neq 1]$

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \rightarrow (\exists^p y)[M^y(x) = 1]$
$x \notin A \rightarrow (\forall^p y)[M^y(x) \neq 1]$

If $x \in A$ then we think of y as being the EVIDENCE that $x \in A$. This evidence is short (only $p(|x|)$ long) and checkable in poly time. The computation $M^y(x)$ may look at all bits of y.

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \rightarrow (\exists^p y)[M^y(x) = 1]$
$x \notin A \rightarrow (\forall^p y)[M^y(x) \neq 1]$

If $x \in A$ then we think of y as being the EVIDENCE that $x \in A$. This evidence is short (only $p(|x|)$ long) and checkable in poly time. The computation $M^y(x)$ may look at all bits of y.

To formalize our prior discussion about modifying NP we will:

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \rightarrow (\exists^p y)[M^y(x) = 1]$
$x \notin A \rightarrow (\forall^p y)[M^y(x) \neq 1]$

If $x \in A$ then we think of y as being the EVIDENCE that $x \in A$. This evidence is short (only $p(|x|)$ long) and checkable in poly time. The computation $M^y(x)$ may look at all bits of y.

To formalize our prior discussion about modifying NP we will:

1. Restrict the number of bits of the oracle that M can look at.

Alternative definition of NP

Def $A \in \mathrm{NP}$ if there exists a POTM-BA $M^{()}$ such that:
$x \in A \rightarrow (\exists^p y)[M^y(x) = 1]$
$x \notin A \rightarrow (\forall^p y)[M^y(x) \neq 1]$

If $x \in A$ then we think of y as being the EVIDENCE that $x \in A$. This evidence is short (only $p(|x|)$ long) and checkable in poly time. The computation $M^y(x)$ may look at all bits of y.

To formalize our prior discussion about modifying NP we will:

1. Restrict the number of bits of the oracle that M can look at.
2. Use a RPOTM-BA.

Parametrizing RPOTM-BA

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$.

Parametrizing RPOTM-BA

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$.
A ***q*(*n*)-query, *r*(*n*)-rand RPOTM-BA $M^{()}$** is a RPOTM-BA where, for all y and for all x, $|x| = n$:

Parametrizing RPOTM-BA

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$.
A **$q(n)$-query, $r(n)$-rand RPOTM-BA $M^{()}$** is a RPOTM-BA where, for all y and for all x, $|x| = n$:

1. $M^y(x)$ makes $q(n)$ bit queries

Parametrizing RPOTM-BA

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$.
A $\boldsymbol{q(n)}$**-query,** $\boldsymbol{r(n)}$**-rand RPOTM-BA** $\boldsymbol{M^{()}}$ is a RPOTM-BA where, for all y and for all x, $|x| = n$:

1. $M^y(x)$ makes $q(n)$ bit queries
2. $M^y(x)$ flips $r(n)$ coins.

PCP

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$ and $\epsilon(n)$ be a mono decreasing function from $\mathbb{N}$ to $[0, 1]$.

PCP

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$ and $\epsilon(n)$ be a mono decreasing function from $\mathbb{N}$ to $[0,1]$.

$\boldsymbol{A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))}$ if there exists
a $q(n)$-query, $r(n)$-rand RPOTM-BA $M^{()}$
such that, for all n, for all $x \in \{0,1\}^n$, the following holds.

PCP

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$ and $\epsilon(n)$ be a mono decreasing function from $\mathbb{N}$ to $[0, 1]$.

$A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ if there exists
a $q(n)$-query, $r(n)$-rand RPOTM-BA $M^{()}$
such that, for all n, for all $x \in \{0, 1\}^n$, the following holds.

1. If $x \in A$ then there exists y such that, for all τ with $|\tau| = r(n)$, $M^y(x, \tau)$ accepts. In other words, the probability of acceptance is 1.

PCP

Def Let $q(n)$ and $r(n)$ be mono increasing functions from $\mathbb{N}$ to $\mathbb{N}$ and $\epsilon(n)$ be a mono decreasing function from $\mathbb{N}$ to $[0,1]$.

$\boldsymbol{A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))}$ if there exists
a $q(n)$-query, $r(n)$-rand RPOTM-BA $M^{()}$
such that, for all n, for all $x \in \{0,1\}^n$, the following holds.

1. If $x \in A$ then there exists y such that, for all τ with $|\tau| = r(n)$, $M^y(x,\tau)$ accepts. In other words, the probability of acceptance is 1.
2. If $x \notin A$ then for all y at most $\epsilon(n)$ of the τ's with $|\tau| = r(n)$ make $M^y(x,\tau)$ accept. In other words, the probability of acceptance is $\leq \epsilon(n)$.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.
2. How many questions could be asked?

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.
2. How many questions could be asked? Draw out the tree of all computations. This will branch two ways for every query and for every rand bit.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.
2. How many questions could be asked? Draw out the tree of all computations. This will branch two ways for every query and for every rand bit. Hence there are $2^{q(n)+r(n)}$ possible questions.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.
2. How many questions could be asked?
   Draw out the tree of all computations. This will branch two ways for every query and for every rand bit.
   Hence there are $2^{q(n)+r(n)}$ possible questions.
   Hence we can take $|y| = 2^{q(n)+r(n)}$.

Adaptive Queries and the Length of y

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$.

1. In the $M^y(x)$ calculation the queries are made adaptively. E.g., the 2nd question may depend on the answer to the 1st question.
2. How many questions could be asked?
   Draw out the tree of all computations. This will branch two ways for every query and for every rand bit.
   Hence there are $2^{q(n)+r(n)}$ possible questions.
   Hence we can take $|y| = 2^{q(n)+r(n)}$.
   We will always have $q(n), r(n) = O(\log n)$ so $|y|$ is poly in n.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$?

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:

1) How do you simulate a rand computation?

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:

1) How do you simulate a rand computation?

2) What about the queries? We don't have y.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:

1) How do you simulate a rand computation?

2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.
Will need $r(n) = O(\log n)$ and $q(n) = O(\log n)$.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.
Will need $r(n) = O(\log n)$ and $q(n) = O(\log n)$.
We will do this in our proof that CLIQ is hard to approx.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) $\forall$ **rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.
Will need $r(n) = O(\log n)$ and $q(n) = O(\log n)$.
We will do this in our proof that CLIQ is hard to approx.

2) $\forall$ **rand τ find Bool fml of bit-answers** that leads to accept.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.
Will need $r(n) = O(\log n)$ and $q(n) = O(\log n)$.
We will do this in our proof that CLIQ is hard to approx.

2) **$\forall$ rand τ find Bool fml of bit-answers** that leads to accept.
Need $r(n) = O(\log n)$ and $q(n) = O(1)$.

Simulating a PCP

Let $A \in \mathrm{PCP}(q(n), r(n), \epsilon(n))$ via $M^{()}$.

Can we simulate $M^{()}(x)$? No:
1) How do you simulate a rand computation?
2) What about the queries? We don't have y.

There are two ways to look at this.

1) **$\forall$ rand τ and $\forall$ bit answers σ do a simulation.**
If we do this we will need to also keep track of what bit-queries were asked and how they were answered.
Will need $r(n) = O(\log n)$ and $q(n) = O(\log n)$.
We will do this in our proof that CLIQ is hard to approx.

2) **$\forall$ rand τ find Bool fml of bit-answers** that leads to accept.
Need $r(n) = O(\log n)$ and $q(n) = O(1)$.
We will do this in our proof that $\mathrm{MAX3SAT}$ is hard to approx.

The PCP Theorem and Some Additions to it

We state but do not proof (hard!) PCP theorem and some variants.

Thm

The PCP Theorem and Some Additions to it

We state but do not proof (hard!) PCP theorem and some variants.

Thm

1. $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \frac{1}{2})$.

The PCP Theorem and Some Additions to it

We state but do not proof (hard!) PCP theorem and some variants.

Thm

1. $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \frac{1}{2})$.
2. For all constants $0 < \epsilon < 1$, $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \epsilon)$. (Iterating Part 1.)

The PCP Theorem and Some Additions to it

We state but do not proof (hard!) PCP theorem and some variants.

Thm

1. $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \frac{1}{2})$.
2. For all constants $0 < \epsilon < 1$, $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \epsilon)$. (Iterating Part 1.)
3. $\mathrm{SAT} \in \mathrm{PCP}(O(\log n), O(\log^2 n), \frac{1}{n})$. (Iterating Part 1.)

The PCP Theorem and Some Additions to it

We state but do not proof (hard!) PCP theorem and some variants.

Thm

1. $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \frac{1}{2})$.
2. For all constants $0 < \epsilon < 1$, $\mathrm{SAT} \in \mathrm{PCP}(O(1), O(\log n), \epsilon)$. (Iterating Part 1.)
3. $\mathrm{SAT} \in \mathrm{PCP}(O(\log n), O(\log^2 n), \frac{1}{n})$. (Iterating Part 1.)
4. For all $0 < \epsilon < 1$, $\mathrm{SAT} \in \mathrm{PCP}(O(\log n), O(\log n), \frac{1}{n})$. (Hard! Reuse Random Bits..)

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5 \log n, 0.1).$$

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5\log n, 0.1).$$

Number of possible queries is $2^{4+5\log n} = 2^4 n^5 = p(n)$.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5\log n, 0.1).$$

Number of possible queries is $2^{4+5\log n} = 2^4 n^5 = p(n)$.
Number of rand bits is $\lceil 5\lg n \rceil = L(n)$.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5\log n, 0.1).$$

Number of possible queries is $2^{4+5\log n} = 2^4 n^5 = p(n)$.
Number of rand bits is $\lceil 5 \lg n \rceil = L(n)$.
$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = 1| \le 0.1 \times 2^{L(n)}]$.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5\log n, 0.1).$$

Number of possible queries is $2^{4+5\log n} = 2^4 n^5 = p(n)$.
Number of rand bits is $\lceil 5\lg n \rceil = L(n)$.
$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x,\tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x,\tau) = 1| \le 0.1 \times 2^{L(n)}]$.

Idea One Try all y. There are $2^{p(n)}$ y's, to many.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

I will make up number here to avoid to much notation. $A \in \mathrm{NP}$.

$$A \in \mathrm{PCP}(4, 5\log n, 0.1).$$

Number of possible queries is $2^{4+5\log n} = 2^4 n^5 = p(n)$.
Number of rand bits is $\lceil 5 \lg n \rceil = L(n)$.
$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = 1| \le 0.1 \times 2^{L(n)}]$.

Idea One Try all y. There are $2^{p(n)}$ y's, to many.

Idea Two Try all 38-long bit sequenes for answers. Next Slide.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x,\tau).$$
$$x \notin A \implies (\forall^p y)[|\tau : M^y(x,\tau) = Y| \le 0.1 \times 2^{L(n)}].$$

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau).$$
$$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = Y| \leq 0.1 \times 2^{L(n)}].$$

Idea Two Try all 5-bit sequenes for answers.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau).$$
$$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = Y| \leq 0.1 \times 2^{L(n)}].$$

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x,\tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x,\tau) = Y| \le 0.1 \times 2^{L(n)}]$.

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Look at answer sequene 00000. So, for all $\tau \in \{0,1\}^8$ run $M(x,\tau)$ and answer all queries NO.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = Y| \le 0.1 \times 2^{L(n)}]$.

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Look at answer sequene 00000. So, for all $\tau \in \{0, 1\}^8$ run $M(x, \tau)$ and answer all queries NO.

If use query answers 00000 and rand bits 00000000 then queries were to the 23rd, 32nd, 38th, 40th, 74th bit. Say anwer is NO.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x,\tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x,\tau) = Y| \le 0.1 \times 2^{L(n)}]$.

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Look at answer sequene 00000. So, for all $\tau \in \{0,1\}^8$ run $M(x,\tau)$ and answer all queries NO.

If use query answers 00000 and rand bits 00000000 then queries were to the 23rd, 32nd, 38th, 40th, 74th bit. Say anwer is NO.

If use query answers 00000 and rand bits 0000001 then queries were to the 8th, 15th, 21st, 47th, 98th bit. Say answer is YES.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = Y| \leq 0.1 \times 2^{L(n)}]$.

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Look at answer sequene 00000. So, for all $\tau \in \{0, 1\}^8$ run $M(x, \tau)$ and answer all queries NO.

If use query answers 00000 and rand bits 00000000 then queries were to the 23rd, 32nd, 38th, 40th, 74th bit. Say anwer is NO.

If use query answers 00000 and rand bits 0000001 then queries were to the 8th, 15th, 21st, 47th, 98th bit. Say answer is YES.
Do the same for all query answers length 5, rand bits length 8.

Why does PCP Theorem NOT Imply $\mathrm{P} = \mathrm{NP}$?

$x \in A \implies (\exists^p y)(\forall^L \tau)[M^y(x, \tau)$.
$x \notin A \implies (\forall^p y)[|\tau : M^y(x, \tau) = Y| \le 0.1 \times 2^{L(n)}]$.

Idea Two Try all 5-bit sequenes for answers.
Lets see what happens. We make up the following scenario. Take n such that $L(n) = 8$.

Look at answer sequene 00000. So, for all $\tau \in \{0, 1\}^8$ run $M(x, \tau)$ and answer all queries NO.

If use query answers 00000 and rand bits 00000000 then queries were to the 23rd, 32nd, 38th, 40th, 74th bit. Say anwer is NO.

If use query answers 00000 and rand bits 0000001 then queries were to the 8th, 15th, 21st, 47th, 98th bit. Say answer is YES.
Do the same for all query answers length 5, rand bits length 8.

Put the YES entries in a table. See next page.

Make a Table

Only put the YES's on the table.

Bit Ans	Rand Bits	Queries
00001	00000001	23,32,38,40,74
00001	01000001	13,12,18,19,20
00001	01000001	3,10,32,29,29
00011	00000001	23,30,35,40,80
00011	01000101	13,12,18,19,20
00011	01100001	3,10,32,29,29
00011	01000101	23,37,38,41,75
⋮	⋮	⋮

Make a Table

Only put the YES's on the table.

Bit Ans	Rand Bits	Queries
00001	00000001	23,32,38,40,74
00001	01000001	13,12,18,19,20
00001	01000001	3,10,32,29,29
00011	00000001	23,30,35,40,80
00011	01000101	13,12,18,19,20
00011	01100001	3,10,32,29,29
00011	01000101	23,37,38,41,75
$\vdots$	$\vdots$	$\vdots$

Is there a y such that for all $\tau \in \{0,1\}^8$ there is a sequence of 5 query bits that is consistent with y? If so then $x \in A$, else not.

Make a Table

Only put the YES's on the table.

Bit Ans	Rand Bits	Queries
00001	00000001	23,32,38,40,74
00001	01000001	13,12,18,19,20
00001	01000001	3,10,32,29,29
00011	00000001	23,30,35,40,80
00011	01000101	13,12,18,19,20
00011	01100001	3,10,32,29,29
00011	01000101	23,37,38,41,75
$\vdots$	$\vdots$	$\vdots$

Is there a y such that for all $\tau \in \{0,1\}^8$ there is a sequence of 5 query bits that is consistent with y? If so then $x \in A$, else not.

Example 1st and 7th row consistent with a y that has 23rd bit-0, 32nd bit-0, 37th bit-0, 38th bit-0, 40th bit-0, 41st bit-1, 74th bit-1, 75th bit-1

How Hard is this Consistency Problem?

If you formalize this problem (and have $L(n)$ random bits, not 8) then it is a string-consistency problem that is

How Hard is this Consistency Problem?

If you formalize this problem (and have $L(n)$ random bits, not 8) then it is a string-consistency problem that is

NP-Complete

How Hard is this Consistency Problem?

If you formalize this problem (and have $L(n)$ random bits, not 8) then it is a string-consistency problem that is

NP-Complete

PRO This means that the PCP theorem does not show $\mathrm{P} = \mathrm{NP}$. Hence the Erik-Bill-Mohammad book will not be obsolete upon publication.

How Hard is this Consistency Problem?

If you formalize this problem (and have $L(n)$ random bits, not 8) then it is a string-consistency problem that is

NP-Complete

PRO This means that the PCP theorem does not show $\mathrm{P} = \mathrm{NP}$. Hence the Erik-Bill-Mohammad book will not be obsolete upon publication.

CON If we DID get $\mathrm{P} = \mathrm{NP}$ we could solve lots of things quickly.

How Hard is this Consistency Problem?

If you formalize this problem (and have $L(n)$ random bits, not 8) then it is a string-consistency problem that is

NP-Complete

PRO This means that the PCP theorem does not show $\mathrm{P} = \mathrm{NP}$. Hence the Erik-Bill-Mohammad book will not be obsolete upon publication.

CON If we DID get $\mathrm{P} = \mathrm{NP}$ we could solve lots of things quickly.

I WONDER The PCP theorem can be intrepted as-

NP is easier than we thought

and hence evidence that $\mathrm{P} = \mathrm{NP}$.