

Assignment Project Exam Help

Algorithms: https://powscoeler.com

Add WeChat powcoder

School of Computer Science and Engineering University of New South Wales

11. INTRACTABILITY

• We say that a (sequential) algorithm is *polynomial time* if for every input it terminates in polynomially many steps in the length of the input.

Assignment Project Exam Help

https://powcoder.com

• We say that a (sequential) algorithm is *polynomial time* if for every input it terminates in polynomially many steps in the length of the input.

As subsection that the property of the input. As subsection of the input.

As subsection in $T(n) = O(n^n)$ many steps, where n is the size of the input.

https://powcoder.com

• We say that a (sequential) algorithm is *polynomial time* if for every input it terminates in polynomially many steps in the length of the input.

As that we algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

• A delicitups://powcoder.com

• We say that a (sequential) algorithm is *polynomial time* if for every input it terminates in polynomially many steps in the length of the input.

As subsection that the algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicitupe id problem with AFT Com
- Examples are:
 - \bullet "Input number n is a prime number."

• We say that a (sequential) algorithm is *polynomial time* if for every input it terminates in polynomially many steps in the length of the input.

As that we algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicitupes://powcoder.com
- Examples are:
 - "Input number n is a prime number."
 - "And We Chat powcoder

• We say that a (sequential) algorithm is polynomial time if for every input it terminates in polynomially many steps in the length of the input.

Signs means the deeper set of tanger and the find pening of the next to that the algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicity problem is a problem with a XEN XC answer on
- Examples are:
 - "Input number n is a prime number."

 - "Aput or on the control of the contr

• We say that a (sequential) algorithm is polynomial time if for every input it terminates in polynomially many steps in the length of the input.

s means that the pectist a takes a point k independing of the point to the algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicity problem is a problem with a XEN XC answer on
- Examples are:
 - "Input number n is a prime number."

 - "Aput or on the control of the contr
- We say that a decision problem A is in polynomial time if there exists a polynomial time algorithm which solves it.

• We say that a (sequential) algorithm is polynomial time if for every input it terminates in polynomially many steps in the length of the input.

s means ma Aere exist. A facult amount k into pealing of the next to the algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicity problem with a XEA NO answer of the Control of the North Control of the Control of
- Examples are:
 - "Input number n is a prime number."

 - "Aput or on the control of the contr
- We say that a decision problem A is in polynomial time if there exists a polynomial time algorithm which solves it.
- Thus, given an input x, such an algorithm outputs YES for all x which satisfy A and outputs NO for all x which do not satisfy A.

• We say that a (sequential) algorithm is polynomial time if for every input it terminates in polynomially many steps in the length of the input.

s means that the peckist a takes a part k the pecking of the pecking the algorithm terminates in $T(n) = O(n^n)$ many steps, where n is the size of the input.

- A delicity problem is a problem with a XEN XC answer on
- Examples are:
 - "Input number n is a prime number."

 - "Aput or on the control of the contr
- We say that a decision problem A is in polynomial time if there exists a polynomial time algorithm which solves it.
- Thus, given an input x, such an algorithm outputs YES for all x which satisfy A and outputs NO for all x which do not satisfy A.
- We denote this by $A \in \mathbf{P}$.

Assignment Project Exam Help

https://powcoder.com

Assignment Project Exam Help

• It is the number of symbols needed to describe the input precisely.

https://powcoder.com

Assignment Project Exam Help

- It is the number of symbols needed to describe the input precisely.
- Examples:

https://powcoder.com

Assignment Project Exam Help

- It is the number of symbols needed to describe the input precisely.
- Examples:
 - Hithing San integration of x.

Assignment Project Exam Help

- It is the number of symbols needed to describe the input precisely.
- Examples:
 - Hitu San integration of x.
 - As we will see, definition of polynomial time computability is quite robust with respect to how we represent inputs.

Assignment Project Exam Help

- It is the number of symbols needed to describe the input precisely.
- Examples:
 - Hithing San integration of x.
 - As we will see, definition of polynomial time computability is quite robust with respect to how we represent inputs.
 - For example, we could also define the length of an integer r as the number of digits in the desimal terrespectation of COCT

Assignment Project Exam Help

- It is the number of symbols needed to describe the input precisely.
- Examples:
 - Hithir San integral to the table to be the full of bits in the binary representation of x.
 - As we will see, definition of polynomial time computability is quite robust with respect to how we represent inputs.
 - Har example, we could also define the length of an integer r as the number ordigits in the decimal depresentation of COCT
 - This can only change the constants involved in the expression $T(n) = O(n^k)$ but not the asymptotic bound.

If input is a weighted graph G, then G can be described by giving for each As represented in binary. The properties with their distance P

https://powcoder.com

If input is a weighted graph G, then G can be described by giving for each $As_{eproposited}$ in the following f

ullet Alternatively, we can represent G with its adjacency matrix.

https://powcoder.com

• If input is a weighted graph G, then G can be described by giving for each $As_{\text{represented in Binery.}}^{\text{graph}}$ for each $As_{\text{represented in Binery.}}^{\text{graph}}$

- ullet Alternatively, we can represent G with its adjacency matrix.
- If the input raphs are ar sparse, this can unnecessarily increase the length of the representation of the graph.

If input is a weighted graph G, then G can be described by giving for each G carries and interpretable G in the G can be described by giving for each G can be described by G ca

- ullet Alternatively, we can represent G with its adjacency matrix.
- If the input raphs are ar sparse, this can unnecessarily increase the length of the representation of the graph.
- However, since we are interested only in whether the algorithm runs in polynomial time and not be the particular degree of the polynomial bounding such a run time, this does not matter.

If input is a weighted graph G, then G can be described by giving for each G carries and interpretable for the state of the state

- ullet Alternatively, we can represent G with its adjacency matrix.
- If the input raphs are ar sparse, this can unnecessarily increase the length of the representation of the graph.
- However, since we are interested paly in whether the algorithm runs in polynomial time and not be the particular degree of the polynomial bounding such a run time, this does not matter.
- In fact, every precise description without artificial redundancies will do.

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment Project Exam Help

https://powcoder.com

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment is P to be P(x,y) such that for every input x, A(x) is true and

https://powcoder.com

- We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:
- Assignment is B(x, y) such that for every input x, A(x) is true and
 - 2 such that the trath of B(x,y) can be varified by an algorithm running in phlyrophasime in the law of OUCL.

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment is B(x, y) such that for every input x, A(x) is true and

- 2 such that the truth of B(x,y) can be verified by an algorithm running in pultropia Sime in the Day of COMCT.
- We call y a *certificate* for x.

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment Axy such that for every input x, A(x) is true and

- 2 such that the truth of B(x,y) can be varified by an algorithm running in pull round Stime in the Day of COUCT. COM
- We call y a *certificate* for x.
- For example decision reform "integer y such that B(x,y) = x is divisible by y" is true.

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment is B(x, y) such that for every input x, A(x) is true and

- 2 such that the truth of B(x,y) can be varified by an algorithm running in bluropia sine in the new G.O. Let COM
- We call y a *certificate* for x.
- For example decision repen "integer x is not represent that a = a = b is true just in case there exists integer y such that a = b is divisible by a = b is true.
- Clearly, the problem "x is divisible by y" is decidable by an algorithm which runs in time polynomial in the length of x only.

• We say that a decision problem A(x) is in non-deterministic polynomial time, denoted by $A \in \mathbf{NP}$, if:

Assignment is P there exists a problem B(x, y) such that for every input x, A(x) is true and

- 2 such that the truth of B(x,y) can be varified by an algorithm running in pull round Stime in the Day of COUCT. COM
- We call y a *certificate* for x.
- For example decision replet "integer x is not represent in the case there exists integer y such that B(x,y)= "x is divisible by y" is true.
- Clearly, the problem "x is divisible by y" is decidable by an algorithm which runs in time polynomial in the length of x only.
- In fact, "integer x is not prime" is actually decidable in (deterministic) polynomial time, but this is a hard theorem to prove.

Examples of NP-decision problems:

Assignment Project Exam Help

https://powcoder.com

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

Assignment Project Exam Help

https://powcoder.com

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

Assignment Project Exam Help Clearly, given a subset of vertices U we can determine in polynomial time if U

Clearly, given a subset of vertices U we can determine in polynomial time if U is a vertex cover of G with at most k elements.

https://powcoder.com

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

Assignment Project Exam Help Clearly, given a subset of vertices U we can determine in polynomial time if U

Clearly, given a subset of vertices U we can determine in polynomial time if U is a vertex cover of G with at most k elements.

• (SATING S. pyordi) in White Off F or $\mathbb{N}^{c_1} \wedge \ldots \wedge C_n$ where each cause C_i is a disjunction of propositional variables or their negations, for example

 $\stackrel{(P_1 \vee \neg P_2 \vee P_3 \vee \neg P_5)}{Add} \stackrel{(P_2 \vee P_3 \vee \neg P_5 \vee \neg P_6) \wedge (\neg P_3 \vee \neg P_4 \vee P_5)}{echat}$

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

SSIGNATURE AND A STATE OF STAT

Clearly, given a subset of vertices U we can determine in polynomial time if U is a vertex cover of G with at most k elements.

• (SAThrstap S proposition With the NF or $M^2 \wedge \dots \wedge C_n$ where each clause C_i is a disjunction of propositional variables or their negations, for example

 $(P_1 \vee P_3 \vee P_3 \vee P_5) \wedge (P_2 \vee P_3 \vee P_5 \vee \neg P_6) \wedge (\neg P_3 \vee \neg P_4 \vee P_5)$ Problem The exist are always that the problem was the formula true".

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

Assignment Project Exam Help Clearly, given a subset of vertices U we can determine in polynomial time if U

Clearly, given a subset of vertices U we can determine in polynomial time if U is a vertex cover of G with at most k elements.

• (SAT) ristances proposition With Other Form $C_2 \wedge \ldots \wedge C_n$ where each clause C_i is a disjunction of propositional variables or their negations, for example

 $(P_1 \vee P_2 \vee P_3 \vee P_5) \wedge (P_2 \vee P_3 \vee P_5 \vee P_6) \wedge (\neg P_3 \vee P_4 \vee P_5)$ Problem The exist accomplished the problem when the formula true".

Clearly, given an evaluation of the propositional variables one can determine in polynomial time if the formula is true for such an evaluation.

Examples of NP-decision problems:

• (Vertex Cover) Instance: a graph G and an integer k. Problem: "There exists a subset U consisting of at most k vertices of G (called a vertex cover of G)

Assignment Projecte Exam Help Clearly, given a subset of vertices U we can determine in polynomial time if U

Clearly, given a subset of vertices U we can determine in polynomial time if U is a vertex cover of G with at most k elements.

• (SAThistap S proposition While the Form $C_2 \wedge ... \wedge C_n$ where each dause C_i is a disjunction of propositional variables or their negations, for example

 $(P_1 \vee P_3 \vee P_3 \vee P_5) \wedge (P_2 \vee P_3 \vee \neg P_5 \vee \neg P_6) \wedge (\neg P_3 \vee \neg P_4 \vee P_5)$ Problem Tice exist a calculation the problem when the formula true".

Clearly, given an evaluation of the propositional variables one can determine in polynomial time if the formula is true for such an evaluation.

• If each clause C_i involves exactly three variables we call such decision problem 3SAT.

COMP3121/9101 6 /

ullet As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P.

Assignment Project Exam Help

https://powcoder.com

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists

As staglimme presented result, proved by Indian computer scientists As staglimme present Project Exam Help

https://powcoder.com

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists.

As stagisment project Exam Help

• Is it the case that *every* NP problem is also in P??

https://powcoder.com

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists.

As stagisment Project Exam Help

- \bullet Is it the case that every NP problem is also in P??
- For example, is it possible that finding out if one of 2^n possible evaluations of n propositional variables of a propositional formula Φ makes such formula true is not much harder than simply checking if a given particular evaluation of these propositional variables makes Φ true?

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists

Assignment Project Exam Help

- Is it the case that every NP problem is also in P??
- For example, is it possible that finding out if one of 2^n possible evaluations of n propositional variables of a propositional formula Φ makes such formula true is not much harder than simply checking if a given particular evaluation of these propositional variables makes Φ true?
- Intuitively, this should not be the case; determining if such evaluation exists should be a harder whole than short, choose we will a latter makes such a formula true.

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists

Assignment Project Exam Help

- Is it the case that every NP problem is also in P??
- For example, is it possible that finding out if one of 2^n possible evaluations of n propositional variables of a propositional formula Φ makes such formula true is not much harder than simply checking if a given particular evaluation of these propositional variables makes Φ true?
- Intuitizely, this should not be the case; determining if such evaluation exists should be a further motion than should check the way of a formula true.
- However, so far, no one has been able to prove (or disprove) that this is indeed the case, despite a huge effort of very many very famous people!!

• As we have mentioned, for example, the decision problem "integer n is not prime" is obviously in NP, but it has been proved in 2002 that it is also in P. (This is a famous and unexpected result, proved by Indian computer scientists

Assignment Project Exam Help

- Is it the case that every NP problem is also in P??
- For example, is it possible that finding out if one of 2^n possible evaluations of n propositional variables of a propositional formula Φ makes such formula true is not much harder than simply checking if a given particular evaluation of these propositional variables makes Φ true?
- Intuitively, this should not be the case; determining if such evaluation exists should be a faith than should cheen the latest makes such a formula true.
- However, so far, no one has been able to prove (or disprove) that this is indeed the case, despite a huge effort of very many very famous people!!
- Conjecture that NP is a strictly larger class of decision problems than P is known as " $P \neq NP$ " hypothesis, and it is widely believed that it is one of the hardest open problems in the whole of Mathematics!!

• Let U and V be two decision problems. We say that U is polynomially reducible to V if and only if there exists a function f(x) such that:

Assignment Project Exam Help

https://powcoder.com

• Let U and V be two decision problems. We say that U is polynomially reducible to V if and only if there exists a function f(x) such that:

Assignment Project Exam Help

https://powcoder.com

• Let U and V be two decision problems. We say that U is polynomially reducible to V if and only if there exists a function f(x) such that:

As f(x) maps instances of U into instances of V:

As f(x) maps instance of f(x) maps instances of f(x) maps instance of f(x) maps instances of f(x) maps instances of f(x) maps instances of f(x) maps instances of f(x) maps instance of f(x) maps instances of f(x) maps instances of f(x) maps instance of f(x) maps instances of f(x) maps instance of

https://powcoder.com

- Let U and V be two decision problems. We say that U is polynomially reducible to V if and only if there exists a function f(x) such that:
 - \bullet f(x) maps instances of U into instances of V:

Assignment Project (x Exam defelp

3 f(x) is computable by a polynomial time algorithm.

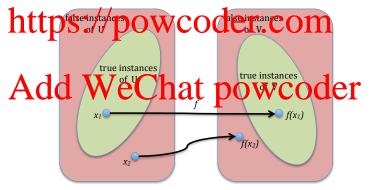
https://powcoder.com

• Let U and V be two decision problems. We say that U is polynomially reducible to V if and only if there exists a function f(x) such that:

A S S for every nearest x of y into instances of y:

The property of th

3 f(x) is computable by a polynomial time algorithm.



Example of a polynomial reduction:

Assignment Project Exam Help

https://powcoder.com

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

Assignment Project Exam Help

https://powcoder.com

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

As rejugation of several clauses.

https://powcoder.com

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

As signification of several clauses.

• For example, we replace clause

https://powcoder.com

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

Assignment of Broiled reconstruction of several clauses.

• For example, we replace clause

https://powcoder.com
with the following conjunction of "chained" 3-clauses with new propositional variables Q_1, Q_2, Q_3 :

Add We hat powcoder (2)

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

As spin on the property of the configuration of several clauses.

• For example, we replace clause

https://powcoder.com
with the following conjunction of "chained" 3-clauses with new propositional variables Q_1, Q_2, Q_3 :

Add Wethat powcoder (2)

• Easy to verify that if an evaluation of P'_is makes (??) true, then such evaluation can be extended by an evaluation of Q'_js such that (??) is also true

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

As spin of several clauses.

• For example, we replace clause

https://powcoder.com
with the following conjunction of "chained" 3-clauses with new propositional variables Q_1, Q_2, Q_3 :

(1)

Add Wethat powcoder (2)

- Easy to verify that if an evaluation of P'_is makes (??) true, then such evaluation can be extended by an evaluation of Q'_js such that (??) is also true
- ullet and vice versa: every evaluation which makes $(\ref{eq:condition})$ true also makes $(\ref{eq:condition})$ true.

Example of a polynomial reduction:

Every instance of SAT is polynomially reducible to an instance of 3SAT.

As seignamentosi Project recxemuse Help

• For example, we replace clause

https://powcoder.com
with the following conjunction of "chained" 3-clauses with new propositional variables Q_1, Q_2, Q_3 :

Add We Chat powcoder (2)

- Easy to verify that if an evaluation of P'_is makes (??) true, then such evaluation can be extended by an evaluation of Q'_js such that (??) is also true
- and vice versa: every evaluation which makes (??) true also makes (??) true.
- Clearly, (??) can be obtained from (??) using a simple polynomial time algorithm.

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

Assignment Project Exam Help

https://powcoder.com

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

As this means that for every podecision problem (72) there exists apply noticial problem (12) there exists apply noticial problem (13) there exists apply noticinal pr

https://powcoder.com

 \bullet Theorem: Every NP problem is polynomially reducible to the SAT problem.

As this means that for every Podecision problem (7) there exists applynomial Problem (12) there exists applynomial Problem (13) there exists apply

① for every instance x of U, f(x) is a propositional formula Φ_x ;

https://powcoder.com

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

As Significant Project Exam Help

- for every instance x of U, f(x) is a propositional formula Φ_x ;
- ② U(x) is true just in case $\Phi(x)$ has an evaluation of its propositional variables which makes Φ_x true.

 1. **The control of the propositional variables which makes Φ_x true.

 1. **The control of the propositional variables which makes Φ_x true.

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

ASSA English Curton from the CT Exam Help

- for every instance x of U, f(x) is a propositional formula Φ_x ;
- variables which makes Φ_x true. polynomially reducible to U(x).

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

AS This means that for every Podecision problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists apply no inial English Extension problem (7) there exists a positive exists a positive exists and the extension problem (7) the extension problem (7) there exists a positive exist a positive exists a positive exists a positive exist and the exist are exists and the exist are exists and the exist and the exist and the exist are exists and the exist and the exist and the exist are exists and the exist and the exist are exists and the exist are exists and the exist are exists and the exist and the exist are exists and the exist

- for every instance x of U, f(x) is a propositional formula Φ_x ;
- variables which makes Φ_x true.

 An NPLLisch Problem DC iNFC on Center Cthrisp problem is polynomially reducible to U(x).
- Thus, Cook's Theorem says that SAT is NP complete.

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

This means that for every pedecision problem (3) there exists apply nothing the Europe Exam Hell

- for every instance x of U, f(x) is a propositional formula Φ_x ;
- variables which makes Φ_x true.

 An NPLLisch Problem DC iNFC on Center Cthrisp problem is polynomially reducible to U(x).
- Thus, Cook's Theorem says that SAT is NP complete.
- NP complete problems are if a sense universal of we had an absorithm that solves one single NP complete problem U, then we could use such an algorithm to solve every other NP problem:

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

As the means that for every peccision problem (2) there exists appropriate the period of the period

- **1** For every instance x of U, f(x) is a propositional formula Φ_x ;
- ② U(x) is true just in case $\Phi(x)$ has an evaluation of its propositional variables which makes Φ_x true.
- variables which makes Φ_x true.

 An NPLL is on problem D(x) in D(x) in D(x) and D(x) is polynomially reducible to U(x).
- Thus, Cook's Theorem says that SAT is NP complete.
- NP complete problems are if a sense universal of we had an absorithm that solves one single NP complete problem U, then we could use such an algorithm to solve every other NP problem:
- A solution of an instance x of any other NP problem V could simply be obtained by:

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

As this means that for every pecision problem (2) there exists appropriate the period of the period

- **1** For every instance x of U, f(x) is a propositional formula Φ_x ;
- ② U(x) is true just in case $\Phi(x)$ has an evaluation of its propositional variables which makes Φ_x true.
- variables which makes Φ_x true.

 An NPLL is on problem D(x) in D(x) in D(x) and D(x) is polynomially reducible to U(x).
- Thus, Cook's Theorem says that SAT is NP complete.
- NP complete problems are if a sense infivered rift we had an absorithm that solves one single NP complete problem U, then we could use such an algorithm to solve every other NP problem:
- A solution of an instance x of any other NP problem V could simply be obtained by:
 - ① computing in polynomial time the reduction f(x) of V to U,

◆□▶ ◆□▶ ◆■▶ ◆■▶ ● のQで

• Theorem: Every NP problem is polynomially reducible to the SAT problem.

As This means that for every Podecision problem (7) there exists applying his Solidary Help

- for every instance x of U, f(x) is a propositional formula Φ_x ;
- ② U(x) is true just in case $\Phi(x)$ has an evaluation of its propositional variables which makes Φ_x true.
- variables which makes Φ_x true.

 An NPLL is on problem D(x) in D(x) in D(x) and D(x) is polynomially reducible to U(x).
- Thus, Cook's Theorem says that SAT is NP complete.
- NP complete problems are if a sense universal of we had an absorithm that solves one single NP complete problem U, then we could use such an algorithm to solve every other NP problem:
- A solution of an instance x of any other NP problem V could simply be obtained by:
 - ① computing in polynomial time the reduction f(x) of V to U,
 - 2 then running the algorithm that solves U on instance f(x).

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

Assignment Project Exam Help

https://powcoder.com

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

Ssignment Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

https://powcoder.com

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

Assignment Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

• So whttps://powcoder.com

• So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

signment Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

• so whttps://powcoder.com

Maybe SAT is not that important - why should we care about satisfiability of

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

SSIGNMENT Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

• so whttps://powcoder.com

 $\hbox{ Maybe SAT is not that important - why should we care about satisfiability of propositional formulas? } \hbox{ \begin{tabular}{c} \end{tabular} }$

• Maybe NP complete problems only have theoretical significance and no practical relevance??

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

SSIGNMENT Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

• so whttps://powcoder.com

 $\hbox{ Maybe SAT is not that important - why should we care about satisfiability of propositional formulas? } \hbox{ \begin{tabular}{c} \end{tabular} }$

- Maybe NP complete problems only have theoretical significance and no practical relevance??
- Unfortunately, this cannot be farthest from the truth!

Polynomial Reductions

 So NP complete problems are the hardest NP problems - a polynomial time algorithm for solving an NP complete problem would make every other NP problem also solvable in polynomial time.

Ssignment Project Exam Help cannot be any polynomial time algorithms for solving an NP complete problem, not even an algorithm that runs in time $O(n^{1,000,000})$.

• so whttps://powcoder.com

- Maybe NP complete problems only have theoretical significance and no practical relevance??
- Unfortunately, this cannot be farthest from the truth!
- A vast number of practically important decision problems are NP complete!

• Traveling Salesman Problem

Assignment Project Exam Help

https://powcoder.com

• Traveling Salesman Problem

Assignment Project Exam Help

https://powcoder.com

• Traveling Salesman Problem

Assignment Project Exam Help

a map, i.e., a weighted graph with locations as vertices and with edges connecting these vertices which represent roads connecting these locations and with the weights of these edges representing the length of these local WCOCCI.COM

• Traveling Salesman Problem

Assignment Project Exam Help

a map, i.e., a weighted graph with locations as vertices and with edges connecting these vertices which represent roads connecting these locations and with the weights of these edges representing the location of the locati

 \bullet a number L.

• Traveling Salesman Problem

- a map, i.e., a weighted graph with locations as vertices and with edges connecting these vertices which represent roads connecting these locations and with the weights of these edges representing the lengths of the bloods. COCI. COM
 - $oldsymbol{2}$ a number L.
- 2 Problem: Is there a tour along the edges visiting each location (i.e., vertex) exactly once and returning to the starting location such that the call length of the tarting at now to the location of the control of the location (i.e., vertex) exactly once and returning to the starting location such that the

• Traveling Salesman Problem

- a map, i.e., a weighted graph with locations as vertices and with edges connecting these vertices which represent roads connecting these locations and with the weights of these edges representing the length of these locations.
 - 2 a number L.
- 2 Problem: Is there a tour along the edges visiting each location (i.e., vertex) exactly once and returning to the starting location such that the callergth of the tental power of the power of the control of the cont
- ullet Think of a mailman which has to deliver mail to several addresses and then return to the post office. Can he do it while traveling less than L kilometres in total?

Assignment Project Exam Help

https://powcoder.com

Assignment Project Exam Help

https://powcoder.com

Assignment Project Exam Help

lacksquare A graph G with vertices of the graph representing program

https://powcoder.com

Assignment Project Exam Help

• A graph G with vertices of the graph representing program

variables; / f h gra Wideting that the Co fra les corresponding to the vertices of that edge are both needed at the same step of the execution of the program;

Assignment Project Exam Help

• A graph G with vertices of the graph representing program

corresponding to the vertices of that edge are both needed at the same step of the execution of the program;

3 The number of registers K of the processor.

- lacksquare A graph G with vertices of the graph representing program
- corresponding to the vertices of that edge are both needed at the same step of the execution of the program;
 - 3 The number of registers K of the processor.
- 2 Poblem s it possible to assign wrights to registers of a prefede has both vertices assigned to the same register.

- ${\bf 0}\;$ A graph G with vertices of the graph representing program
- corresponding to the vertices of that edge are both needed at the same step of the execution of the program;
 - 3 The number of registers K of the processor.
- 2 Mobile nos it possible to assign variables to register so the edge has both vertices assigned to the same register.
- ullet In graph theoretic terms: Is it possible to color the vertices of a graph G with at most K colors so that no edge has both vertices of the same color.

Assignment Project Exam Help

https://powcoder.com

Assignment Project Exam Help

 Assume you want to buy DVDs, each with one out of N movie that you like.

https://powcoder.com

Assignment Project Exam Help

- ullet Assume you want to buy DVDs, each with one out of N movie that you like.
- Unfortunately, the store does not sell individual DVDs, but only sells builds of DVDs, the CarVM bandles and bundle B_i has price p_i .

- Assume you want to buy DVDs, each with one out of N movie that you like.
- Unfortunately, the store does not sell individual DVDs, but only sells builded in VDs; the CrVV builded bundle B_i has price p_i .
- Every movie which you want to buy is in at least one of such bundles; some bundles may contain several movies that you want to buy.
- For each bundle R, you have a list l_i of all movies in that bundle. Add WeChat powcoder

- ullet Assume you want to buy DVDs, each with one out of N movie that you like.
- Unfortunately, the store does not sell individual DVDs, but only sells buildes in VDs; the CrVV builds and bundle B_i has price p_i .
- Every movie which you want to buy is in at least one of such bundles; some bundles may contain several movies that you want to buy.
- For each bundle B_i you have a list l_i of all movies in that bundle.
- Aurgout is to those set of builder y high the effect on an 11 of the movies you want to buy and which has the smallest total cost.

- \bullet Assume you want to buy DVDs, each with one out of N movie that you like.
- Unfortunately, the store does not sell individual DVDs, but only sells builded in VDs; the VDs; the VDs is VDs.
- Every movie which you want to buy is in at least one of such bundles; some bundles may contain several movies that you want to buy.
- For each bundle B_i you have a list l_i of all movies in that bundle.
- Aur gour is the voice a set of builder y not tygether on an all of the movies you want to buy and which has the smallest total cost.
- As we will see, many other practically important problems are NP complete.

• Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).

Assignment Project Exam Help

https://powcoder.com

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means A besides the usual computational steps, can also consult a "black box" for solving problem A(y) for any intermediate output y.

https://powcoder.com

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means S up this virice in the lengt X of the first put at light A besides the usual computational steps, can also consult a "black box" for solving problem A(y) for any intermediate output y.
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.e., if we can solve every NP problem U sing a polynomial time algorithm which can also use a black box to some only instance of A

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means A if A in the length of the besides the usual computational steps, can also consult a "black box" for solving problem A(y) for any intermediate output y.
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.e. if we can solve every NP problem U sing a polynomial time algorithm which can also use a black box to some only instance of A
- Note that we do NOT require that A be an NP problem; we even do not require it to be a decision problem it can also be an optimisation problem.

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means S up to the property of the length of the
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.d. if we can solve every NP problem U sing a polynomial time algorithm which can also use a black box to solve any interaction of A
- Note that we do NOT require that A be an NP problem; we even do not require it to be a decision problem it can also be an optimisation problem.
- Example: Add We Chat powcoder

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means S up to the property of the length of the
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.e., if we can solve every NP problem U sing a polynomial time algorithm which can also use a black box to some only instance of A
- Note that we do NOT require that A be an NP problem; we even do not require it to be a decision problem it can also be an optimisation problem.
- Example: (The Traveling Saleshin Optimisation Problem)
 Instanted weighter graph (Lidas of Dearlons) Cit Original representing the lengths of the edges of the graph (roads between locations);

- Let A be a problem and assume that we have a 'black box' device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means S Sgretnifs in the lengt to the lengt to the pout and the besides the usual computational steps, can also consult a "black box" for solving problem A(y) for any intermediate output y.
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.e. if we can solve every NP problem U sing a polynomial time algorithm which can also use a blad box to solve any instance of NP instance.
- Note that we do NOT require that A be an NP problem; we even do not require it to be a decision problem it can also be an optimisation problem.
- Example: (The Traveling Saleshila Optimisation Problem)
 Instance A weighter graph (1 day of Dearlow) Lith Weight representing the lengths of the edges of the graph (roads between locations);
 - Problem: Find a cycle in the graph containing all vertices which is of minimal possible total length.

- Let A be a problem and assume that we have a 'black box" device (also called "an oracle") which for every input x instantaneously computes A(x).
- We consider algorithms which are polynomial time in A. This means S Spritting in the lengt X of the final condition of the lengt X of the final condition besides the usual computational steps, can also consult a "black box" for solving problem A(y) for any intermediate output y.
- We say that a problem A is NP hard if every NP problem is polynomial time in A, i.e. if we can solve every NP problem U sing a polynomial time algorithm which can also use a black box to some only instance of A
- Note that we do NOT require that A be an NP problem; we even do not require it to be a decision problem - it can also be an optimisation problem.
- Example: (The Traveling Saleshin Optimisation Problem)
 Instanted weighter graph (Lotations) with weighter representing the lengths of the edges of the graph (roads between locations);
 - Problem: Find a cycle in the graph containing all vertices which is of minimal possible total length.
- Think of a mailman having to deliver mail to several addresses while having to travel as small total distance as possible.

As The Traveling Salesman Opin is the Problem is Clearly NP hard: Help becision problem: Decision problem:

- Given a weighted graph G and a number L we can determine if there is a cycle containing all vertices of the graphy and whose length is at most L.
- We do that by solving the Traveling Salesman Optimisation Problem thus determining the length of the cycle of minimal possible length and comparing the length of such a cycle with L.
- Since Al order NP wablens are polynomial time returble to the Traveling Salesman Decision problem (which is NP complete), then every other NP problem is solvable using a "black box" for the Traveling Salesman Optimisation Problem.

• It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e., Assignifine Project Exam Help)

https://powcoder.com

• It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e., spokenial time) chair. Project Exam Help So west do we do when we encounted an NP hard problem?

https://powcoder.com

• It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e.,

SSI Print Project Exam Help

• If this problem is an optimisation problem, we can try to solve such a problem in an approximate sense by finding a solution which might not be optimal, but it is properly solved to the properly of the company of t

• It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e.,

SSI grid in Chit. Project Exam Help
So west do we do when we encount an NP hard problem?

• If this problem is an optimisation problem, we can try to solve such a problem in an approximate sense by finding a solution which might not be optimal, but it is passive to the primal solution. C1.

• For example, in the case of the Traveling Salesman Optimisation Problem we might look for a tour which is not longer than twice the length of the shortest possible tour.

It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e., splinging time) which. Project Exam Help So was do we do when we encounted an NP hard problem?

- If this problem is an optimisation problem, we can try to solve such a problem in an approximate sense by finding a solution which might not be optimal, but it is east at a close to an entire order.
- For example, in the case of the Traveling Salesman Optimisation Problem we might look for a tour which is not longer than twice the length of the shortest possible tour.
- Thus Ar partical voltage who are are the would be:

It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e., splinging time) chair. Project Exam Help So west do we do when we encounted an NP hard problem?

• If this problem is an optimisation problem, we can try to solve such a problem in an approximate sense by finding a solution which might not be optimal, but it is east an Cose to make the print of the contract of the con

• For example, in the case of the Traveling Salesman Optimisation Problem we might look for a tour which is not longer than twice the length of the shortest possible tour.

• Thus, Ar Charlica Wolfe who a fear Do Wy Cle or test would be:

• prove that the problem is indeed NP hard, to justify not trying solving the problem exactly;

• It is important to be able to figure out if a problem at hand is NP hard in order to know that one has to abandon trying to come up with a feasible (i.e.,

S So was do we do when we encount of an NP hard problem? Help

• If this problem is an optimisation problem, we can try to solve such a problem in an approximate sense by finding a solution which might not be optimal, but it is passive to the primal solution. C1.

• For example, in the case of the Traveling Salesman Optimisation Problem we might look for a tour which is not longer than twice the length of the shortest possible tour.

• Thus, Ar Chartica Wolfer who are an Ob Wyo Clerch test would be:

- prove that the problem is indeed NP hard, to justify not trying solving the problem exactly;
- look for an approximation algorithm which provides a feasible sub-optimal solution that it is not too bad.

4日 → 4日 → 4 目 → 4 目 → 9 Q ○

Warning: sometimes distinction between a problem in P and an NP complete problem can be subtle!

Assignment Project Exam Help

- Given a graph G and two vertices s and t, is there a path from s to t of length at most K?
- Given a propositional formula in CNF form such that every clause
- CNF form such that every clause has at most **two** propositional variables, toos the formula have a satisfying assignment?
- Given a graph G, does G have a tour where every **edge** is traversed exactly once? (An *Euler tour*.)

- Given a graph G and two vertices s and t, is there a simple path from s to t of length at least K?
- Given a propositional formula in CNF form such that every clause has at most **three** propositional variables closs the formula have a satisfying assignment?
- Given a graph G, does G have a tour where every **vertex** is visited exactly once? (A *Hamiltonian* cycle.)

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

Assignment Project Exam Help

https://powcoder.com

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

Assignment V be an NP complete problem, and let V be another NP Assignment V be another NP Complete Problem of the Problem of

https://powcoder.com

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

Theorem: Let U be an NP complete problem, and let V be another NP

Assignmental Problem V the E is also NP complete p

Proof. Assume that a(x) is a polymenial reduction of U to V, and let W be

• Proof: Assume that g(x) is a polynomial reduction of U to V, and let W be any other NP problem.

https://powcoder.com

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

Theorem: Let *U* be an NP complete problem, and let *V* be another NP ASSIGNMENT Problem of the Problem of the

- Proof: Assume that g(x) is a polynomial reduction of U to V, and let W be any other NP problem.
- Since U is NP complete, there exists a polynomial reduction f(x) of W to U.

 **Proposition of the complete of the c

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

• Theorem: Let U be an NP complete problem, and let V be another NP signification of the Property of the Exam collection of the Exam col

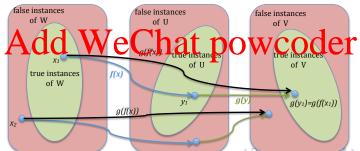
- Proof: Assume that q(x) is a polynomial reduction of U to V, and let W be any other NP problem.
- Since U is NP complete, there exists a polynomial reduction f(x) of W to U.
 We can that g(f(x)) is polynomial reduction of U.

Taking for granted that SAT is NP complete, how do we prove NP completeness of another NP problem??

• Theorem: Let U be an NP complete problem, and let V be another NP

signification problem of the Exam complete Signification of the Exam complete Signific • Proof: Assume that q(x) is a polynomial reduction of U to V, and let W be

- any other NP problem.
- Since U is NP complete, there exists a polynomial reduction f(x) of W to U.
 We call the g(f(x)) is a polynomial collection of the collecti



• g(f(x)) is a polynomial time reduction of W to V because:

Assignment Project Exam Help

https://powcoder.com

- g(f(x)) is a polynomial time reduction of W to V because:
- Assignment Project Exam Help

https://powcoder.com

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true; Assignment Project Exam Help

https://powcoder.com

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true; Assignment Project Exam Help
 - Thus W(x) is true just in case V(g(f(x))) is true, i.e., g(f(x)) is a
 - https://powcoder.com

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true;

 Assignment Project Exame Help
 - Thus W(x) is true just in case V(g(f(x))) is true, i.e., g(f(x)) is a reduction of W to V.
 - Finter f(x) is the operator polynomial time computable function, the rength f(x) of the output f(x) can be at most a polynomial in |x|, i.e., for some polynomial (with positive coefficients) P we have $|f(x)| \leq P(|x|)$.

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true;

 Assignment Project Exame Help
 - Thus W(x) is true just in case V(g(f(x))) is true, i.e., g(f(x)) is a reduction of W to V.
 - Finte f(x) is the property polynomial time computable function, the length f(x) of the output f(x) can be at most a polynomial in |x|, i.e., for some polynomial (with positive coefficients) P we have $|f(x)| \leq P(|x|)$.
 - since g(y) is polynomial time computable as well, there exists a polynomial Q with that for elect input to involve the first erminates after at most Q(|y|) many steps.

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true;

 Assignment Project Exame Help
 - Thus W(x) is true just in case V(g(f(x))) is true, i.e., g(f(x)) is a reduction of W to V.
 - Finte f(x) is the output of a polynomial time computable function, the rength f(x) of the output f(x) can be at most a polynomial in |x|, i.e., for some polynomial (with positive coefficients) P we have $|f(x)| \le P(|x|)$.
 - since g(y) is polynomial time computable as well, there exists a polynomial Q with that for peal input Q multiplies Q reminates after at most Q(|y|) many steps.
 - Thus, the computation of g(f(x)) terminates in at most P(|x|) many steps (computation of f(x)) plus $Q(|f(x)|) \leq Q(P(|x|))$ many steps (computation of g(y) for y = f(x)).

- g(f(x)) is a polynomial time reduction of W to V because:
- since f is a reduction of W to U, W(x) is true just in case U(f(x)) is true;

 Assignment Project Exame Help
 - Thus W(x) is true just in case V(g(f(x))) is true, i.e., g(f(x)) is a reduction of W to V.
 - Since f(x) is the property polynomial time computable function, the length |f(x)| of the output f(x) can be at most a polynomial in |x|, i.e., for some polynomial (with positive coefficients) P we have $|f(x)| \le P(|x|)$.
 - since q(y) is polynomial time computable as well, there exists a polynomial Q with that for each north Q in the formula Q is a first at most Q(|y|) many steps.
 - Thus, the computation of g(f(x)) terminates in at most P(|x|) many steps (computation of f(x)) plus $Q(|f(x)|) \leq Q(P(|x|))$ many steps (computation of g(y) for y = f(x)).
 - In total, the computation of g(f(x)) terminates in at most P(|x|) + Q(P(|x|)) many steps, which is a polynomial bound in |x|.

 \bullet Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:

Assignment Project Exam Help

https://powcoder.com

- \bullet Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges

Assignment Project Exam Help

https://powcoder.com

- \bullet Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- Assignment value of exempt what plable of the second with a superior of the second with a superi

https://powcoder.com

- Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus
 proving that VC is NP complete:
 - for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- Assignment value of connecting these vertices of the connecting these vertices of the connecting these vertices of the connection variable of the connection variables of
 - 3 for each clause C_i connect the three corresponding vertices v_1^i, v_2^i, v_3^i with one end of the three segments corresponding to the variable appearing in C_i ;

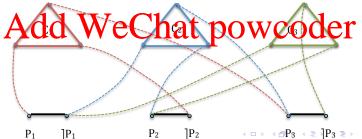
https://powcoder.com

- Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - **1** for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- SSI Philipping value of the second with a real label of personal ways are a label of personal with a real label of personal with a real label of personal ways are a label of personal with a real label of personal ways are a label of personal ways a
 - \bullet for each clause C_i connect the three corresponding vertices v_1^i, v_2^i, v_3^i with one end of the three segments corresponding to the variable appearing in C_i ;
 - $\begin{picture}(100,0) \put(0,0){\line(1,0){100}} \put(0,0){\line(1,0){10$

- Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - **1** for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- SSI Entre value of the company of th
 - \bullet for each clause C_i connect the three corresponding vertices v_1^i, v_2^i, v_3^i with one end of the three segments corresponding to the variable appearing in C_i ;
 - if the variable appears with a negation sign, connect it with the end PPN M POWOUGET.COM
 - otherwise connect it with the end labeled with that letter:

- \bullet Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- Assignmenting these vertices of the segment with a right lab let ellip
 - \bullet for each clause C_i connect the three corresponding vertices v_1^i, v_2^i, v_3^i with one end of the three segments corresponding to the variable appearing in C_i ;
 - httable with the end with the e
 - otherwise connect it with the end labeled with that letter;

- ullet Reducing an instance of 3SAT to an instance of a Vertex Cover (VC) problem, thus proving that VC is NP complete:
 - for each clause C_i draw a triangle with three vertices v_1^i, v_2^i, v_3^i and three edges
- Assignating these vertices of the segment with a rice lab let ellip
 - 3 for each clause C_i connect the three corresponding vertices v_1^i, v_2^i, v_3^i with one end of the three segments corresponding to the variable appearing in C_i ;
 - if the variable appears with a negation sign, connect it with the end
 - otherwise connect it with the end labeled with that letter;



 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

Assignment Project Exam Help

https://powcoder.com

P1 | P2 | P2 | P2 | P3 | P3 | P3 |

Add WeChat powcoder

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

Assignment Project Exam Help https://powcoder.com P1 | P2 | P2 | P3 | P3 | P3 | P3

• Claim An in time of W. Technisting of M claims and which makes that instance true if and only if the corresponding graph has a Vertex Cover of size at most 2M+N.

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

- Claim An in tance on N Technisting of M claims and continuing in $\operatorname{Col} N$ propositional variables has an assignment of variables which makes that instance true if and only if the corresponding graph has a Vertex Cover of size at most 2M+N.
- Assume there is a vertex cover with at most 2M + N vertices chosen. Then

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

- Claim An in time of W. Technisting of M claims and containing it $\operatorname{Cont} N$ propositional variables has an assignment of variables which makes that instance true if and only if the corresponding graph has a Vertex Cover of size at most 2M + N.
- Assume there is a vertex cover with at most 2M + N vertices chosen. Then
 - Each triangle must have at least two vertices chosen;

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

- Claim An in time of W. Technisting of M claims and containing it $\operatorname{Cont} N$ propositional variables has an assignment of variables which makes that instance true if and only if the corresponding graph has a Vertex Cover of size at most 2M + N.
- Assume there is a vertex cover with at most 2M + N vertices chosen. Then
 - 1 Each triangle must have at least two vertices chosen;
 - 2 Each segment must have at least one of its ends chosen.

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

- Claim An in time of W. Technisting of M claims and containing it $\operatorname{Cont} N$ propositional variables has an assignment of variables which makes that instance true if and only if the corresponding graph has a Vertex Cover of size at most 2M + N.
- Assume there is a vertex cover with at most 2M + N vertices chosen. Then
 - Each triangle must have at least two vertices chosen;
 - 2 Each segment must have at least one of its ends chosen.
- This is in total 2M + N points; thus each triangle must have exactly two vertices chosen and each segment must have exactly one of its ends chosen.

 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

Assignment Project Exam Help https://powcoder.com

tional letter C_t to true if Pend of the segment corresponding to P_i is covered;

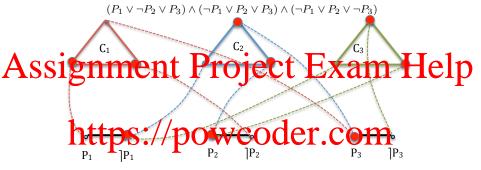
 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

- Set each propositional letter P_i to true if P_i end of the segment corresponding to P_i is covered;
- Otherwise, set a propositional letter P_i to false if $\neg P_i$ is covered,

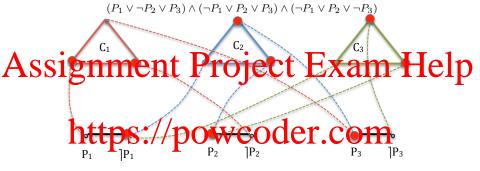
 $(P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee P_3) \wedge (\neg P_1 \vee P_2 \vee \neg P_3)$

Assignment Project Exam Help https://powcoder.com

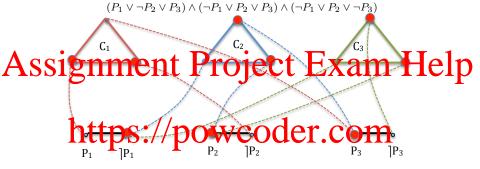
- Set each propositional letter P_i to true if P_i end of the segment corresponding to P_i is covered;
- Otherwise, set a propositional letter P_i to false if $\neg P_i$ is covered,
- In a vertex cover of such a graph every uncovered vertex of each triangle must be connected to a covered end of a segment, which guarantees that the clause corresponding to each triangle is true.



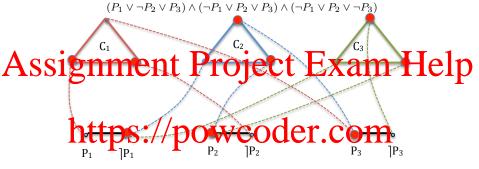
• Opposite, assume that the formula has an assignment of the variables which makes it true.



- Opposite, assume that the formula has an assignment of the variables which makes it
- For each segmen, if iver spends of a repositional type P_i with such a satisfying evaluation sets to true cover its P_i end.

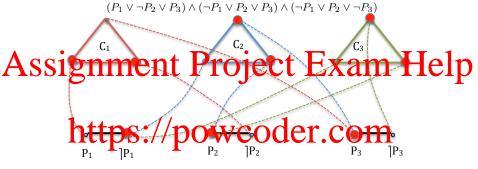


- Opposite, assume that the formula has an assignment of the variables which makes it true.
- For each segmen, if iver responds to a propositional layer P_i with step a satisfying evaluation sets to true cover its P_i end.
- Otherwise, if a propositional letter P_i is set to to false by the satisfying evaluation, cover its $\neg P_i$ end.



- Opposite, assume that the formula has an assignment of the variables which makes it true.
- For each segmen, if it corresponds to a propositional layer P_i with step a satisfying evaluation sets to true cover its P_i end.
- Otherwise, if a propositional letter P_i is set to to false by the satisfying evaluation, cover its ¬P_i end.
- For each triangle corresponding to a clause at least one vertex must be connected to a covered end of a segment, namely to the segment corresponding to the variable which makes that clause true; cover the remaining two vertices of the triangle.

Reducing 3SAT to VC



- Opposite, assume that the formula has an assignment of the variables which makes it true.
- For each segmen, if it corresponds to a propositional layer P_i with step a satisfying evaluation sets to true cover its P_i end.
- Otherwise, if a propositional letter P_i is set to to false by the satisfying evaluation, cover its ¬P_i end.
- For each triangle corresponding to a clause at least one vertex must be connected to a covered end of a segment, namely to the segment corresponding to the variable which makes that clause true; cover the remaining two vertices of the triangle.
- in this way we cover exactly 2M+N vertices of the graph and clearly every edge between a segment and a triangle has at least one end covered.

Assignment Project Exam Help

https://powcoder.com

• If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.

Assignment Project Exam Help

https://powcoder.com

• If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.

S Sange Mre Core has an Ipprovious of algorithm Animalways colors a covering set with at most twice the number of the smallest vertex cover.

https://powcoder.com

• If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.

S School Mre Cop has an Ipor in Cot algorith. And Illustrates a covering set with at most twice the number of the smallest vertex cover.

• Algorithm: (somewhat counter intuitive!)

https://powcoder.com

- If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.
- A Scanged Wreke for has an Ipon sina tortal prith and law we bed as a covering set with at most twice the number of the smallest vertex cover.
 - Algorithm: (somewhat counter intuitive!)
 - https://powcoder.com

- If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.
- A Sanger Wrecher has an ipon situation algorithm the limit ways believed a covering set with at most twice the number of the smallest vertex cover.
 - Algorithm: (somewhat counter intuitive!)

covered.

- Pick an arbitrary edge and cover BOTH of its ends.

 Rendy Dishe edges the wife and show to the only with edges which have either both ends covered or no end
 - Add WeChat powcoder

- If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.
- Scanged Wret Coler has an approximation algorithm and always both as a covering set with at most twice the number of the smallest vertex cover.
 - Algorithm: (somewhat counter intuitive!)
 - Pick an arbitrary edge and cover BOTH of its ends.
 - Removed the edges which have either both ends covered or no end covered.
 - 3 Repeat picking edges with both ends uncovered until no edges are left.

- If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.
- Scannel Vice Coler has an ipproximation algorithm the laws both des a covering set with at most twice the number of the smallest vertex cover.
 - Algorithm: (somewhat counter intuitive!)
 - Pick an arbitrary edge and cover BOTH of its ends.
 - 2 Replay 15 he/edgs (bby need show toy red if his way you are left only with edges which have either both ends covered or no end covered.
 - 3 Repeat picking edges with both ends uncovered until no edges are left.
 - Clear extlis reduces a vertex cover de ause lige we remoted my if one of their end is covered and we perform this procedure until no edges are left.

- If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.
- Scannel Virex Core has an ipproximation algorithm the laws both desa covering set with at most twice the number of the smallest vertex cover.
 - Algorithm: (somewhat counter intuitive!)
 - Pick an arbitrary edge and cover BOTH of its ends.
 - 2 Rendy 15 he/edgs (box/ne edg s htw foy red if this way you are left only with edges which have either both ends covered or no end covered.
 - 3 Repeat picking edges with both ends uncovered until no edges are left.
 - Clear Atlis foduciate entex cover because also we removed only if one of their end is covered and we perform this procedure until no edges are left.
 - The number of vertices covered is equal to twice the number of edges with both ends covered. But the minimal vertex cover must cover at least one vertex of each such edge.

• If an optimisation problem is NP hard, we do not try to solve it exactly, but instead, try to find a feasible (i.e., P time) algorithm which produces a solution that is not too bad.

Sanger Vice Cole has an ipor in ion algorith, airl always por a covering set with at most twice the number of the smallest vertex cover.

- Algorithm: (somewhat counter intuitive!)
 - Pick an arbitrary edge and cover BOTH of its ends. 2 Remove all the edges whose one end's now covered. In this way you are left only with edges which have either both ends covered or no end covered.
 - 3 Repeat picking edges with both ends uncovered until no edges are left.
- Clearly, this reduces a vertex cover because alges we removed only if one of their end is covered and we perform this procedure until no edges are left.
- The number of vertices covered is equal to twice the number of edges with both ends covered. But the minimal vertex cover must cover at least one vertex of each such edge.
- Thus we have produced a vertex cover of size at most twice the size of the minimal vertex cover.

• Example: Metric Traveling Salesman Problem (MTSP).

Assignment Project Exam Help

https://powcoder.com

- Example: Metric Traveling Salesman Problem (MTSP).
- Instance: A complete weighted graph G with weights d(i,j) of edges (to be interpreted as distances) satisfying the "triangle inequality": for any three ASSIGNMENT $\Pr_{d(i,j)+d(j,k) \geq d(i,k)} EXAM$ Help

https://powcoder.com

- Example: Metric Traveling Salesman Problem (MTSP).
- Instance: A complete weighted graph G with weights d(i,j) of edges (to be interpreted as distances) satisfying the "triangle inequality": for any three parameters $P_{d(i,j)} + Q_{j,k} \ge d(i,k)$ and $P_{d(i,j)} + Q_{d(i,k)}$

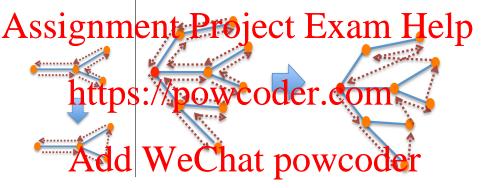
• Claim: MTSP has an approximation algorithm producing a tour of total length at men two length of the writing length of the writing

- Example: Metric Traveling Salesman Problem (MTSP).
- Instance: A complete weighted graph G with weights d(i,j) of edges (to be interpreted as distances) satisfying the "triangle inequality": for any three parameters $P_{d(i,j)} + P_{d(j,k)} \ge d(i,k)$ and $P_{d(i,k)} + P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three parameters $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality": for any three $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle inequality" in $P_{d(i,k)} = P_{d(i,k)}$ is a satisfying the "triangle" inequality".
 - Claim: MTSP has an approximation algorithm producing a tour of total length at most twice the length of the optimal minimal length four the noted by opt.
 - Algorithm: Ind a minimum spanning tree T of G. Since the optimal tour with one of its edges e removed represents a spanning tree, we have that the total weight of T satisfies $w(T) \leq opt w(e) \leq opt$.

- Example: Metric Traveling Salesman Problem (MTSP).
- Instance: A complete weighted graph G with weights d(i,j) of edges (to be interpreted as distances) satisfying the "triangle inequality": for any three project Exam Help
 - Claim: MTSP has an approximation algorithm producing a tour of total length at most twice the length of the optimal minimal length tour, handed by opt.
 - Algorithm: Ind a minimum spanning tree T of G. Since the optimal tour with one of its edges e removed represents a spanning tree, we have that the total weight of T satisfies $w(T) \leq opt w(e) \leq opt$.
 - If we depth instituyerse of the tree we will travel in total a distance of 2w(T)

- Example: Metric Traveling Salesman Problem (MTSP).
- Instance: A complete weighted graph G with weights d(i,j) of edges (to be interpreted as distances) satisfying the "triangle inequality": for any three project Exam Help
 - Claim: MTSP has an approximation algorithm producing a tour of total length at most when the length of the optimal minimal length tour, handed by opt.
 - Algorithm: Ind a minimum spanning tree T of G. Since the optimal tour with one of its edges e removed represents a spanning tree, we have that the total weight of T satisfies $w(T) \leq opt w(e) \leq opt$.
 - If we depth instituyerse of the tree we will travel in total edistance of 2w(T)





• We now take shortcuts to avoid visiting vertices more than once; because of the triangle inequality, this operation does not increase the length of the tour.

• As we have mentioned, all NP complete problems are in a sense equally difficult because any of them is reducible to any other via a polynomial time transformation.

Assignment Project Exam Help

https://powcoder.com

• As we have mentioned, all NP complete problems are in a sense equally difficult because any of them is reducible to any other via a polynomial time transformation.

size.

https://powcoder.com

- As we have mentioned, all NP complete problems are in a sense equally difficult because any of them is reducible to any other via a polynomial time
- ssignment Project Exam Help have seen, the Vertex Cover problem allows an approximation which produces a cover which is at most twice as large as the optimal cover of minimal possible size.
- on the the past, the poewa Godi exercionnen does not allow any approximate solution at all: if $P \neq NP$, then for no K > 0 there can be a polynomial time algorithm which for every instance produces a tour which is at most K times longer than the optimal tour of minimal possible length, no matter of the Richard Powcoder hat powcoder

• As we have mentioned, all NP complete problems are in a sense equally difficult because any of them is reducible to any other via a polynomial time transformation.

stransformation.

Share that call of the c

• On the dile Pand, the instruction at all: if $P \neq NP$, then for no K > 0 there can be a polynomial time algorithm which for every instance produces a tour which is at most K times longer than the optimal tour of minimal possible length, no matter tow large K is chosen!

• To prove this, we show that if there existed K>0 and a polynomial time algorithm producing a tour which is at most K times longer than the optimal tour, then we could obtain an algorithm which solves in polynomial time the Hamiltonian Cycle Problem, i.e., which for every graph G determines if G contains a cycle visiting all vertices exactly once, which is impossible because this problem is known to be NP complete.

ullet To see this, let G be an arbitrary graph with n vertices.

Assignment Project Exam Help

https://powcoder.com

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of ASSTICTION TOLECT EXAM Help

https://powcoder.com

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of Scrie indistrictive that the content of the

• We now apply the approximation algorithm (which we have assumed to exist) to produce a tour of all vertices of total length at most $K \cdot opt$ where opt is the length of the optimal tour through G^* .

https://powcoder.com

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of
- We now apply the approximation algorithm (which we have assumed to exist) to produce a tour of all vertices of total length at most $K \cdot opt$ where opt is the length of the optimal tour through G^* .
- Clearly, if the original graph Gauss Hamiltonian cycle there G has a tour consisting of edges already in G and of weights equal to 1, so such a tour has length of exactly n.

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of
- We now apply the approximation algorithm (which we have assumed to exist) to produce a tour of all vertices of total length at most $K \cdot opt$ where opt is the length of the optimal tour through G^* .
 - Clearly ff the original graph G thus a Hamiltonian cycle there G has a tour consisting of edges already in G and of weights equal to 1, so such a tour has length of exactly n.
- Otherwise, if the original graph G does not have a Hamiltonian cycle, then the optimal tout through G must defining at least one added edge of length $K \cdot n$. Add Welland DOWCOGET

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of
- We now apply the approximation algorithm (which we have assumed to exist) to produce a tour of all vertices of total length at most $K \cdot opt$ where opt is the length of the optimal tour through G^* .
 - Clearly if the original graph G thus a Hamiltonian cycle there G has a tour consisting of edges already in G and of weights equal to 1, so such a tour has length of exactly n.
- Otherwise, if the original graph G does not have a Hamiltonian cycle, then the optimal tout through G' must dentain at least one added edge of length $K \cdot n$.
- In this case the loptimal trun through Ct has length of at least $(K \cdot n) + (n-1) \cdot 1 > K \cdot n$.

- ullet To see this, let G be an arbitrary graph with n vertices.
- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of
- We now apply the approximation algorithm (which we have assumed to exist) to produce a tour of all vertices of total length at most $K \cdot opt$ where opt is the length of the optimal tour through G^* .
- Clearly if the original graph G thus a Hamiltonian cycle there G has a tour consisting of edges already in G and of weights equal to I, so such a tour has length of exactly n.
- Otherwise, if the original graph G does not have a Hamiltonian cycle, then the optimal tout through G' must deltain at least one added edge of length $K \cdot n$.
- In this distribution that the highest highest $K(K \cdot n) + (n-1) \cdot 1 > K \cdot n$.
- Thus, our approximation algorithm can return a tour of length at most $K \cdot n$ if an only if it actually returns a tour of length of size n, which happens just in case G has a Hamiltonian cycle.

ullet To see this, let G be an arbitrary graph with n vertices.

length of the optimal tour through G^* .

- We turn this graph into a complete weighted graph G^* by setting the weights of all existing edges to 1 and then add edges between the remaining pairs of
- Serie mention that the interpretation of the series of total length at most $K \cdot opt$ where opt is the
- Clearly if the original graph G in S a Hamiltonian cycle there G has a tour consisting of edges already in G and of weights equal to I, so such a tour has length of exactly n.
- Otherwise, if the original graph G does not have a Hamiltonian cycle, then the optimal tout through G' must dentain at least one added edge of length $K \cdot n$.
- In this distribution that the throught the highest Mf are the $(K \cdot n) + (n-1) \cdot 1 > K \cdot n$.
- Thus, our approximation algorithm can return a tour of length at most $K \cdot n$ if an only if it actually returns a tour of length of size n, which happens just in case G has a Hamiltonian cycle.
- This is impossible, because this would be a polynomial time decision procedure for determining in G has a Hamiltonian cycle.