COMP 3721

Tutorial 12

P

Definition

The class **P** consists of all decision problems (languages) that are solvable in polynomial time. That is, there exists an algorithm that decides in polynomial time if any given input is a yes-input or a no-input.

Theorem

P *is closed under complement, union, intersection, concatenation, and Kleene star.*

NP

Definition

A nondeterministic TM runs in polynomial time if for any input x, the number of steps of any computation path is $O(n^c)$, where c is a constant and $n = |x|$ is the input size. The class **NP** consists of all decision problems that can be decided by a nondeterministic TM in polynomial time.

Remark: **NP** stands for "nondeterministic polynomial time", not "non-polynomial"!

Theorem

NP *is closed under union, intersection, concatenation and Kleene star.*

NP

Theorem

A decision problem belongs to **NP** *iff for each yes-input, there exists a certificate which allows one to verify in polynomial time that the input is indeed a yes-input.*

P and NP

1) Show that regular languages are in ***P***.

P and NP

1) Show that regular languages are in ***P***.

P is the class of languages that can be **decided** by a **deterministic** Turing machine in **polynomial** time.

Idea:
DFA can be regarded as a special kind of Turing machine.

P and NP

2) Prove that ***P*** is closed under Kleene star.

Hints: Use DP.

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Let L be any language in ***P***. Let A be the polynomial time algorithm decides L.

We will use A to design a polynomial algorithm deciding L^*. Let $y_1 . . . y_n$ be the input string.

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Let L be any language in ***P***. Let A be the polynomial time algorithm decides L.

We will use A to design a polynomial algorithm deciding L^*. Let $y_1 \ldots y_n$ be the input string.

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Subproblems:

For each $1 \leq i \leq j \leq n$, we use $f(i, j)$ to indicate whether the substring $y_i. . . y_j$ is in L^*. If $y_i. . . y_j$ is in L^*, $f(i, j) = 1$; otherwise $f(i, j) = 0$.

Our goal is to compute $f(1, n)$.

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Subproblems:

For each $1 \leq i \leq j \leq n$, we use $f(i, j)$ to indicate whether the substring $y_i . . . y_j$ is in L^*. If $y_i . . . y_j$ is in L^*, $f(i, j) = 1$; otherwise $f(i, j) = 0$.

Our goal is to compute $f(1, n)$.

Recursive Formula:

To compute $f(i, j)$, we first check whether $y_i . . . y_j$ is in L by using A. If it is, we set $f(i, j) = 1$. Otherwise, we try every possible k between i and j, and see whether $y_i . . . y_k$ and $y_{k+1} . . . y_j$ are in L^*

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Subproblems:

For each $1 \le i \le j \le n$, we use $f(i, j)$ to indicate whether the substring $y_i. . . y_j$ is in L^*. If $y_i. . . y_j$ is in L^*, $f(i, j) = 1$; otherwise $f(i, j) = 0$.

Our goal is to compute $f(1, n)$.

Recursive Formula:

$$f(i,j) = \begin{cases} 1 & \text{if } y_i \ldots y_j \text{ is in } L \\ \max_{i \le k \le j-1} f(i,k) * f(k+1,j) & \text{otherwise} \end{cases}$$

P and NP

2) Prove that ***P*** is closed under Kleene star.

Proof(sketch):

Subproblems:

For each $1 \le i \le j \le n$, we use $f(i, j)$ to indicate whether the substring $y_i. . . y_j$ is in L^*. If $y_i. . . y_j$ is in L^*, $f(i, j) = 1$; otherwise $f(i, j) = 0$.

Our goal is to compute $f(1, n)$.

Recursive Formula:

$$f(i,j) = \begin{cases} 1 & \text{if } y_i \dots y_j \text{ is in } L \\ \max_{i \le k \le j-1} f(i,k) * f(k+1,j) & \text{otherwise} \end{cases}$$

More…(Base case.. Evaluation order.. Polynomial time..)

P and NP

3) Prove that graph isomorphism problem is in ***NP***.

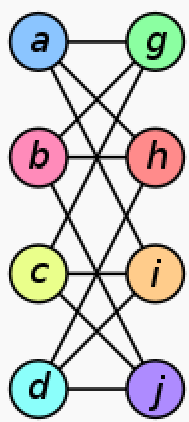
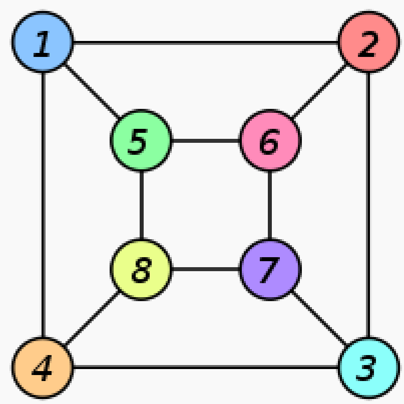
The graph isomorphism problem determining whether two finite graphs, G and H, are isomorphic.

In graph theory, an isomorphism of graphs G and H is a bijection between the vertex sets of G and H

such that *any* two vertices *u* and *v* of G are adjacent in G if and only if *f(u)* and *f(v)* are adjacent in H.

P and NP

3) Prove that graph isomorphism problem is in ***NP***.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

https://commons.wikimedia.org/wiki/File:Graph_isomorphism_a.svg#/media/File:Graph_isomorphism_a.svg

P and NP

3) Prove that graph isomorphism problem is in ***NP***.

The graph isomorphism problem determining whether two finite graphs, G and H, are isomorphic.

Theorem

A decision problem belongs to **NP** *iff for each yes-input, there exists a certificate which allows one to verify in polynomial time that the input is indeed a yes-input.*

P and NP

3) Prove that graph isomorphism problem is in ***NP***.

Proof(sketch):

Certificate: the bijection.

Verify in Polynomial time…