18.404/6.840 Lecture 14

(midterm replaced lecture 13)

Last time:

- TIME
- P
-

Today:

- NTIME
- NP
- P vs NP problem
- Dynamic Programming
- Polynomial-time reducibility

Posted:

- Midterm & solutions, Problem Set 3 solutions, Problem Set 4

Quick Review

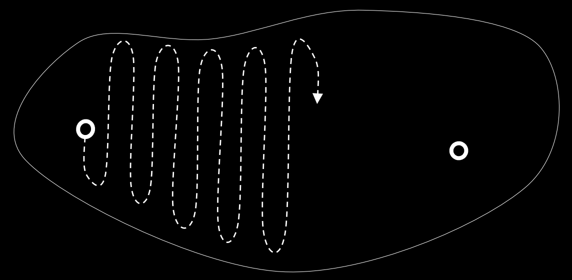
Defn: TIMEsome deterministic 1-tape TM decides
and runs in time

Defn: P
polynomial time decidable languages

is a directed graph with a path from to

Theorem:

is a directed graph with a path from to
that goes through every node of



? Unsolved Proble
[connection to factoring]

Nondeterministic Complexity

In a nondeterministic TM (NTM) decider, all branches halt on all inputs.

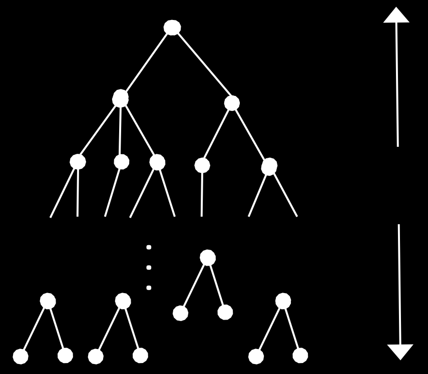
Defn: An NTM runs in time if all branches halt within steps on all inputs of length .

Defn: NTIME$B|$ some 1-tape NTM decides B
and runs in time }

Defn: NP
nondeterministic polynomial time decidable languages

- Invariant for all reasonable nondeterministic models
- Corresponds roughly to easily verifiable problems

Computation tree
for NTM on input .



all branches halt
within steps

NP

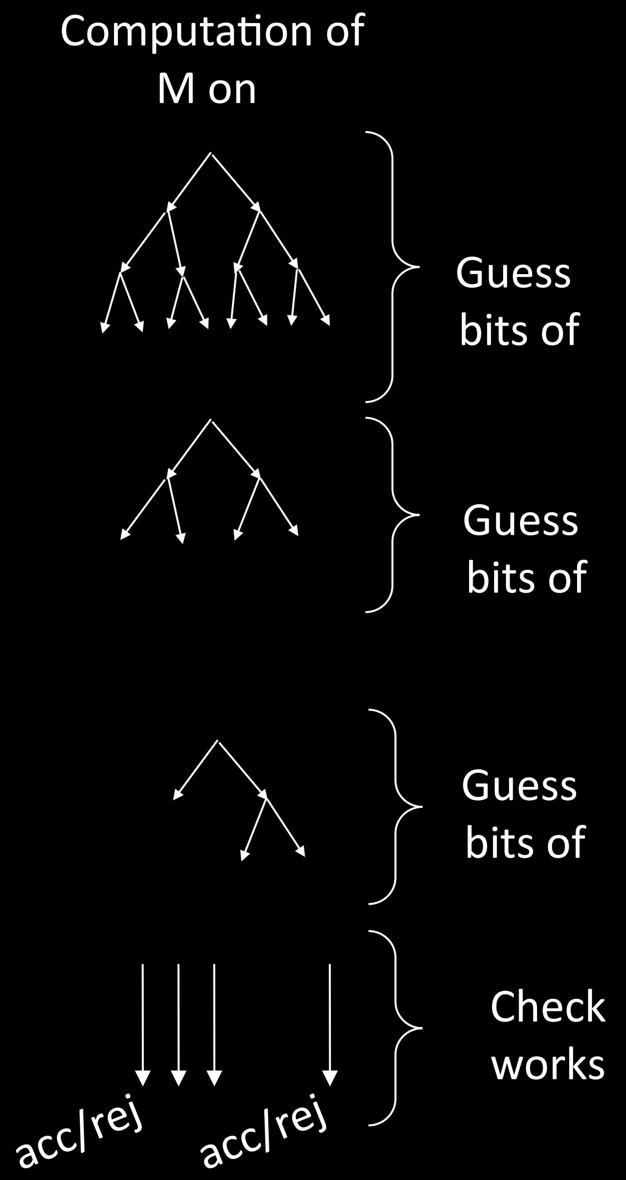
Theorem: NP

Proof:

"On input (Say has nodes.)

1. Nondeterministically write a sequence of nodes.
2. *Accept* if

   each is an edge and no repeats.
3. *Reject* if any condition fails."



NP

Defn: is not prime and is written in binary}

for integers , in binary}

Theorem: NP

Proof: "On input

1. Nondeterministically write where .
2. *Accept* if divides with remainder .
   Reject if not."

Note: Using base 10 instead of base 2 wouldn't matter because can convert in polynomial time.

Bad encoding: write number in unary: , exponentially longer.

Theorem (2002): P

We won't cover this proof.

Intuition for P and NP

NP = All languages where can <u>verify</u> membership quickly
P = All languages where can <u>test</u> membership quickly

Examples of quickly verifying membership:

- : Give the Hamiltonian path.
- : Give the factor.

The <u>Hamiltonian path</u> and the <u>factor</u> are called ***short certificates*** of membership.

Check-in 14.1

Let be the complement of .

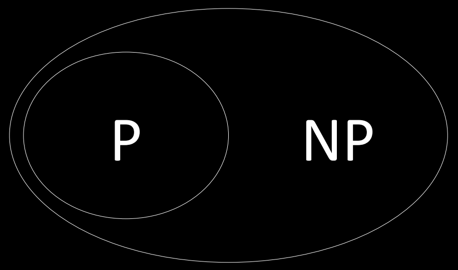
So if does <u>not</u> have a Hamiltonian path from to .

Is NP?

(a) Yes, we can invert the accept/reject output of the NTM for .

(b) No, we cannot give a short certificate for a graph not to have a Hamiltonian path.

(c) I don’t know.



Check-in 14.1


Coffee
Break

Recall

Recall:

Theorem: is decidable

Proof: "On input

1. Convert into Chomsky Normal Form.
2. Try all derivations of length .
3. *Accept* if any generate . *Reject* if not.

Chomsky Normal Form (CNF):

A BC

B b

Let's always assume is in CNF.

Theorem: NP

Proof: "On input

4. Nondeterministically pick some derivation of length .
5. *Accept* if it generates . *Reject* if not.

Attempt to show P

Theorem: P

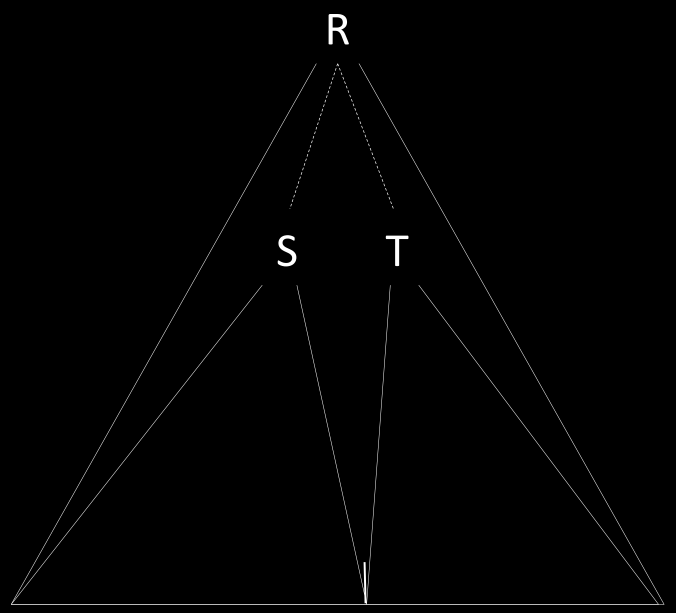
Proof attempt:

Recursive algorithm tests if generates , starting at any specified variable R.

“On input

1. For each way to divide and for each rule R ST
2.   Use to test and
3.   *Accept* if both accept
4. *Reject* if none of the above accepted.”

Then decide by starting from ’s start variable.



is a correct algorithm, but it takes non-polynomial time.

(Each recursion makes calls and depth is roughly .)

Fix: Use recursion + memory called *Dynamic Programming* (DP)

Observation: String of length has substrings

therefore there are only possible sub-problems to solve.

DP shows P

Theorem: P

Proof : Use DP (Dynamic Programming) = recursion + memory.

"On input

"memoization"

0. If previously solved, then answer same, else continue.
1. For each way to divide and for each rule R S T
2. Use to test and
3. *Accept* if both accept
4. *Reject* if none of the above accepted."

Then decide by starting from G's start variable.

same as before

Total number of calls is so time used is polynomial.

Alternately, solve all smaller sub-problems first: "bottom up"

Check-in 14.2

Suppose is a CFL.
Does that imply that P?

(a) Yes

(b) No.

Check-in 14.2

P & Bottom-up DP

Theorem: P

Proof : Use bottom-up DP.

“On input

1. For each and variable R
   Solve by checking if R is a rule.
2. For and each substring of where and variable R
   Solve by checking for each R ST and each division
   if both and were positive.
3. *Accept* if is positive where S is the original start variable.
4. *Reject* if not.”

Solve for substrings of length 1

Solve for substrings of length by using previous answers for substrings of length .

Total number of calls is so time used is polynomial.

Often, bottom-up DP is shown as filling out a table.

Satisfiability Problem

Defn: A ***Boolean formula*** has Boolean variables (TRUE/FALSE values) and Boolean operations AND (), OR (), and NOT ().

Defn: is ***satisfiable*** if evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let (Notation: means)
Then is satisfiable (x=1, y=0)

Defn: is a satisfiable Boolean formula}

Theorem (Cook, Levin 1971): P P = NP
Proof method: polynomial time (mapping) reducibility

Check-in 14.3

Is NP?

(a) Yes.

(b) No.

(c) I don't know.

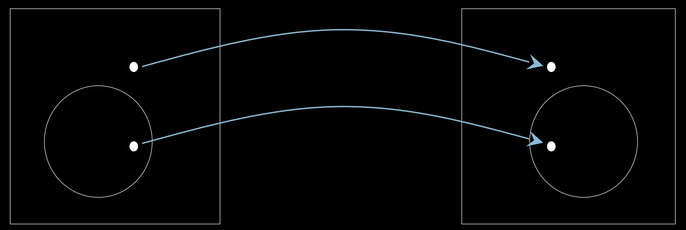
(d) No one knows.

Check-in 14.3

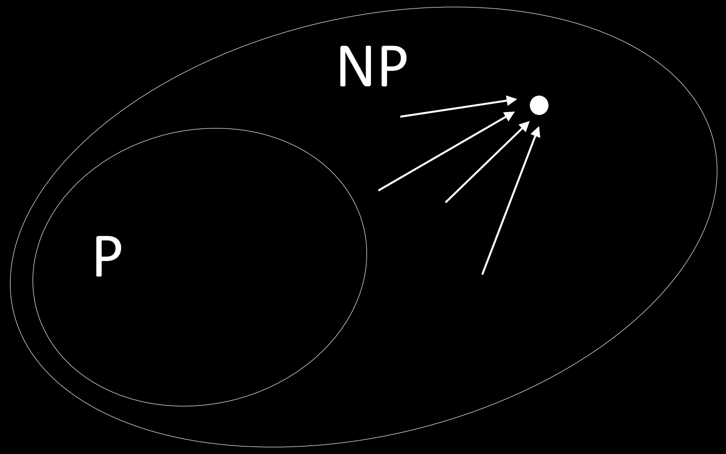
Polynomial Time Reducibility

Defn: is polynomial time reducible to () if
by a reduction function that is computable in polynomial time.

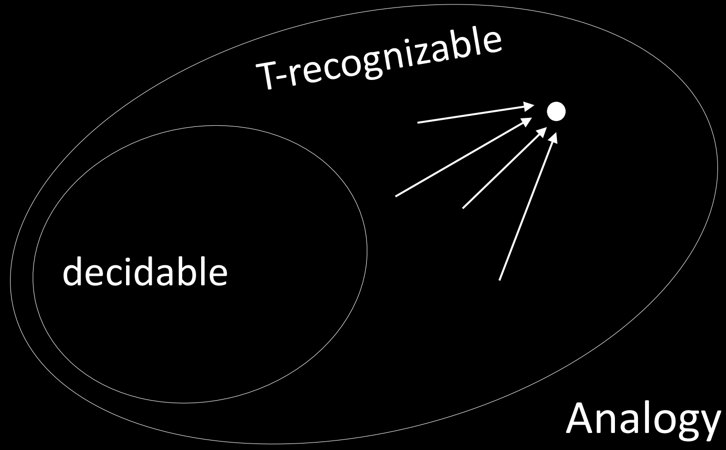
Theorem: If and P then P.



is computable in polynomial time



Idea to show P P = NP



Analogy with

Quick review of today

1. NTIME and NP
2. **and NP**
3. P versus NP question
4. P via Dynamic Programming
5. The Satisfiability Problem
6. Polynomial time reducibility