15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

RANDOMIZED COMPLEXITY

Tuesday April 22

Checking MATRIX MULTIPLICATION

L = { (M_1,M_2,N) | M_1, M_2 and N are matrices and M_1M_2 = N }

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

Checking MATRIX MULTIPLICATION

$L = \{ (M_1, M_2, N) \mid M_1, M_2$ and N are matrices and $M_1 M_2 = N \}$

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

Aside: multiplication of complex #s: standard method

$(a+ ib)(c + id) = ac + iad + ibc – bd = (ac – bd) + i(ad + bc)$
requires 4 (real) multiplications

Checking MATRIX MULTIPLICATION

L = { (M_1, M_2, N) | M_1, M_2 and N are matrices and $M_1 M_2 = N$ }

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

Aside: multiplication of complex #s: standard method

$(a+ ib)(c + id) = ac + iad + ibc – bd = (ac – bd) + i(ad + bc)$ requires 4 (real) multiplications

Another method (if multiplication expensive, addition cheap):
Let A = (a+b)(c+d) = ac+ad+bc+bd ; B = ac, C = bd

Then (a+bi)(c+di) = (B-C) + (A-B-C)i, which only requires 3 multiplications!

Checking MATRIX MULTIPLICATION

L = { (M_1,M_2,N) | M_1, M_2 and N are matrices and M_1M_2 = N }

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

$O(n^2)$ randomized algorithm to CHECK:

Checking MATRIX MULTIPLICATION

$L = \{ (M_1,M_2,N) \mid M_1, M_2$ and N are matrices and $M_1M_2 = N \}$

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

$O(n^2)$ randomized algorithm to CHECK:

Pick a 0-1 bit vector r at random, test if $M_1M_2r = Nr$

Checking MATRIX MULTIPLICATION

$L = \{ (M_1,M_2,N) \mid M_1, M_2$ and N are matrices and $M_1M_2 = N \}$

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

$O(n^2)$ randomized algorithm to CHECK:

Pick a 0-1 bit vector r at random, test if $M_1M_2r = Nr$

If $M_1M_2 = N$, then $\Pr[M_1M_2r = Nr] = 1$
If $M_1M_2 \neq N$, then $\Pr[M_1M_2r = Nr] \leq ½$ (CLAIM)

Checking MATRIX MULTIPLICATION

L = { (M_1,M_2,N) | M_1, M_2 and N are matrices and M_1M_2 = N }

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

$O(n^2)$ randomized algorithm to CHECK:

Pick a 0-1 bit vector r at random, test if $M_1M_2r = Nr$

If $M_1M_2 = N$, then Pr [$M_1M_2r = Nr$] = 1
If $M_1M_2 \neq N$, then Pr [$M_1M_2r = Nr$] $\leq$ ½ (CLAIM)

So, if we pick 300 random vectors and test them all, what is the probability of failing? **$1/2^{300}$**

Checking MATRIX MULTIPLICATION

L = { (M_1,M_2,N) | M_1, M_2 and N are matrices and $M_1M_2 = N$ }

If M_1 and M_2 are n x n matrices, multiplying them takes $O(n^3)$ time normally, and $O(n^{2.3727})$ time using newer methods.

$O(n^2)$ randomized algorithm to CHECK:

Pick a 0-1 bit vector r at random, test if $M_1M_2r = Nr$

If $M_1M_2 = N$, then Pr $[M_1M_2r = Nr] = 1$

If $M_1M_2 \neq N$, then Pr $[M_1M_2r = Nr] \leq 1/2$ (CLAIM)

Proof of CLAIN

- Consider the matrix M' = $\mathbf{M_1}$,$\mathbf{M_2}$– N.
- Suppose M' $\neq$ 0-matrix.
- We want to know Pr[M' r = 0-vector].

- Consider the matrix M’ = $\mathbf{M_1 M_2}$ – N.
- Suppose M’ $\neq$ 0-matrix.
- We want to know Pr[M’ r = 0-vector].
- Let i be a row of M’ that has a non-zero entry.
- Think of it as a vector v with $v_1 \neq 0$, say.
- Now Pr[M’ r = 0-vector] $\leq$ Pr[v r = 0].
- $\Pr[v_1 r_1 + v_2 r_2 + \ldots + v_n r_n = 0] = ?$

- Consider the matrix M’ = $\mathbf{M_1, M_2}$– N.
- Suppose M’ $\neq$ 0-matrix.
- We want to know Pr[M’ r = 0-vector].
- Let i be a row of M’ that has a non-zero entry.
- Think of it as a vector v with $v_1 \neq 0$, say.
- Now Pr[M’ r = 0-vector] $\leq$ Pr[v r = 0].
- $\Pr[v_1 r_1 + v_2 r_2 + \ldots + v_n r_n = 0] = ?$
- Suppose we’ve already chosen $r_2, \ldots, r_n$.
- If $v_1 r_1 + v_2 r_2 + \ldots + v_n r_n = 0$, then
- we must have $r_1 = (v_2 r_2 + \ldots + v_n r_n)/v_1$.
- There are two choices for r_1 though.
- So the probability we pick r_1 to be exactly this expression is at most ½.

TESTING POLYNOMIALS

Let p be a 1-variable polynomial.

How do we determine if p is always 0?

Let $p = a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_dx_1^d$

Simply try d+1 distinct values for the variables!

TESTING POLYNOMIALS

Let p be a 1-variable polynomial.

How do we determine if p is always 0?

Let $p = a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_dx_1^d$

Simply try d+1 distinct values for the variables!

(A degree d polynomial has at most d roots.)

TESTING POLYNOMIALS

Let p an n-variable polynomial over a finite field. How do we determine if p is always 0?

$$(2332x_1 + 4603x_2 - 3878x_3)(5566x_1 + 31x_4 - 171)(677x_7 - 1)(x_5 + 7x_6 + 3x_2 + 1001x_1) = 0 \pmod{6709}$$

Not given in standard way.

Simply try random values for the variables!

Theorem (Schwartz-Zippel): Let F be a finite field and let p be a NONZERO polynomial on the variables $x_1, x_2, \ldots, x_m$, where each variable has degree at most d. (Generally want: $|F| > 2md$)

If $a_1, \ldots, a_m$ are selected randomly from F, then:

$$\Pr[\, p(a_1, \ldots, a_m) = 0 \,] \leq md/|F|$$

Theorem (Schwartz-Zippel): Let F be a finite field and let p be a NONZERO polynomial on the variables $x_1,x_2,\ldots,x_m$, where each variable has degree at most d. (Generally want: $|F| > 2md$)

If $a_1,\ldots, a_m$ are selected randomly from F, then:

$$\Pr [\ p(a_1, \ldots, a_m) = 0\ ] \leq md/|F|$$

Proof (by induction on m):

Base Case (m = 1):

$$\Pr [\ p(a_1) = 0\ ] \leq d/|F|$$

A polynomial of degree d can have at most d roots, so at most d elements in F make p = 0

Inductive Step ($m > 1$):

Assume true for $m-1$ and prove true for m

Let x_1 be one of the variables

Write: $p = p_0 + x_1 p_1 + x_1^2 p_2 + \ldots + x_1^d p_d$

where x_1 does not occur in any p_i

If $p(a_1,\ldots,a_m) = 0$, one of two things can happen:

Inductive Step ($m > 1$):

Assume true for $m-1$ and prove true for m

Let x_1 be one of the variables

Write: $p = p_0 + x_1 p_1 + x_1^2 p_2 + \ldots + x_1^d p_d$

where x_1 does not occur in any p_i

If $p(a_1,\ldots,a_m) = 0$, one of two things can happen:

(1) For all i, $p_i(a_2,\ldots,a_m) = 0$

(2) Some I, $p_i(a_2,\ldots,a_m)$ is not 0, and a_1 is a root of the single variable polynomial on x_1 that results from evaluating $p_0,\ldots,p_m$ with $a_2,\ldots,a_m$

Inductive Step (m > 1):

Assume true for m-1 and prove true for m
Let x_1 be one of the variables

Write: $p = p_0 + x_1 p_1 + x_1^2 p_2 + \ldots + x_1^d p_d$

where x_1 does not occur in any p_i

If $p(a_1,\ldots,a_m) = 0$, one of two things can happen:

(1) For all i, $p_i(a_2,\ldots,a_m) = 0$

(2) Some I, $p_i(a_2,\ldots,a_m)$ is not 0, and a_1 is a root of the single variable polynomial on x_1 that results from evaluating $p_0,\ldots,p_m$ with $a_2,\ldots,a_m$

Pr [(1)] $\leq$ (m-1)d/|F| Pr [(2)] $\leq$ d/|F|

Pr [(1) or (2)] $\leq$ md/|F|

PROBABILISTIC ALGORITHMS

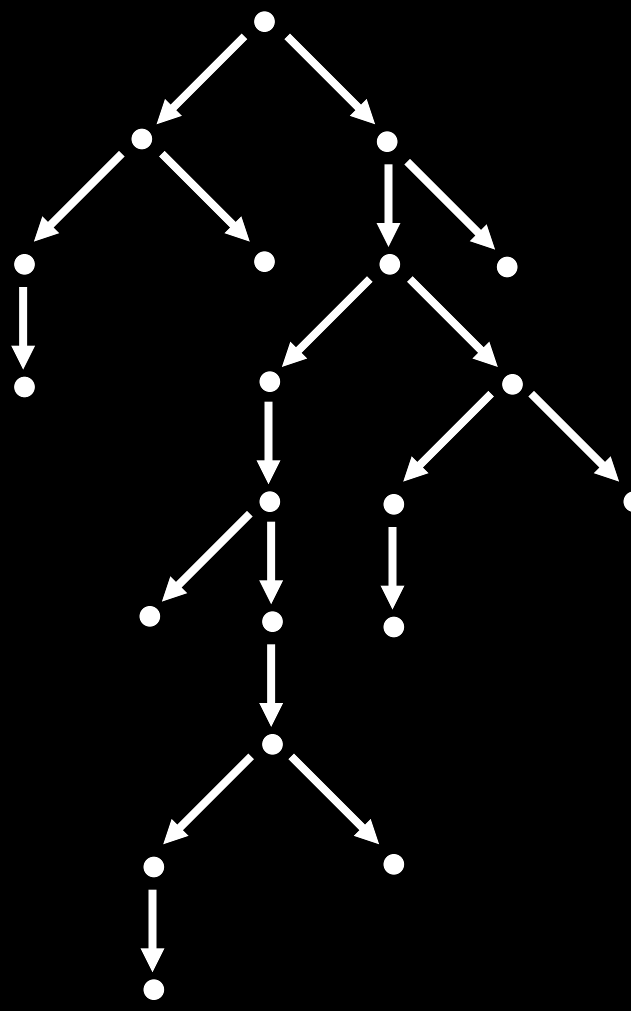
Why do we study probabilistic algorithms?

1. **Can be simpler than deterministic algs**

2. **Can be more efficient than deterministic algorithms**

3. **Does randomness make problems much easier to solve? We don't know!**

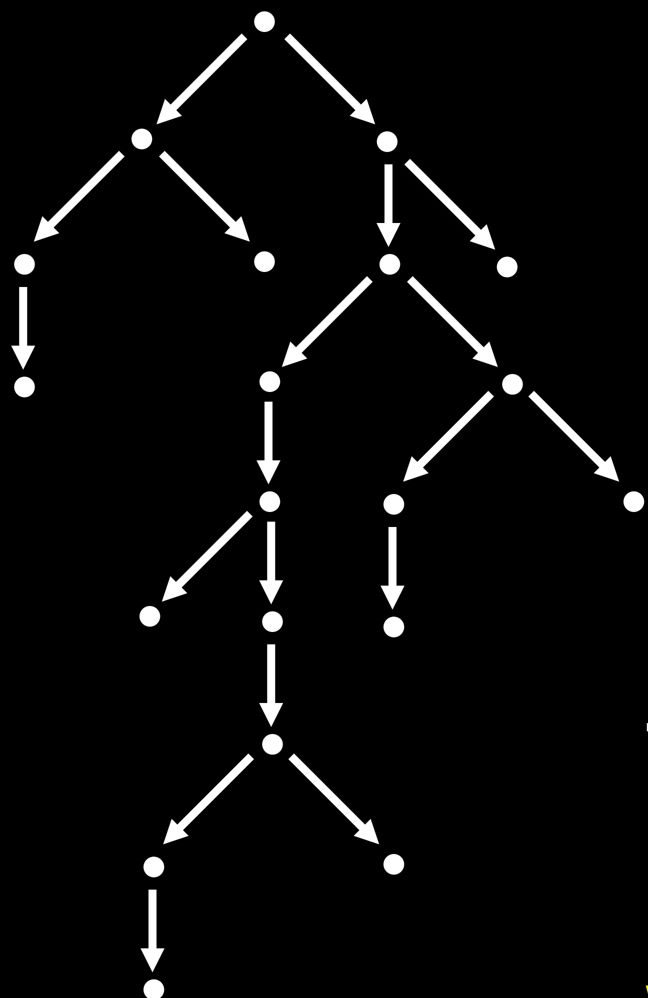
PROBABILISTIC TMs



A probabilistic TM M is a non-deterministic TM where:

- **Each non-deterministic step is called a coin flip**
- **Each non-deterministic step has only two legal next moves**

PROBABILISTIC TMs



A **probabilistic TM** M is a non-deterministic TM where:

Each non-deterministic step is called a **coin flip**

Each non-deterministic step has only two legal next moves

The probability of branch b is:

$$\Pr[\,b\,] = 2^{-k}$$

where k is the number of coin flips that occur on branch b

$$\Pr[\text{ M accepts w }] = \sum_{\text{b is an accepting branch}} \Pr[\text{ b }]$$

Definition: M recognizes language A **with error ε** if for all strings w:

$w \in A \Leftrightarrow \Pr[\text{ M accepts } w\ ] \geq 1 - \varepsilon$

$w \notin A \Leftrightarrow \Pr[\text{ M doesn't accept } w\ ] \geq 1 - \varepsilon$

BPP = { L | L is recognized by a probabilistic poly-time TM with error 1/3 }

Why 1/3?

BPP = { L | L is recognized by a probabilistic poly-time TM with error **1/3** }

Why 1/3?

Because it doesn't matter what number we pick as long as it is smaller than 1/2!

Theorem: Let ε be a constant, $0 < \varepsilon < 1/2$ and let p(n) be a polynomial.

If M_1 has error ε then there is an equivalent M_2 with error $2^{-p(n)}$

Proof Idea:

M_2 simply runs M_1 many times and takes the majority output

Let F be a finite field

ZERO-POLY_F = { p | p is a polynomial over F (with 2md < |F|) that is zero on all points}

$\text{ZERO-POLY}_F \in \text{BPP}$

BPP = { L | L is recognized by a probabilistic poly-time TM with error 1/3 }

Is BPP $\subseteq$ NP?

BPP = { L | L is recognized by a probabilistic poly-time TM with error 1/3 }

Is BPP $\subseteq$ NP?

Nobody knows for sure!

Is NP $\subseteq$ BPP?

Is NP $\subseteq$ BPP?

Nobody knows for sure!

Is BPP $\subseteq$ PSPACE?

Is BPP $\subseteq$ PSPACE?

Yes! Simply run all branches and count the number of branches that accept.

Definition: A language A is in **RP (Randomized P)** if there is a nondeterministic polynomial time TM M such that for all strings x:

$x \notin A \Leftrightarrow$ No computation paths accept

$x \in A \Leftrightarrow$ At least half of the paths accept

Theorem: A language A is in **RP (Randomized P)** if **for each k** there is a nondeterministic polynomial time TM M such that for all strings x:

$x \notin A \Leftrightarrow$ M(x) always rejects

$x \in A \Leftrightarrow$ M(x) accepts with probability at least $1 - 2^k$

Is RP $\subseteq$ BPP?

Is RP $\subseteq$ BPP?

Yes!

Is RP $\subseteq$ NP?

Is RP $\subseteq$ NP?

Yes!

PRIMES = { p | p is a prime number}

Used to be:

PRIMES $\in$ BPP

COMPOSITES $\in$ RP

By an extension of Fermat's Little Theorem:

p, prime , a^{p-1} =1 (mod p) for a $\neq$ 0 (mod p)

PRIMES = { p | p is a prime number}

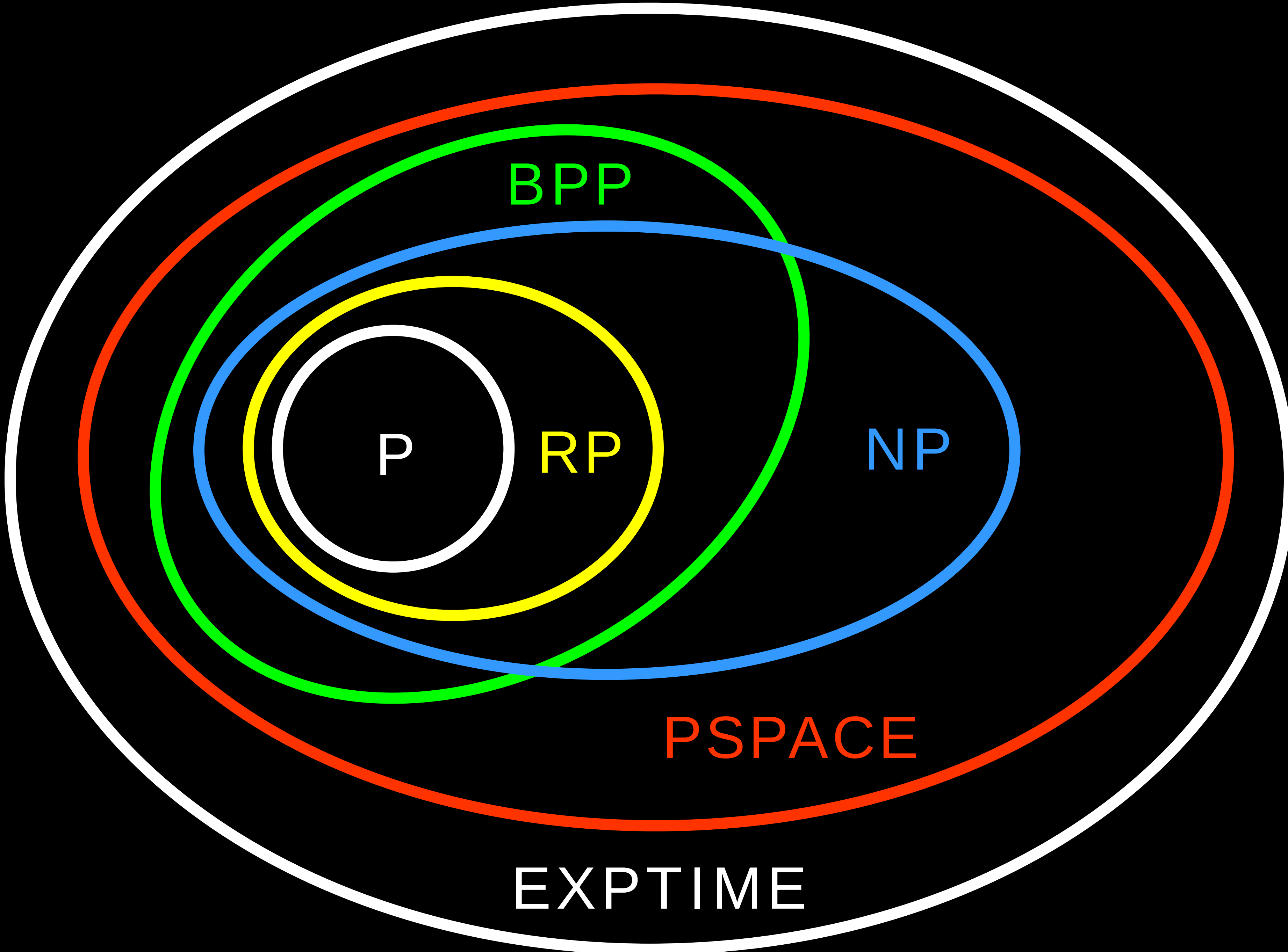
PRIMES is in P

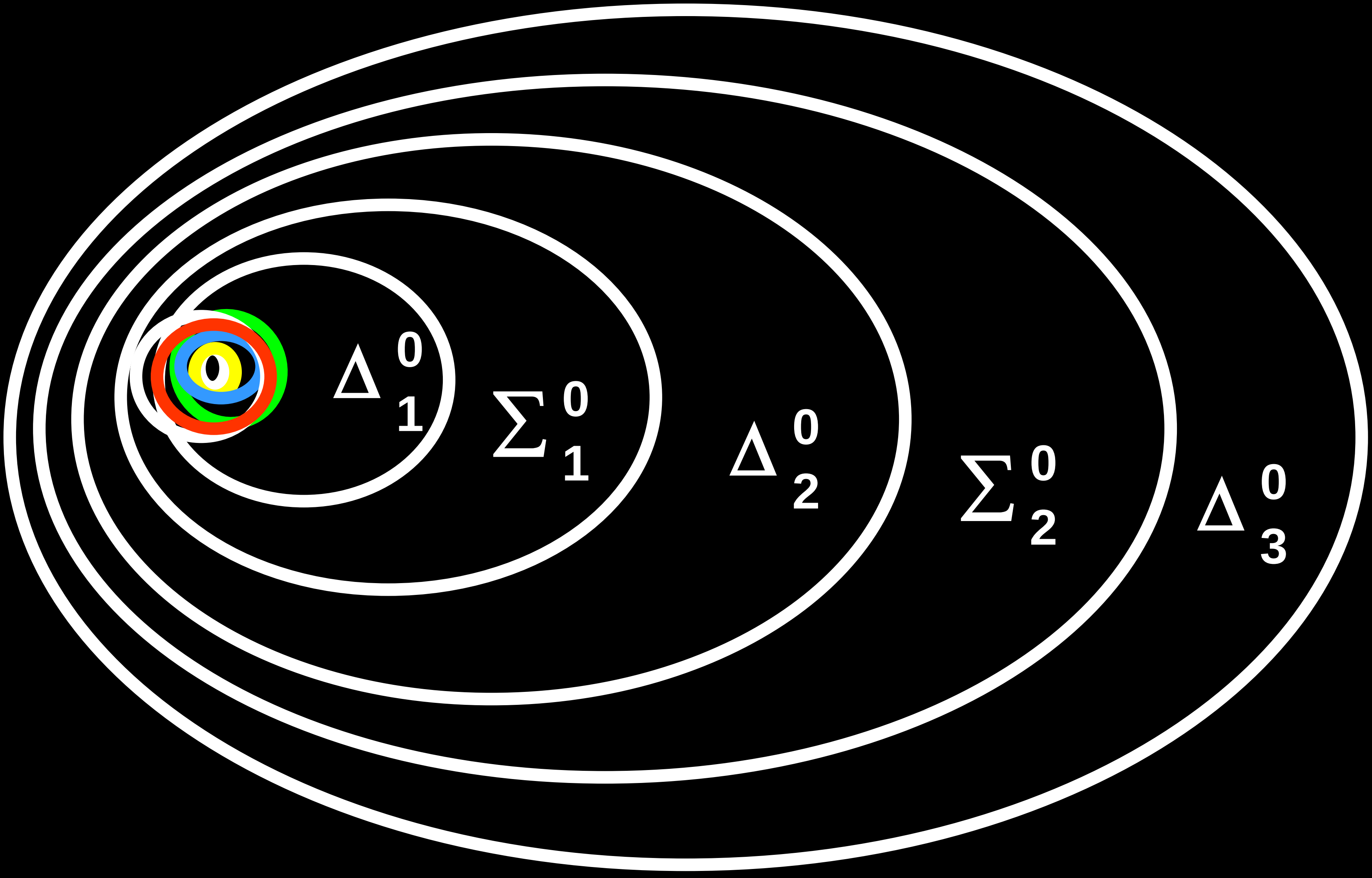
Manindra Agrawal, Neeraj Kayal and Nitin Saxena

Source: Ann. of Math. Volume 160, Number 2 (2004), 781-793.

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.


BPP
P
RP
NP
PSPACE
EXPTIME


Δ^0_1
Σ^0_1
Δ^0_2
Σ^0_2
Δ^0_3