Exponents

- The **exponent** of $m \in \Phi(p)$ is the least $k \in \mathbb{Z}^+$ such that
$$m^k = 1 \bmod p.$$
- Every residue $s \in \Phi(p)$ has an exponent.
  - $1, s, s^2, s^3, \ldots$ eventually repeats itself, say $s^i = s^j \bmod p$, which means $s^{j-i} = 1 \bmod p$.
- If the exponent of m is k and $m^\ell = 1 \bmod p$, then $k | \ell$.
  - Otherwise, $\ell = qk + a$ for $0 < a < k$, and $m^\ell = m^{qk+a} = m^a = 1 \bmod p$, a contradiction.

Lemma 54 *Any nonzero polynomial of degree k has at most k distinct roots modulo p.*

Exponents and Primitive Roots

- From Fermat's "little" theorem, all exponents divide $p - 1$.
- A primitive root of p is thus a number with exponent $p - 1$.
- Let $R(k)$ denote the total number of residues in $\Phi(p)$ that have exponent k.
- We already knew that $R(k) = 0$ for $k \not| (p - 1)$.
- So $\sum_{k|(p-1)} R(k) = p - 1$ as every number has an exponent.

Size of $R(k)$

- Any $a \in \Phi(p)$ of exponent k satisfies $x^k = 1 \bmod p$.
- Hence there are at most k residues of exponent k, i.e., $R(k) \leq k$, by Lemma 54 on p. 370.
- Let s be a residue of exponent k.
- $1, s, s^2, \ldots, s^{k-1}$ are all distinct modulo p.
  - Otherwise, $s^i = s^j \bmod p$ with $i < j$ and s is of exponent $j - i < k$, a contradiction.
- As all these k distinct numbers satisfy $x^k = 1 \bmod p$, they are all the solutions of $x^k = 1 \bmod p$.
- But do all of them have exponent k (i.e., $R(k) = k$)?

Size of $R(k)$ (continued)

- And if not (i.e., $R(k) < k$), how many of them do?
- Suppose $\ell < k$ and $\ell \notin \Phi(k)$ with $\gcd(\ell, k) = d > 1$.
- Then
$$(s^{\ell})^{k/d} = (s^{k})^{\ell/d} = 1 \bmod p.$$
- Therefore, s^{ℓ} has exponent at most k/d, which is less than k.
- We conclude that
$$R(k) \leq \phi(k).$$

Size of $R(k)$ (concluded)

- Because all $p-1$ residues have an exponent,

$$p-1=\sum_{k\,|\,(p-1)} R(k) \leq \sum_{k\,|\,(p-1)} \phi(k)=p-1$$

by Lemma 50 on p. 359.

- Hence

$$R(k)=\begin{cases}\phi(k) & \text{when } k|(p-1)\\ 0 & \text{otherwise}\end{cases}$$

- In particular, $R(p-1)=\phi(p-1)>0$, and p has at least one primitive root.

- This proves one direction of Theorem 46 (p. 351).

A Few Calculations

- Let $p = 13$.
- From p. 367, we know $\phi(p-1) = 4$.
- Hence $R(12) = 4$.
- And there are 4 primitives roots of p.
- As $\Phi(p-1) = \{1, 5, 7, 11\}$, the primitive roots are g^1, g^5, g^7, g^{11} for any primitive root g.

The Other Direction of Theorem 46 (p. 351)

- We must show p is a prime only if there is a number r (called primitive root) such that
  1. $r^{p-1} = 1 \bmod p$, and
  2. $r^{(p-1)/q} \neq 1 \bmod p$ for all prime divisors q of $p - 1$.
- Suppose p is not a prime.
- We proceed to show that no primitive roots exist.
- Suppose $r^{p-1} = 1 \bmod p$ (note $\gcd(r, p) = 1$).
- We will show that the 2nd condition must be violated.

The Proof (concluded)

- $r^{\phi(p)} = 1 \bmod p$ by the Fermat-Euler theorem (p. 367).
- Because p is not a prime, $\phi(p) < p - 1$.
- Let k be the smallest integer such that $r^k = 1 \bmod p$.
- As $k \leq \phi(p)$, $k < p - 1$.
- Let q be a prime divisor of $(p-1)/k > 1$.
- Then $k | (p-1)/q$.
- Therefore, by virtue of the definition of k,

$$r^{(p-1)/q} = 1 \bmod p.$$

- But this violates the 2nd condition.

Function Problems

- Decisions problem are yes/no problems (SAT, TSP (D), etc.).
- **Function problems** require a solution (a satisfying truth assignment, a best TSP tour, etc.).
- Optimization problems are clearly function problems.
- What is the relation between function and decision problems?
- Which one is harder?

Function Problems Cannot Be Easier than Decision Problems

- If we know how to generate a solution, we can solve the corresponding decision problem.
  - If you can find a satisfying truth assignment efficiently, then SAT is in P.
  - If you can find the best TSP tour efficiently, then TSP (D) is in P.
- But decision problems can be as hard as the corresponding function problems.

FSAT

- FSAT is this function problem:
  - Let $\phi(x_1, x_2, \ldots, x_n)$ be a boolean expression.
  - If ϕ is satisfiable, then return a satisfying truth assignment.
  - Otherwise, return "no."
- We next show that if SAT $\in$ P, then FSAT has a polynomial-time algorithm.

An Algorithm for FSAT Using SAT

```
t := ϵ;
if ϕ ∈ SAT then
    for i = 1, 2, ..., n do
        if ϕ[ x_i = true ] ∈ SAT then
            t := t ∪ { x_i = true };
            ϕ := ϕ[ x_i = true ];
        else
            t := t ∪ { x_i = false };
            ϕ := ϕ[ x_i = false ];
        end if
    end for
    return t;
else
    return "no";
end if
```

Analysis

- There are $\leq n+1$ calls to the algorithm for SAT.[a]
- Shorter boolean expressions than ϕ are used in each call to the algorithm for SAT.
- So if SAT can be solved in polynomial time, so can FSAT.
- Hence SAT and FSAT are equally hard (or easy).

[a] Contributed by Ms. Eva Ou (R93922132) on November 24, 2004.

TSP and TSP (D) Revisited

- We are given n cities $1, 2, \ldots, n$ and integer distances $d_{ij} = d_{ji}$ between any two cities i and j.
- The TSP asks for a tour with the shortest total distance (not just the shortest total distance, as earlier).
  - The shortest total distance must be at most $2^{|\, x \,|}$, where x is the input.
- TSP (D) asks if there is a tour with a total distance at most B.
- We next show that if TSP (D) $\in$ P, then TSP has a polynomial-time algorithm.

An Algorithm for TSP Using TSP (D)

Perform a binary search over interval $[\,0, 2^{|\,x\,|}\,]$ by calling TSP (D) to obtain the shortest distance C;

for $i, j = 1, 2, \ldots, n$ **do**

  Call TSP (D) with $B = C$ and $d_{ij} = C + 1$;

  if "no" **then**

    Restore d_{ij} to old value; {Edge $[\,i, j\,]$ is critical.}

  end if

end for

return the tour with edges whose $d_{ij} \leq C$;

Analysis

- An edge that is not on *any* optimal tour will be eliminated, with its d_{ij} set to $C+1$.
- An edge which is not on all remaining optimal tours will also be eliminated.
- So the algorithm ends with n edges which are not eliminated (why?).
- There are $O(|x| + n^2)$ calls to the algorithm for TSP (D).
- So if TSP (D) can be solved in polynomial time, so can TSP.
- Hence TSP (D) and TSP are equally hard (or easy).

Randomized Computation

I know that half my advertising works,
I just don't know which half.
— John Wanamaker

I know that half my advertising is
a waste of money,
I just don't know which half!
— McGraw-Hill ad.

Randomized Algorithms[a]

- Randomized algorithms flip unbiased coins.
- There are important problems for which there are no known efficient *deterministic* algorithms but for which very efficient randomized algorithms exist.
  - Extraction of square roots, for instance.
- There are problems where randomization is *necessary*.
  - Secure protocols.
- Randomized version can be more efficient.
  - Parallel algorithm for maximal independent set.
- Are randomized algorithms algorithms?

[a]Rabin (1976); Solovay and Strassen (1977).

"Four Most Important Randomized Algorithms"[a]

1. Primality testing.[b]

2. Graph connectivity using random walks.[c]

3. Polynomial identity testing.[d]

4. Algorithms for approximate counting.[e]

---

[a] Trevisan (2006).
[b] Rabin (1976); Solovay and Strassen (1977).
[c] Aleliunas, Karp, Lipton, Lovász, and Rackoff (1979).
[d] Schwartz (1980); Zippel (1979).
[e] Sinclair and Jerrum (1989).

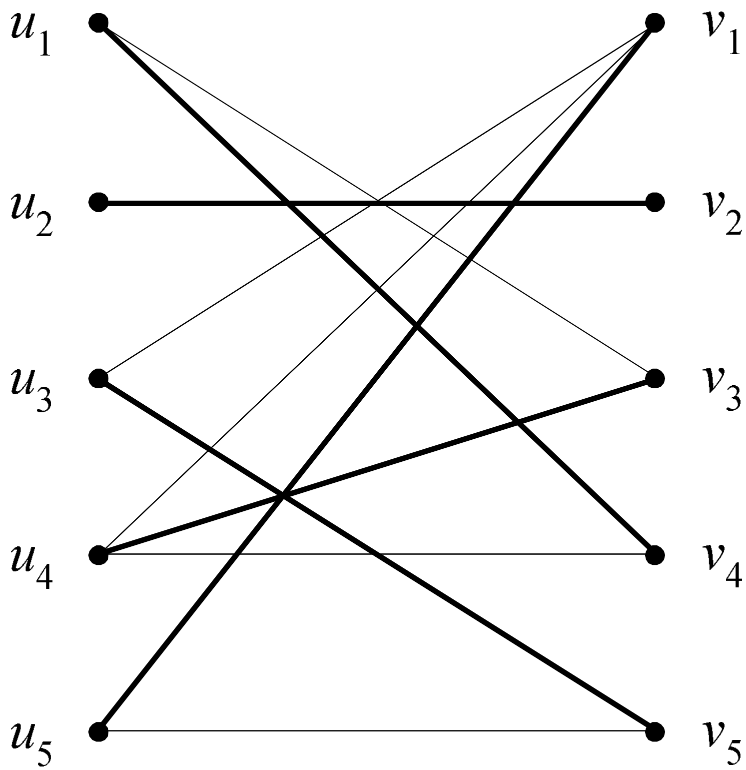
Bipartite Perfect Matching

- We are given a **bipartite graph** $G = (U, V, E)$.
  - $U = \{u_1, u_2, \ldots, u_n\}$.
  - $V = \{v_1, v_2, \ldots, v_n\}$.
  - $E \subseteq U \times V$.
- We are asked if there is a **perfect matching**.
  - A permutation π of $\{1, 2, \ldots, n\}$ such that

$$(u_i, v_{\pi(i)}) \in E$$

    for all $u_i \in U$.

A Perfect Matching



Symbolic Determinants

- Given a bipartite graph G, construct the $n \times n$ matrix A^G whose (i, j)th entry A^G_{ij} is a variable x_{ij} if $(u_i, v_j) \in E$ and zero otherwise.
- The **determinant** of A^G is

$$\det(A^G) = \sum_{\pi} \operatorname{sgn}(\pi) \prod_{i=1}^{n} A^G_{i,\pi(i)}. \tag{5}$$

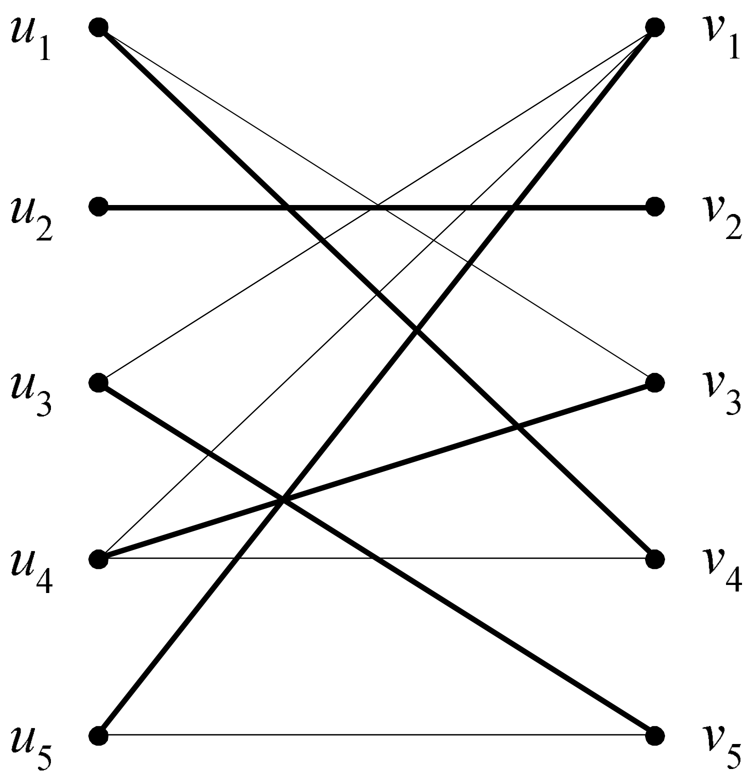
  - π ranges over all permutations of n elements.
  - $\operatorname{sgn}(\pi)$ is 1 if π is the product of an even number of transpositions and -1 otherwise.

Determinant and Bipartite Perfect Matching

- In $\sum_\pi \operatorname{sgn}(\pi) \prod_{i=1}^n A^G_{i,\pi(i)}$, note the following:
  - Each summand corresponds to a possible prefect matching π.
  - As all variables appear only *once*, all of these summands are different monomials and will not cancel.
- It is essentially an exhaustive enumeration.

Proposition 55 (Edmonds (1967)) *G has a perfect matching if and only if* $\det(A^G)$ *is not identically zero.*

A Perfect Matching in a Bipartite Graph



The Perfect Matching in the Determinant

- The matrix is

$$A^G = \begin{bmatrix} 0 & 0 & x_{13} & \boxed{x_{14}} & 0 \\ 0 & \boxed{x_{22}} & 0 & 0 & 0 \\ x_{31} & 0 & 0 & 0 & \boxed{x_{35}} \\ x_{41} & 0 & \boxed{x_{43}} & x_{44} & 0 \\ \boxed{x_{51}} & 0 & 0 & 0 & x_{55} \end{bmatrix}.$$

- $\det(A^G) = -x_{14}x_{22}x_{35}x_{43}x_{51} + x_{13}x_{22}x_{35}x_{44}x_{51} + x_{14}x_{22}x_{31}x_{43}x_{55} - x_{13}x_{22}x_{31}x_{44}x_{55}$, each denoting a perfect matching.

How To Test If a Polynomial Is Identically Zero?

- $\det(A^G)$ is a polynomial in n^2 variables.
- There are exponentially many terms in $\det(A^G)$.
- Expanding the determinant polynomial is not feasible.
  - Too many terms.
- Observation: If $\det(A^G)$ is *identically zero*, then it remains zero if we substitute *arbitrary* integers for the variables $x_{11}, \ldots, x_{nn}$.
- What is the likelihood of obtaining a zero when $\det(A^G)$ is *not* identically zero?

Number of Roots of a Polynomial

Lemma 56 (Schwartz (1980)) *Let* $p(x_1, x_2, \ldots, x_m) \not\equiv 0$ *be a polynomial in* m *variables each of degree at most* d*. Let* $M \in \mathbb{Z}^+$*. Then the number of* m*-tuples*

$$(x_1, x_2, \ldots, x_m) \in \{0, 1, \ldots, M-1\}^m$$

such that $p(x_1, x_2, \ldots, x_m) = 0$ *is*

$$\leq mdM^{m-1}.$$

- By induction on m (consult the textbook).

Density Attack

- The density of roots in the domain is at most

$$\frac{mdM^{m-1}}{M^m} = \frac{md}{M}.$$

- So suppose $p(x_1, x_2, \ldots, x_m) \not\equiv 0$.
- Then a random

$$(x_1, x_2, \ldots, x_n) \in \{\, 0, 1, \ldots, M-1 \,\}^n$$

has a probability of $\leq md/M$ of being a root of p.

Density Attack (concluded)

Here is a sampling algorithm to test if $p(x_1, x_2, \ldots, x_m) \not\equiv 0$.

1: Choose $i_1, \ldots, i_m$ from $\{0, 1, \ldots, M-1\}$ randomly;
2: **if** $p(i_1, i_2, \ldots, i_m) \neq 0$ **then**
3:     **return** "p is not identically zero";
4: **else**
5:     **return** "p is identically zero";
6: **end if**

A Randomized Bipartite Perfect Matching Algorithm[a]

We now return to the original problem of bipartite perfect matching.

Choose n^2 integers $i_{11}, \ldots, i_{nn}$ from $\{0, 1, \ldots, b-1\}$ randomly;
Calculate $\det(A^G(i_{11}, \ldots, i_{nn}))$ by Gaussian elimination;
if $\det(A^G(i_{11}, \ldots, i_{nn})) \neq 0$ **then**
    return "G has a perfect matching";
else
    return "G has no perfect matchings";
end if

---

[a] Lovász (1979).

Analysis

- Pick $b = 2n^2$.
- If G has no perfect matchings, the algorithm will always be correct.
- Suppose G has a perfect matching.
  - The algorithm will answer incorrectly with probability at most $n^2 d/b = 0.5$ because $d = 1$.
  - Run the algorithm *independently* k times and output "G has no perfect matchings" if they all say no.
  - The error probability is now reduced to at most 2^{-k}.
- Is there an $(i_{11}, \ldots, i_{nn})$ that will always give correct answers for all bipartite graphs of $2n$ nodes?[a]

[a] Thanks to a lively class discussion on November 24, 2004.

Perfect Matching for General Graphs

- Page 390 is about bipartite perfect matching
- Now we are given a graph $G = (V, E)$.
  - $V = \{v_1, v_2, \ldots, v_{2n}\}$.
- We are asked if there is a perfect matching.
  - A permutation π of $\{1, 2, \ldots, 2n\}$ such that

$$(v_i, v_{\pi(i)}) \in E$$

    for all $v_i \in V$.

The Tutte Matrix[a]

- Given a graph $G = (V, E)$, construct the $2n \times 2n$ **Tutte** matrix T^G such that

$$T^G_{ij} = \begin{cases} x_{ij} & \text{if } (v_i, v_j) \in E \text{ and } i < j, \\ -x_{ij} & \text{if } (v_i, v_j) \in E \text{ and } i > j, \\ 0 & \text{othersie.} \end{cases}$$

- The Tutte matrix is a skew-symmetric symbolic matrix.
- Similar to Proposition 55 (p. 393):

Proposition 57 *G has a perfect matching if and only if $\det(T^G)$ is not identically zero.*

---

[a]William Thomas Tutte (1917–2002).

Monte Carlo Algorithms[a]

- The randomized bipartite perfect matching algorithm is called a **Monte Carlo algorithm** in the sense that
  - If the algorithm finds that a matching exists, it is always correct (no **false positives**).
  - If the algorithm answers in the negative, then it may make an error (**false negative**).
- The algorithm makes a false negative with probability ≤ 0.5.
- This probability is *not* over the space of all graphs or determinants, but *over* the algorithm's own coin flips.
  - It holds for *any* bipartite graph.

[a] Metropolis and Ulam (1949).

The Markov Inequality[a]

Lemma 58 *Let x be a random variable taking nonnegative integer values. Then for any $k > 0$,*

$$\text{prob}[x \geq kE[\,x\,]] \leq 1/k.$$

- Let p_i denote the probability that $x = i$.

$$\begin{aligned} E[\,x\,] &= \sum_i ip_i \\ &= \sum_{i<kE[\,x\,]} ip_i + \sum_{i \geq kE[\,x\,]} ip_i \\ &\geq kE[\,x\,] \times \text{prob}[x \geq kE[\,x\,]]. \end{aligned}$$

---

[a]Andrei Andreyevich Markov (1856–1922).

An Application of Markov's Inequality

- Algorithm C runs in expected time $T(n)$ and always gives the right answer.
- Consider an algorithm that runs C for time $kT(n)$ and rejects the input if C does not stop within the time bound.
- By Markov's inequality, this new algorithm runs in time $kT(n)$ and gives the wrong answer with probability $\leq 1/k$.
- By running this algorithm m times, we reduce the error probability to $\leq k^{-m}$.

An Application of Markov's Inequality (concluded)

- Suppose, instead, we run the algorithm for the same running time $mkT(n)$ once and rejects the input if it does not stop within the time bound.
- By Markov's inequality, this new algorithm gives the wrong answer with probability $\leq 1/(mk)$.
- This is a far cry from the previous algorithm's error probability of $\leq k^{-m}$.
- The loss comes from the fact that Markov's inequality does not take advantage of any specific feature of the random variable.

FSAT for k-SAT Formulas (p. 380)

- Let $\phi(x_1, x_2, \ldots, x_n)$ be a k-SAT formula.
- If ϕ is satisfiable, then return a satisfying truth assignment.
- Otherwise, return "no."
- We next propose a randomized algorithm for this problem.

A Random Walk Algorithm for ϕ in CNF Form

Start with an *arbitrary* truth assignment T;
for $i = 1, 2, \ldots, r$ **do**
    if $T \models \phi$ **then**
        return "ϕ is satisfiable with T";
    else
        Let c be an unsatisfiable clause in ϕ under T; {All of its literals are false under T.}
        Pick any x of these literals *at random*;
        Modify T to make x true;
    end if
end for
return "ϕ is unsatisfiable";

3SAT vs. 2SAT Again

- Note that if ϕ is unsatisfiable, the algorithm will not refute it.
- The random walk algorithm needs expected exponential time for 3SAT.
  - In fact, it runs in expected $O((1.333\cdots+\epsilon)^n)$ time with $r = 3n$,[a] much better than $O(2^n)$.[b]
- We will show immediately that it works well for 2SAT.
- The state of the art is expected $O(1.322^n)$ time for 3SAT and expected $O(1.474^n)$ time for 4SAT.[c]

---

[a] Use this setting per run of the algorithm.
[b] Schöning (1999).
[c] Kwama and Tamaki (2004); Rolf (2006).

Random Walk Works for 2SAT[a]

Theorem 59 *Suppose the random walk algorithm with $r = 2n^2$ is applied to any satisfiable* 2SAT *problem with n variables. Then a satisfying truth assignment will be discovered with probability at least 0.5.*

- Let $\hat{T}$ be a truth assignment such that $\hat{T} \models \phi$.
- Let $t(i)$ denote the expected number of repetitions of the flipping step until a satisfying truth assignment is found if our starting T differs from $\hat{T}$ in i values.
  - Their Hamming distance is i.

[a]Papadimitriou (1991).

The Proof

- It can be shown that $t(i)$ is finite.
- $t(0) = 0$ because it means that $T = \hat{T}$ and hence $T \models \phi$.
- If $T \neq \hat{T}$ or T is not equal to any other satisfying truth assignment, then we need to flip at least once.
- We flip to pick among the 2 literals of a clause not satisfied by the present T.
- At least one of the 2 literals is true under $\hat{T}$, because $\hat{T}$ satisfies all clauses.
- So we have at least 0.5 chance of moving closer to $\hat{T}$.

The Proof (continued)

- Thus
$$t(i) \leq \frac{t(i-1)+t(i+1)}{2} + 1$$
for $0 < i < n$.
  - Inequality is used because, for example, T may differ from $\hat{T}$ in both literals.
- It must also hold that
$$t(n) \leq t(n-1) + 1$$
because at $i = n$, we can only decrease i.

The Proof (continued)

- As we are only interested in upper bounds, we solve

$$
\begin{aligned}
x(0) &= 0 \\
x(n) &= x(n-1)+1 \\
x(i) &= \frac{x(i-1)+x(i+1)}{2}+1, \quad 0<i<n
\end{aligned}
$$

- This is one-dimensional random walk with a reflecting and an absorbing barrier.

The Proof (continued)

- Add the equations up to obtain

$$\begin{aligned} & x(1) + x(2) + \cdots + x(n) \\ = \quad & \frac{x(0) + x(1) + 2x(2) + \cdots + 2x(n-2) + x(n-1) + x(n)}{2} \\ & +n + x(n-1). \end{aligned}$$

- Simplify to yield

$$\frac{x(1) + x(n) - x(n-1)}{2} = n.$$

- As $x(n) - x(n-1) = 1$, we have

$$x(1) = 2n - 1.$$

The Proof (continued)

- Iteratively, we obtain

$$\begin{aligned} x(2) &= 4n - 4, \\ &\vdots \\ x(i) &= 2in - i^2. \end{aligned}$$

- The worst case happens when $i = n$, in which case

$$x(n) = n^2.$$

The Proof (concluded)

- We therefore reach the conclusion that

$$t(i) \leq x(i) \leq x(n) = n^2.$$

- So the expected number of steps is at most n^2.
- The algorithm picks a running time $2n^2$.
- This amounts to invoking the Markov inequality (p. 405) with $k = 2$, with the consequence of having a probability of 0.5.
- The proof does not yield a polynomial bound for 3SAT.[a]

---

[a]Contributed by Mr. Cheng-Yu Lee (R95922035) on November 8, 2006.

Boosting the Performance

- We can pick $r = 2mn^2$ to have an error probability of $\leq (2m)^{-1}$ by Markov's inequality.
- Alternatively, with the same running time, we can run the "$r = 2n^2$" algorithm m times.
- But the error probability is reduced to $\leq 2^{-m}$!
- Again, the gain comes from the fact that Markov's inequality does not take advantage of any specific feature of the random variable.
- The gain also comes from the fact that the two algorithms are different.

How about Random CNF?

- Select m clauses independently and uniformly from the set of all possible disjunctions of k distinct, non-complementary literals with n boolean variables.
- Let $m = cn$.
- The formula is satisfiable with probability approaching 1 as $n \to \infty$ if $c < c_k$ for some $c_k < 2^k \ln 2 - O(1)$.
- The formula is unsatisfiable with probability approaching 1 as $n \to \infty$ if $c > c_k$ for some $c_k > 2^k \ln 2 - O(k)$.
- The above bounds are not tight yet.