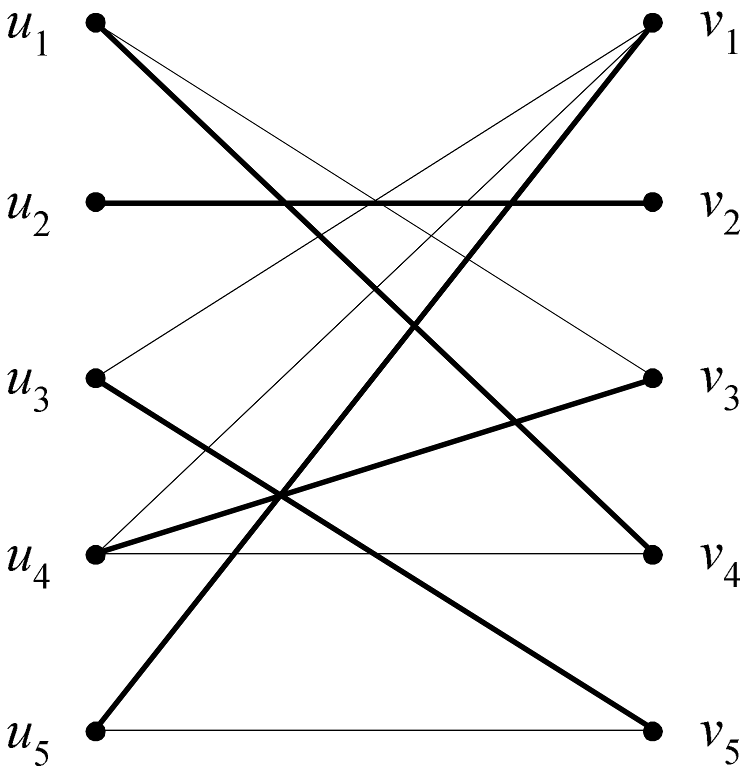
Bipartite Perfect Matching

- 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 $i \in \{1, 2, \ldots, n\}$.

A Perfect Matching in a Bipartite Graph



Symbolic Determinants

- We are given a bipartite graph G.
- Construct the $n \times n$ matrix A^G whose (i, j)th entry A^G_{ij} is a symbolic variable x_{ij} if $(u_i, v_j) \in E$ and 0 otherwise, or

$$A^G_{ij} = \begin{cases} x_{ij}, & \text{if } (u_i, v_j) \in E, \\ 0, & \text{othersie.} \end{cases}$$

Symbolic Determinants (continued)

- The matrix for the bipartite graph G on p. 481 is[a]

$$A^G = \begin{bmatrix} 0 & 0 & x_{13} & x_{14} & 0 \\ 0 & x_{22} & 0 & 0 & 0 \\ x_{31} & 0 & 0 & 0 & x_{35} \\ x_{41} & 0 & x_{43} & x_{44} & 0 \\ x_{51} & 0 & 0 & 0 & x_{55} \end{bmatrix}. \tag{6}$$

---

[a] The idea is similar to the Tanner graph in coding theory by Tanner (1981).

Symbolic Determinants (concluded)

- The **determinant** of A^G is
$$\det(A^G) = \sum_{\pi} \text{sgn}(\pi) \prod_{i=1}^{n} A^G_{i,\pi(i)}. \tag{7}$$
  - π ranges over all permutations of n elements.
  - $\text{sgn}(\pi)$ is 1 if π is the product of an even number of transpositions and -1 otherwise.
  - Equivalently, $\text{sgn}(\pi) = 1$ if the number of (i, j)s such that $i < j$ and $\pi(i) > \pi(j)$ is even.[a]
- $\det(A^G)$ contains $n!$ terms, many of which may be 0s.

---

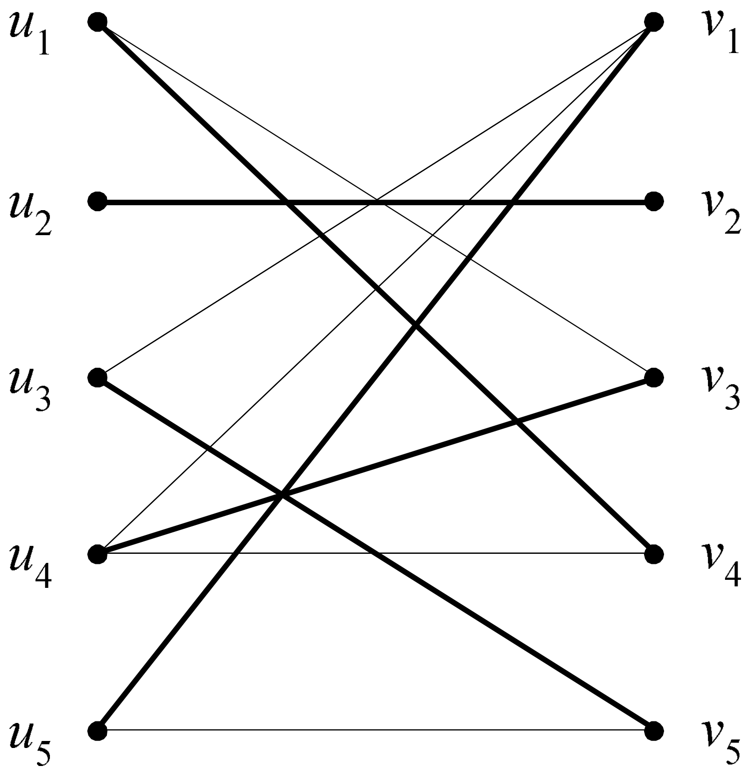
[a] Contributed by Mr. Hwan-Jeu Yu (`D95922028`) on May 1, 2008.

Determinant and Bipartite Perfect Matching

- In $\sum_\pi \text{sgn}(\pi) \prod_{i=1}^n A^G_{i,\pi(i)}$, note the following:
  - Each summand corresponds to a possible perfect matching π.
  - All of these summands $\prod_{i=1}^n A^G_{i,\pi(i)}$ are distinct monomials and *will not cancel.*
- $\det(A^G)$ is essentially an exhaustive enumeration.

Proposition 58 (Edmonds (1967)) *G has a perfect matching if and only if* $\det(A^G)$ *is not identically zero.*

Perfect Matching and Determinant (p. 481)



Perfect Matching and Determinant (concluded)

- The matrix is (p. 483)

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

- $\det(A^G) = -x_{14}x_{22}x_{35}x_{43}x_{51} + x_{13}x_{22}x_{35}x_{44}x_{51} + x_{14}x_{22}x_{31}x_{43}x_{55} - x_{13}x_{22}x_{31}x_{44}x_{55}$.
- Each nonzero term denotes a perfect matching, and vice versa.

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.
- If $\det(A^G) \equiv 0$, then it remains zero if we substitute *arbitrary* integers for the variables $x_{11}, \ldots, x_{nn}$.
- When $\det(A^G) \not\equiv 0$, what is the likelihood of obtaining a zero?

Number of Roots of a Polynomial

Lemma 59 (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

$$\frac{mdM^{m-1}}{M^m} = \frac{md}{M}. \tag{8}$$

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

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

  has a probability of $\leq md/M$ of being a root of p.
- Note that M is under our control!
  - One can raise M to lower the error probability, e.g.

Density Attack (concluded)

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

```
Choose $i_1, \ldots, i_m$ from $\{0, 1, \ldots, M-1\}$ randomly;
if $p(i_1, i_2, \ldots, i_m) \neq 0$ then
    return "$p$ is not identically zero";
else
    return "$p$ is (probably) identically zero";
end if
```

Analysis

- If $p(x_1, x_2, \ldots, x_m) \equiv 0$, the algorithm will always be correct as $p(i_1, i_2, \ldots, i_m) = 0$.
- Suppose $p(x_1, x_2, \ldots, x_m) \not\equiv 0$.
  - The algorithm will answer incorrectly with probability at most md/M by Eq. (8) on p. 490.
- We next return to the original problem of bipartite perfect matching.

A Randomized Bipartite Perfect Matching Algorithm[a]

```
Choose $n^2$ integers $i_{11}, \ldots, i_{nn}$ from $\{0, 1, \ldots, 2n^2 - 1\}$ randomly; {So $M = 2n^2$.}
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). According to Paul Erdős, Lovász wrote his first significant paper "at the ripe old age of 17."

Analysis

- If G has no perfect matchings, the algorithm will always be correct as $\det(A^G(i_{11}, \ldots, i_{nn})) = 0$.
- Suppose G has a perfect matching.
  - The algorithm will answer incorrectly with probability at most $md/M = 0.5$ with $m = n^2$, $d = 1$ and $M = 2n^2$ in Eq. (8) on p. 490.
- Run the algorithm *independently* k times.
- Output "G has no perfect matchings" if and only if *all* say no.
- The error probability is now reduced to at most 2^{-k}.

Lószló Lovász (1948–)

Remarks[a]

- Note that we are calculating

$$\text{prob}[\,\text{algorithm answers "no"}\,|\,G \text{ has no perfect matchings}\,],$$

$$\text{prob}[\,\text{algorithm answers "yes"}\,|\,G \text{ has a perfect matching}\,].$$

- We are *not* calculating[b]

$$\text{prob}[\,G \text{ has no perfect matchings}\,|\,\text{algorithm answers "no"}\,],$$

$$\text{prob}[\,G \text{ has a perfect matching}\,|\,\text{algorithm answers "yes"}\,].$$

---

[a] Thanks to a lively class discussion on May 1, 2008.

[b] *Numerical Recipes in C* (1988), "[As] we already remarked, statistics is *not* a branch of mathematics!"

But How Large Can det$(A^G(i_{11}, \ldots, i_{nn}))$ Be?

- It is at most
$$n!\left(2n^2\right)^n.$$
- Stirling's formula says $n! \sim \sqrt{2\pi n}\,(n/e)^n$.
- Hence
$$\log_2 \det(A^G(i_{11}, \ldots, i_{nn})) = O(n \log_2 n)$$
bits are sufficient for representing the determinant.
- We skip the details about how to make sure that all *intermediate* results are of polynomial sizes.

An Intriguing Question[a]

- Is there an $(i_{11}, \ldots, i_{nn})$ that will always give correct answers for the algorithm on p. 493?
- A theorem on p. 591 shows that such an $(i_{11}, \ldots, i_{nn})$ exists!
  - Whether it can be found efficiently is another matter.
- Once $(i_{11}, \ldots, i_{nn})$ is available, the algorithm can be made deterministic.

---

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

Randomization vs. Nondeterminism[a]

- What are the differences between randomized algorithms and nondeterministic algorithms?
- One can think of a randomized algorithm as a nondeterministic algorithm but with a probability associated with every guess/branch.
- So each computation path of a randomized algorithm has a probability associated with it.

---

[a] Contributed by Mr. Olivier Valery (`D01922033`) and Mr. Hasan Alhasan (`D01922034`) on November 27, 2012.

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 negatives**).

---

[a] Metropolis and Ulam (1949).

Monte Carlo Algorithms (continued)

- The algorithm makes a false negative with probability ≤ 0.5.[a]
  - Note this probability refers to[b]

$$\text{prob}[\,\text{algorithm answers “no”} \mid G \text{ has a perfect matching}\,]$$

not

$$\text{prob}[\,G \text{ has a perfect matching} \mid \text{algorithm answers “no”}\,].$$

---

[a] Equivalently, among the coin flip sequences, at most half of them lead to the wrong answer.

[b] In general, prob[algorithm answers “no” | input is a “yes” instance].

Monte Carlo Algorithms (concluded)

- This probability 0.5 is *not* over the space of all graphs or determinants, but *over* the algorithm's own coin flips.
  - It holds for *any* bipartite graph.

The Markov Inequality[a]

Lemma 60 *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 $x = i$.

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

---

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

Andrei Andreyevich Markov (1856–1922)

An Application of Markov's Inequality

- Suppose 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 $\leq 1/k$.

An Application of Markov's Inequality (concluded)

- By running this algorithm m times (the total running time is $mkT(n)$), we reduce the error probability to $\leq k^{-m}$.[a]
- 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 much worse than the previous algorithm's error probability of $\leq k^{-m}$ for the same amount of time.

---

[a] With the same input. Thanks to a question on December 7, 2010.

FSAT for k-SAT Formulas (p. 469)

- 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 "$\phi$ is satisfiable with $T$";
    else
        Let $c$ be an unsatisfied clause in $\phi$ 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 "$\phi$ 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 as of 2006 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 61 *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 $\hat{T} \models \phi$.
- Assume our starting T differs from $\hat{T}$ in i values.
  - Their Hamming distance is i.
  - Recall T is arbitrary.

---

[a] Papadimitriou (1991).

The Proof

- Let $t(i)$ denote the expected number of repetitions of the flipping step[a] until a satisfying truth assignment is found.
- 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 any other satisfying truth assignment, then we need to flip the coin at least once.
- We flip a coin 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.

---

[a] That is, Statement 7.

The Proof (continued)

- So we have at least 0.5 chance of moving closer to $\hat{T}$.
- 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)

- Now, put the necessary relations together:

$$\begin{aligned} t(0) &= 0, && (9) \\ t(i) &\leq \frac{t(i-1)+t(i+1)}{2}+1, \quad 0<i<n, && (10) \\ t(n) &\leq t(n-1)+1. && (11) \end{aligned}$$

- Technically, this is a one-dimensional random walk with an absorbing barrier at $i = 0$ and a reflecting barrier at $i = n$ (if we replace "$\leq$" with "$=$").[a]

---

[a] The proof in the textbook does exactly that. But a student pointed out difficulties with this proof technique on December 8, 2004. So our proof here uses the original inequalities.

The Proof (continued)

- Add up the relations for $2t(1), 2t(2), 2t(3), \ldots, 2t(n-1), t(n)$ to obtain[a]

$$\begin{aligned} & 2t(1) + 2t(2) + \cdots + 2t(n-1) + t(n) \\ \leq\ & t(0) + t(1) + 2t(2) + \cdots + 2t(n-2) + 2t(n-1) + t(n) \\ & +2(n-1) + 1. \end{aligned}$$

- Simplify it to yield

$$t(1) \leq 2n - 1. \tag{12}$$

---

[a] Adding up the relations for $t(1), t(2), t(3), \ldots, t(n-1)$ will also work, thanks to Mr. Yen-Wu Ti (D91922010).

The Proof (continued)

- Add up the relations for $2t(2), 2t(3), \ldots, 2t(n-1), t(n)$ to obtain

$$\begin{aligned} & 2t(2) + \cdots + 2t(n-1) + t(n) \\ \leq\ & t(1) + t(2) + 2t(3) + \cdots + 2t(n-2) + 2t(n-1) + t(n) \\ & +2(n-2) + 1. \end{aligned}$$

- Simplify it to yield

$$t(2) \leq t(1) + 2n - 3 \leq 2n - 1 + 2n - 3 = 4n - 4$$

by Eq. (12) on p. 514.

The Proof (continued)

- Continuing the process, we shall obtain
$$t(i) \leq 2in - i^2.$$
- The worst upper bound happens when $i = n$, in which case
$$t(n) \leq n^2.$$
- We conclude that
$$t(i) \leq t(n) \leq n^2$$
for $0 \leq i \leq n$.

The Proof (concluded)

- So the expected number of steps is at most n^2.
- The algorithm picks $r = 2n^2$.
  - This amounts to invoking the Markov inequality (p. 503) with $k = 2$, resulting in a probability of 0.5.[a]
- The proof does *not* yield a polynomial bound for 3SAT.[b]

---

[a] Recall p. 505.

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

Christos Papadimitriou (1949–)

Boosting the Performance

- We can pick $r = 2mn^2$ to have an error probability of
$$\leq \frac{1}{2m}$$
by Markov's inequality.
- Alternatively, with the same running time, we can run the "$r = 2n^2$" algorithm m times.
- The error probability is now reduced to
$$\leq 2^{-m}.$$

Primality Tests

- PRIMES asks if a number N is a prime.
- The classic algorithm tests if $k \mid N$ for $k = 2, 3, \ldots, \sqrt{N}$.
- But it runs in $\Omega(2^{(\log_2 N)/2})$ steps.

Primality Tests (concluded)

- Suppose $N = PQ$ is a product of 2 distinct primes.
- The probability of success of the density attack (p. 450) is
$$\approx \frac{2}{\sqrt{N}}$$
when $P \approx Q$.
- This probability is exponentially small in terms of the input length $\log_2 N$.

The Fermat Test for Primality

Fermat's "little" theorem (p. 453) suggests the following primality test for any given number N:

```
Pick a number $a$ randomly from $\{1, 2, \ldots, N-1\}$;
if $a^{N-1} \neq 1 \bmod N$ then
    return "$N$ is composite";
else
    return "$N$ is a prime";
end if
```

The Fermat Test for Primality (concluded)

- **Carmichael numbers** are composite numbers that will pass the Fermat test for *all* $a \in \{1, 2, \ldots, N - 1\}$.[a]
  - The Fermat test will return "N is a prime" for all Carmichael numbers N.
- Unfortunately, there are infinitely many Carmichael numbers.[b]
- In fact, the number of Carmichael numbers less than N exceeds $N^{2/7}$ for N large enough.
- So the Fermat test is an incorrect algorithm for PRIMES.

---

[a] Carmichael (1910). Lo (1994) mantions an investment strategy based on such numbers!

[b] Alford, Granville, and Pomerance (1992).

Square Roots Modulo a Prime

- Equation $x^2 = a \bmod p$ has at most two (distinct) roots by Lemma 57 (p. 458).
  - The roots are called **square roots**.
  - Numbers a with square roots *and* $\gcd(a, p) = 1$ are called **quadratic residues**.
    - They are

$$1^2 \bmod p, 2^2 \bmod p, \ldots, (p-1)^2 \bmod p.$$

- We shall show that a number either has two roots or has none, and testing which is the case is trivial.[a]

---

[a] But no efficient *deterministic* general-purpose square-root-extracting algorithms are known yet.

Euler's Test

Lemma 62 (Euler) *Let p be an odd prime and $a \neq 0 \bmod p$.*

1. *If*
$$a^{(p-1)/2} = 1 \bmod p,$$
then $x^2 = a \bmod p$ has two roots.

2. *If*
$$a^{(p-1)/2} \neq 1 \bmod p,$$
then
$$a^{(p-1)/2} = -1 \bmod p$$
and $x^2 = a \bmod p$ has no roots.

The Proof (continued)

- Let r be a primitive root of p.
- By Fermat's "little" theorem,

  $$r^{(p-1)/2}$$

  is a square root of 1.
- So

  $$r^{(p-1)/2} = 1 \text{ or } -1 \bmod p.$$
- But as r is a primitive root, $r^{(p-1)/2} \neq 1 \bmod p$.
- Hence

  $$r^{(p-1)/2} = -1 \bmod p.$$

The Proof (continued)

- Let $a = r^k \bmod p$ for some k.
- Then
$$1 = a^{(p-1)/2} = r^{k(p-1)/2} = \left[r^{(p-1)/2} \right]^k = (-1)^k \bmod p.$$
- So k must be even.
- Suppose $a = r^{2j}$ for some $1 \leq j \leq (p-1)/2$.
- Then $a^{(p-1)/2} = r^{j(p-1)} = 1 \bmod p$, and a's two *distinct* roots are $r^j, -r^j (= r^{j+(p-1)/2} \bmod p)$.
  - If $r^j = -r^j \bmod p$, then $2r^j = 0 \bmod p$, which implies $r^j = 0 \bmod p$, a contradiction.

The Proof (continued)

- As $1 \leq j \leq (p-1)/2$, there are $(p-1)/2$ such a's.
- Each such a has 2 distinct square roots.
- The square roots of all the a's are distinct.
  - The square roots of different a's must be different.
- Hence the set of *square roots* is $\{1, 2, \ldots, p-1\}$.
- As a result, $a = r^{2j}$, $1 \leq j \leq (p-1)/2$, exhaust all the quadratic residues.

The Proof (concluded)

- If $a = r^{2j+1}$, then it has no roots because all the square roots have been taken.
- Now,

$$a^{(p-1)/2} = \left[r^{(p-1)/2} \right]^{2j+1} = (-1)^{2j+1} = -1 \bmod p.$$

The Legendre Symbol[a] and Quadratic Residuacity Test

- By Lemma 62 (p. 525) $a^{(p-1)/2} \bmod p = \pm 1$ for $a \neq 0 \bmod p$.
- For odd prime p, define the **Legendre symbol** $(a \,|\, p)$ as

$$(a \,|\, p) = \begin{cases} 0 & \text{if } p \,|\, a, \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is a \textbf{quadratic nonresidue} modulo } p. \end{cases}$$

- Euler's test (p. 525) implies

$$a^{(p-1)/2} = (a \,|\, p) \bmod p$$

for any odd prime p and any integer a.

- Note that $(ab|p) = (a|p)(b|p)$.

---

[a] Andrien-Marie Legendre (1752–1833).