Function Problems Are Not Harder than Decision Problems If P = NP

Theorem 57 *Suppose that $P = NP$. Then, for every NP language L there exists a polynomial-time TM B that on input $x \in L$ outputs a certificate for x.*

- We are looking for a certificate in the sense of Proposition 31 (p. 274).
- That is, a certificate y for every $x \in L$ such that

$$(x, y) \in R,$$

where R is a polynomially decidable and polynomially balanced relation.

The Proof (concluded)

- Recall the algorithm for FSAT on p. 428.
- The reduction of Cook's Theorem L to SAT is a Levin reduction (p. 278).
- So there is a polynomial-time computable function R such that $x \in L$ iff $R(x) \in$ SAT.
- In fact, more is true: R maps a satisfying assignment of $R(x)$ into a certificate for x.
- Therefore, we can use the algorithm for FSAT to come up with an assignment for $R(x)$ and then map it back into a certificate for x.

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.

[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).

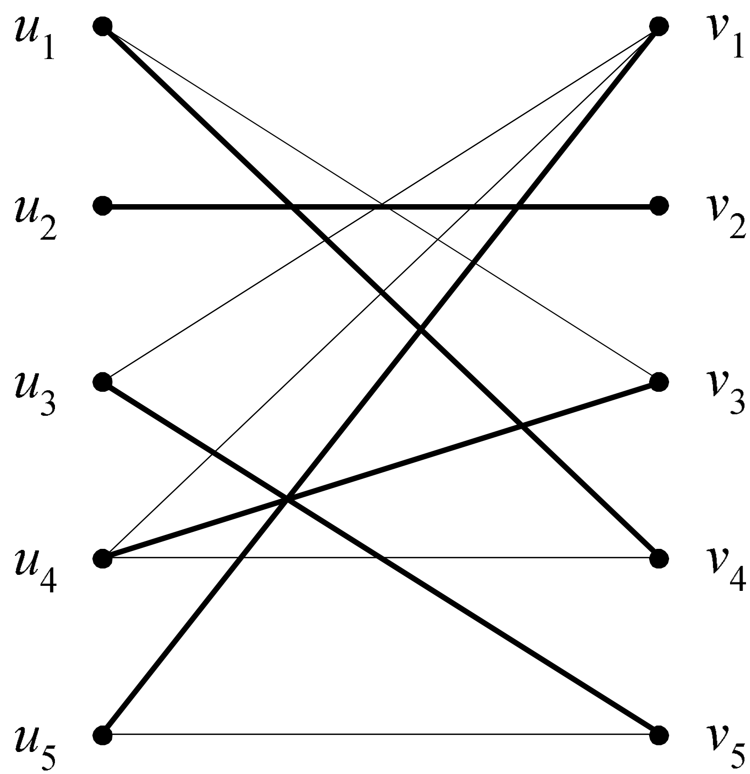
Bipartite Perfect Matching

- 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

- 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 variable x_{ij} if $(u_i, v_j) \in E$ and zero otherwise.

Symbolic Determinants (concluded)

- The **determinant** of A^G is

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

  - 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.
  - Equivalently, $\operatorname{sgn}(\pi) = 1$ if the number of (i, j)s such that $i < j)$ and $\pi(i) > \pi(j)$ is even.[a]

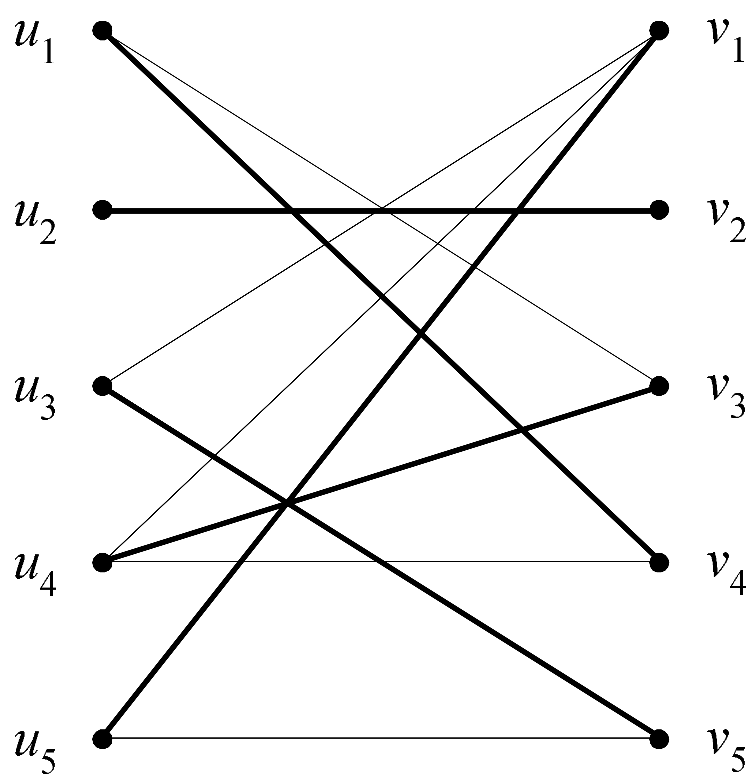
[a] Contributed by Mr. Hwan-Jeu Yu (D95922028) on May 1, 2008.

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.
  - As all variables appear only *once*, all of these summands are different monomials and will not cancel.
- It 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.*

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 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

$$\frac{mdM^{m-1}}{M^m} = \frac{md}{M}. \tag{6}$$

- 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.

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, ..., i_m from {0, 1, ..., M − 1} randomly;
if p(i_1, i_2, ..., i_m) ≠ 0 then
    return "p is not identically zero";
else
    return "p is identically zero";
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 $\{0, 1, \ldots, 2n^2 - 1\}$ 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). 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.
- Suppose G has a perfect matching.
  - The algorithm will answer incorrectly with probability at most $n^2 d/(2n^2) = 0.5$ with $d = 1$ in Eq. (6) on p. 448.
  - 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.

Analysis (concluded)[a]

- Note that we are calculating

$$\text{prob}[\,\text{algorithm answers "yes"} \mid G \text{ has a perfect matching}\,],$$
$$\text{prob}[\,\text{algorithm answers "no"} \mid G \text{ has no perfect matchings}\,].$$

- We are *not* calculating

$$\text{prob}[\,G \text{ has a perfect matching} \mid \text{algorithm answers "yes"}\,],$$
$$\text{prob}[\,G \text{ has no perfect matchings} \mid \text{algorithm answers "no"}\,].$$

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

Lószló Lovász (1948–)

Perfect Matching for General Graphs

- Page 439 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 58 (p. 443):

Proposition 60 *G has a perfect matching if and only if* $\det(T^G)$ *is not identically zero.*

[a]William Thomas Tutte (1917–2002).

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**).

[a] Metropolis and Ulam (1949).

Monte Carlo Algorithms (concluded)

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

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

- 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.

False Positives and False Negatives in Human Behavior?[a]

- "[Men] tend to misinterpret innocent friendliness as a sign that women are [$\cdots$] interested in them."
  - A false positive.
- "[Women] tend to undervalue signs that a man is interested in a committed relationship."
  - A false negative.

[a] "Don't misunderestimate yourself." *The Economist*, 2006.

The Markov Inequality[a]

Lemma 61 *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).

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. 427)

- 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, ..., r do
    if T ⊨ φ 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 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 62 *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. 460) 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.

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 $\Omega(2^{n/2})$ steps, where $n = |\, N \,| = \log_2 N$.

The Density Attack for PRIMES

```
Pick k ∈ {2, ..., N − 1} randomly; {Assume N > 2.}
if k | N then
    return "N is composite";
else
    return "N is a prime";
end if
```


Analysis[a]

- Suppose $N = PQ$, a product of 2 primes.
- The probability of success is

$$< 1 - \frac{\phi(N)}{N} = 1 - \frac{(P-1)(Q-1)}{PQ} = \frac{P+Q-1}{PQ}.$$

- In the case where $P \approx Q$, this probability becomes

$$< \frac{1}{P} + \frac{1}{Q} \approx \frac{2}{\sqrt{N}}.$$

- This probability is exponentially small.

[a]See also p. 410.

The Fermat Test for Primality

Fermat's "little" theorem on p. 412 suggests the following primality test for any given number p:

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)

- Unfortunately, there are composite numbers called **Carmichael numbers** that will pass the Fermat test for *all* $a \in \{1, 2, \ldots, N-1\}$.[a]
- 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.

[a] Carmichael (1910).
[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 56 (p. 417).
  - 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 one is true is trivial.
- There are no known efficient *deterministic* algorithms to find the roots, however.

Euler's Test

Lemma 63 (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.*

- Let r be a primitive root of p.
- By Fermat's "little" theorem, $r^{(p-1)/2}$ is a square root of 1, so $r^{(p-1)/2} = 1 \bmod p$ or $r^{(p-1)/2} = -1 \bmod p$.
- 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 its 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 $\{1, 2, \ldots, p-1\}$.
  - Because there are $(p-1)/2$ such a's and each a has two square roots.
- As a result, $a = r^{2j}$, $1 \leq j \leq (p-1)/2$, are 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 63 (p. 481) $a^{(p-1)/2} \bmod p = \pm 1$ for $a \neq 0 \bmod p$.
- For odd prime p, define the **Legendre symbol** $(a \,|\, p)$ as

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

- Euler's test 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).

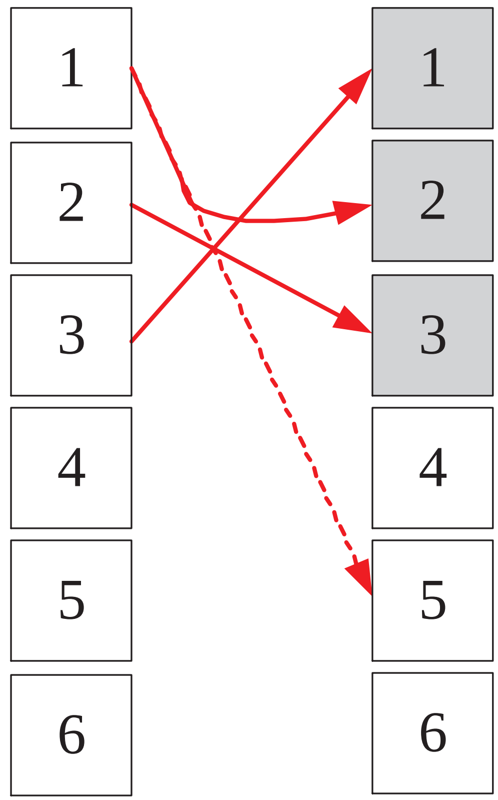
Gauss's Lemma

Lemma 64 (Gauss) *Let p and q be two odd primes. Then $(q|p) = (-1)^m$, where m is the number of residues in $R = \{\, iq \bmod p : 1 \leq i \leq (p-1)/2 \,\}$ that are greater than $(p-1)/2$.*

- All residues in R are distinct.
  - If $iq = jq \bmod p$, then $p|(j-i)\, q$ or $p|q$.
- No two elements of R add up to p.
  - If $iq + jq = 0 \bmod p$, then $p|(i+j)$ or $p|q$.
  - But neither is possible.

The Proof (continued)

- Consider the set R' of residues that result from R if we replace each of the m elements $a \in R$ such that $a > (p-1)/2$ by $p - a$.
  - This is equivalent to performing $-a \bmod p$.
- All residues in R' are now at most $(p-1)/2$.
- In fact, $R' = \{1, 2, \ldots, (p-1)/2\}$ (see illustration next page).
  - Otherwise, two elements of R would add up to p, which has been shown to be impossible.



$p = 7$ and $q = 5$.

The Proof (concluded)

- Alternatively, $R' = \{\pm iq \bmod p : 1 \leq i \leq (p-1)/2\}$, where exactly m of the elements have the minus sign.
- Take the product of all elements in the two representations of R'.
- So $[(p-1)/2]! = (-1)^m q^{(p-1)/2} [(p-1)/2]! \bmod p$.
- Because $\gcd([(p-1)/2]!, p) = 1$, the above implies

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