The Primality Problem

- An integer p is **prime** if $p > 1$ and all positive numbers other than 1 and p itself cannot divide it.
- PRIMES asks if an integer N is a prime number.
- Dividing N by $2, 3, \ldots, \sqrt{N}$ is *not* efficient.
  - The length of N is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
- A polynomial-time algorithm for PRIMES was not found until 2002 by Agrawal, Kayal, and Saxena!
- We will focus on efficient "probabilistic" algorithms for PRIMES (used in *Mathematica*, e.g.).

```
if n = a^b for some a, b > 1 then
  return "composite";
end if
for r = 2, 3, ..., n − 1 do
  if gcd(n, r) > 1 then
    return "composite";
  end if
  if r is a prime then
    Let q be the largest prime factor of r − 1;
    if q ≥ 4√r log n and n^((r−1)/q) ≠ 1 mod r then
      break; {Exit the for-loop.}
    end if
  end if
end for{r − 1 has a prime factor q ≥ 4√r log n.}
for a = 1, 2, ..., 2√r log n do
  if (x − a)^n ≠ (x^n − a) mod (x^r − 1) in Z_n[x] then
    return "composite";
  end if
end for
return "prime"; {The only place with "prime" output.}
```

The Primality Problem (concluded)

- $\mathrm{NP} \cap \mathrm{coNP}$ is the class of problems that have succinct certificates and succinct disqualifications.
  - Each "yes" instance has a succinct certificate.
  - Each "no" instance has a succinct disqualification.
  - No instances have both.
- We will see that PRIMES $\in \mathrm{NP} \cap \mathrm{coNP}$.
  - In fact, PRIMES $\in \mathrm{P}$ as mentioned earlier.

Primitive Roots in Finite Fields

Theorem 48 (Lucas and Lehmer (1927)) [a] *A number* $p > 1$ *is prime if and only if there is a number* $1 < r < p$ *(called the* **primitive root** *or* **generator***) 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* $p - 1$.

- We will prove the theorem later.

---

[a] François Edouard Anatole Lucas (1842–1891); Derrick Henry Lehmer (1905–1991).

Derrick Lehmer (1905–1991)

Pratt's Theorem

Theorem 49 (Pratt (1975)) PRIMES $\in$ *NP* $\cap$ *coNP*.

- PRIMES is in coNP because a succinct disqualification is a divisor.
- Suppose p is a prime.
- p's certificate includes the r in Theorem 48 (p. 396).
- Use recursive doubling to check if $r^{p-1} = 1 \bmod p$ in time polynomial in the length of the input, $\log_2 p$.
  - $r, r^2, r^4, \ldots \bmod p$, a total of $\sim \log_2 p$ steps.

The Proof (concluded)

- We also need all *prime* divisors of $p-1$: $q_1, q_2, \ldots, q_k$.
- Checking $r^{(p-1)/q_i} \neq 1 \bmod p$ is also easy.
- Checking $q_1, q_2, \ldots, q_k$ are all the divisors of $p-1$ is easy.
- We still need certificates for the primality of the q_i's.
- The complete certificate is recursive and tree-like:

$$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)).$$

- $C(p)$ can also be checked in polynomial time.
- We next prove that $C(p)$ is succinct.

The Succinctness of the Certificate

Lemma 50 *The length of $C(p)$ is at most quadratic at $5 \log_2^2 p$.*

- This claim holds when $p = 2$ or $p = 3$.
- In general, $p - 1$ has $k \leq \log_2 p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
  - Reason: $2^k \leq \prod_{i=1}^{k} q_i \leq p - 1$.
- $C(p)$ requires: 2 parentheses and $2k < 2 \log_2 p$ separators (length at most $2 \log_2 p$ long), r (length at most $\log_2 p$), $q_1 = 2$ and its certificate 1 (length at most 5 bits), the q_i's (length at most $2 \log_2 p$), and the $C(q_i)$s.

The Proof (concluded)

- $C(p)$ is succinct because, by induction,

$$
\begin{aligned}
|C(p)| &\leq 5\log_2 p + 5 + 5\sum_{i=2}^{k} \log_2^2 q_i \\
&\leq 5\log_2 p + 5 + 5\left(\sum_{i=2}^{k} \log_2 q_i\right)^2 \\
&\leq 5\log_2 p + 5 + 5\log_2^2 \frac{p-1}{2} \\
&< 5\log_2 p + 5 + 5(\log_2 p - 1)^2 \\
&= 5\log_2^2 p + 10 - 5\log_2 p \leq 5\log_2^2 p
\end{aligned}
$$

  for $p \geq 4$.

A Certificate for 23[a]

- As 7 is a primitive root modulo 23 and $22 = 2 \times 11$, so

$$C(23) = (7, 2, C(2), 11, C(11)).$$

- As 2 is a primitive root modulo 11 and $10 = 2 \times 5$, so

$$C(11) = (2, 2, C(2), 5, C(5)).$$

- As 2 is a primitive root modulo 5 and $4 = 2^2$, so

$$C(5) = (2, 2, C(2)).$$

- In summary,

$$C(23) = (7, 2, C(2), 11, (2, 2, C(2), 5, (2, 2, C(2)))).$$

---

[a]Thanks to a lively discussion on April 24, 2008.

Basic Modular Arithmetics[a]

- Let $m, n \in \mathbb{Z}^+$.
- $m | n$ means m divides n and m is n's **divisor**.
- We call the numbers $0, 1, \ldots, n-1$ the **residue** modulo n.
- The **greatest common divisor** of m and n is denoted $\gcd(m, n)$.
- The r in Theorem 48 (p. 396) is a primitive root of p.
- We now prove the existence of primitive roots and then Theorem 48.

---

[a] Carl Friedrich Gauss.

Euler's[a] Totient or Phi Function

- Let

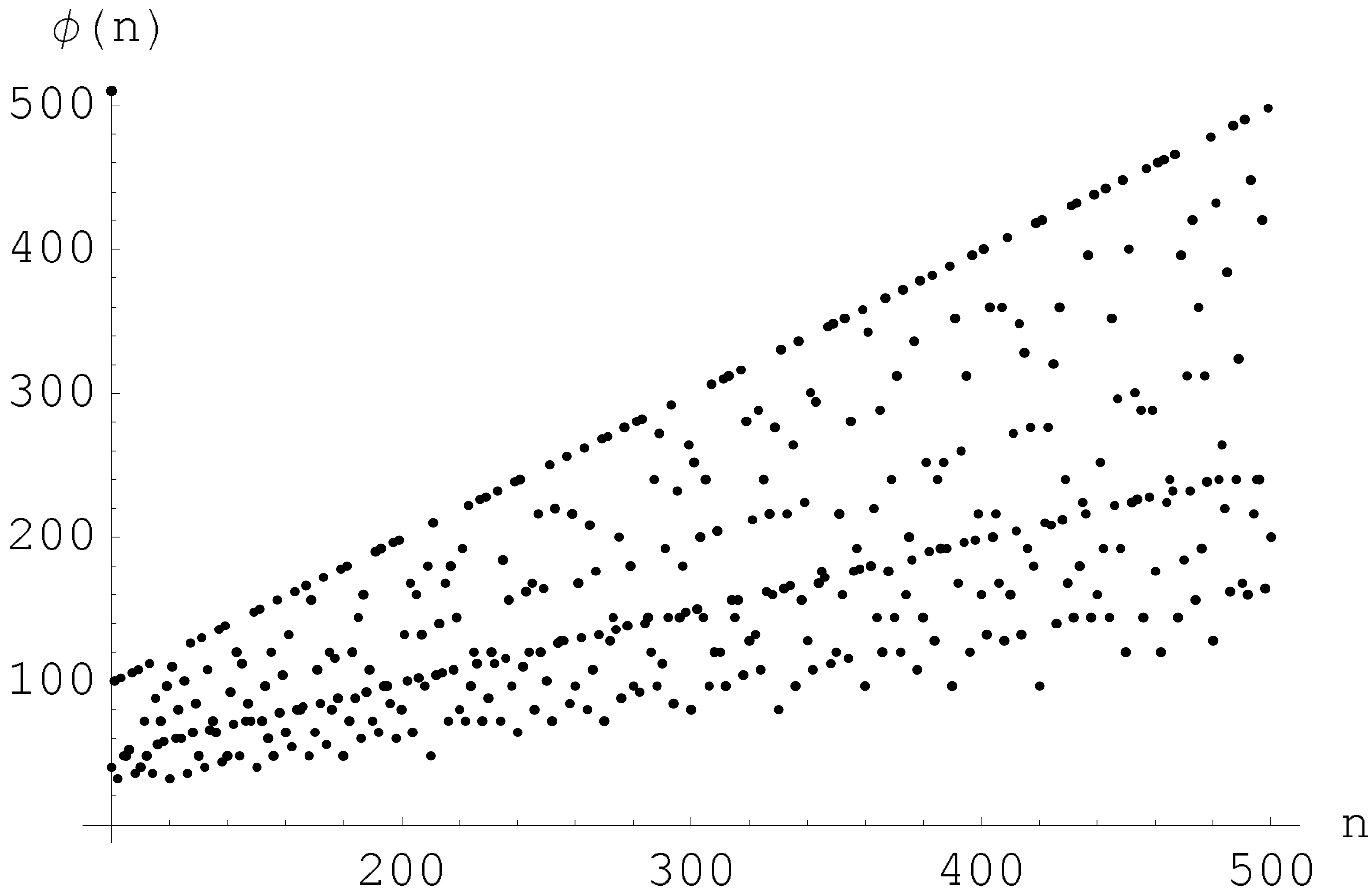
$$\Phi(n) = \{m : 1 \leq m < n, \gcd(m, n) = 1\}$$

be the set of all positive integers less than n that are prime to n (Z_n^* is a more popular notation).

  - $\Phi(12) = \{1, 5, 7, 11\}$.

- Define **Euler's function** of n to be $\phi(n) = |\Phi(n)|$.
- $\phi(p) = p - 1$ for prime p, and $\phi(1) = 1$ by convention.
- Euler's function is not expected to be easy to compute without knowing n's factorization.

[a] Leonhard Euler (1707–1783).



ϕ(n)
500
400
300
200
100
n
200
300
400
500

Two Properties of Euler's Function

The inclusion-exclusion principle[a] can be used to prove the following.

Lemma 51 $\phi(n) = n \prod_{p \mid n} (1 - \frac{1}{p})$.

- If $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_\ell}$ is the prime factorization of n, then

$$\phi(n) = n \prod_{i=1}^{\ell} \left(1 - \frac{1}{p_i} \right).$$

Corollary 52 $\phi(mn) = \phi(m)\,\phi(n)$ *if* $\gcd(m, n) = 1$.

[a] See my *Discrete Mathematics* lecture notes.

A Key Lemma

Lemma 53 $\sum_{m|n} \phi(m) = n$.

- Let $\prod_{i=1}^{\ell} p_i^{k_i}$ be the prime factorization of n and consider

$$\prod_{i=1}^{\ell} [\, \phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i}) \,]. \tag{4}$$

- Equation (4) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 51.

- Expand Eq. (4) to yield

$$\sum_{k'_1 \leq k_1, \ldots, k'_\ell \leq k_\ell} \prod_{i=1}^{\ell} \phi(p_i^{k'_i}).$$

The Proof (concluded)

- By Corollary 52 (p. 406),

$$\prod_{i=1}^{\ell} \phi(p_i^{k'_i}) = \phi\left(\prod_{i=1}^{\ell} p_i^{k'_i}\right).$$

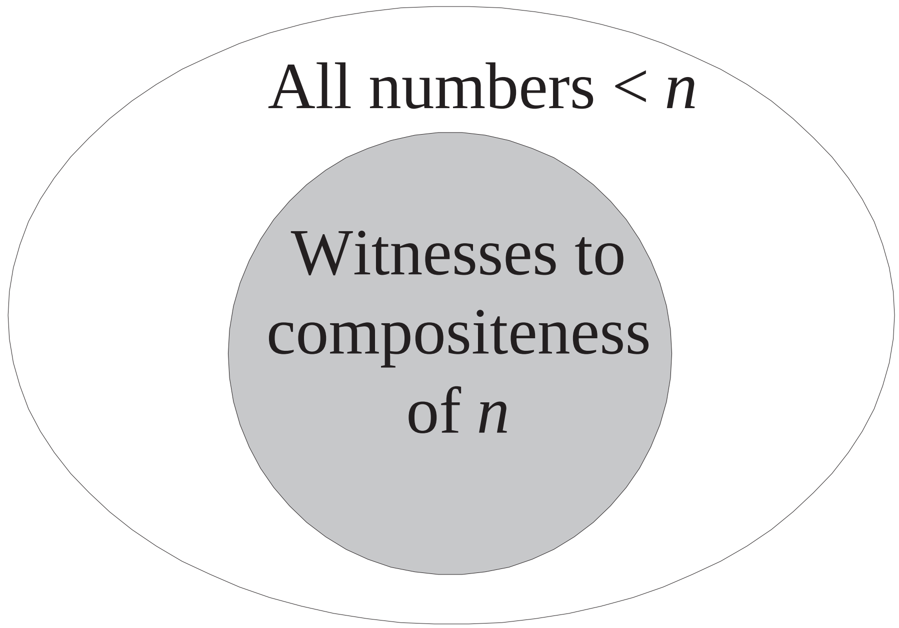
- So Eq. (4) becomes

$$\sum_{k'_1 \leq k_1, \ldots, k'_\ell \leq k_\ell} \phi\left(\prod_{i=1}^{\ell} p_i^{k'_i}\right).$$

- Each $\prod_{i=1}^{\ell} p_i^{k'_i}$ is a unique divisor of $n = \prod_{i=1}^{\ell} p_i^{k_i}$.
- Equation (4) becomes

$$\sum_{m \mid n} \phi(m).$$

The Density Attack for PRIMES



- It works, but does it work well?
- The ratio of numbers $\leq n$ relatively prime to n (the white area) is $\phi(n)/n$.

The Density Attack for PRIMES (concluded)

- When $n = pq$, where p and q are distinct primes,

$$\frac{\phi(n)}{n} = \frac{pq - p - q + 1}{pq} > 1 - \frac{1}{q} - \frac{1}{p}.$$

- So the ratio of numbers $\leq n$ *not* relatively prime to n (the grey area) is $< (1/q) + (1/p)$.
  - The "density attack" has probability $< 2/\sqrt{n}$ of factoring $n = pq$ when $p \sim q = O(\sqrt{n}\,)$.
  - The "density attack" to factor $n = pq$ hence takes $\Omega(\sqrt{n})$ steps on average when $p \sim q = O(\sqrt{n}\,)$.
  - This running time is exponential: $\Omega(2^{0.5 \log_2 n})$.

The Chinese Remainder Theorem

- Let $n = n_1 n_2 \cdots n_k$, where n_i are pairwise relatively prime.
- For any integers $a_1, a_2, \ldots, a_k$, the set of simultaneous equations

$$
\begin{aligned}
x &= a_1 \bmod n_1, \\
x &= a_2 \bmod n_2, \\
&\vdots \\
x &= a_k \bmod n_k,
\end{aligned}
$$

has a unique solution modulo n for the unknown x.

Fermat's "Little" Theorem[a]

Lemma 54 *For all* $0 < a < p$, $a^{p-1} = 1 \bmod p$.

- Consider $a\Phi(p) = \{am \bmod p : m \in \Phi(p)\}$.
- $a\Phi(p) = \Phi(p)$.
  - $a\Phi(p) \subseteq \Phi(p)$ as a remainder must be between 0 and $p-1$.
  - Suppose $am = am' \bmod p$ for $m > m'$, where $m, m' \in \Phi(p)$.
  - That means $a(m-m') = 0 \bmod p$, and p divides a or $m - m'$, which is impossible.

[a] Pierre de Fermat (1601–1665).

The Proof (concluded)

- Multiply all the numbers in $\Phi(p)$ to yield $(p-1)!$.
- Multiply all the numbers in $a\Phi(p)$ to yield $a^{p-1}(p-1)!$.
- As $a\Phi(p) = \Phi(p)$, $a^{p-1}(p-1)! = (p-1)! \bmod p$.
- Finally, $a^{p-1} = 1 \bmod p$ because $p \not| (p-1)!$.

The Fermat-Euler Theorem[a]

Corollary 55 *For all $a \in \Phi(n)$, $a^{\phi(n)} = 1 \bmod n$.*

- The proof is similar to that of Lemma 54 (p. 412).
- Consider $a\Phi(n) = \{am \bmod n : m \in \Phi(n)\}$.
- $a\Phi(n) = \Phi(n)$.
  - $a\Phi(n) \subseteq \Phi(n)$ as a remainder must be between 0 and $n - 1$ and relatively prime to n.
  - Suppose $am = am' \bmod n$ for $m' < m < n$, where $m, m' \in \Phi(n)$.
  - That means $a(m - m') = 0 \bmod n$, and n divides a or $m - m'$, which is impossible.

[a] Proof by Mr. Wei-Cheng Cheng (R93922108) on November 24, 2004.

The Proof (concluded)

- Multiply all the numbers in $\Phi(n)$ to yield $\prod_{m \in \Phi(n)} m$.
- Multiply all the numbers in $a\Phi(n)$ to yield $a^{\Phi(n)} \prod_{m \in \Phi(n)} m$.
- As $a\Phi(n) = \Phi(n)$,

$$\prod_{m \in \Phi(n)} m = a^{\Phi(n)} \left(\prod_{m \in \Phi(n)} m \right) \bmod n.$$

- Finally, $a^{\Phi(n)} = 1 \bmod n$ because $n \not| \prod_{m \in \Phi(n)} m$.

An Example

- As $12 = 2^2 \times 3$,
$$\phi(12) = 12 \times \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 4.$$
- In fact, $\Phi(12) = \{1, 5, 7, 11\}$.
- For example,
$$5^4 = 625 = 1 \bmod 12.$$

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 modulo p, 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 56 *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 56 (p. 417).
- Let s be a residue of exponent k.
- $1, s, s^2, \ldots, s^{k-1}$ are distinct modulo p.
  - Otherwise, $s^i = s^j \bmod p$ with $i < j$.
  - Then $s^{j-i} = 1 \bmod p$ with $j - i < k$, a contradiction.
- As all these k distinct numbers satisfy $x^k = 1 \bmod p$, they comprise *all* solutions of $x^k = 1 \bmod p$.

Size of $R(k)$ (continued)

- But do all of them have exponent k (i.e., $R(k) = k$)?
- 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 52 (p. 406).

- 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 48 (p. 396).

A Few Calculations

- Let $p = 13$.
- From p. 414, we know $\phi(p-1) = 4$.
- Hence $R(12) = 4$.
- Indeed, there are 4 primitive 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 48 (p. 396)

- 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. 414).
- Because p is not a prime, $\phi(p) < p - 1$.
- Let k be the smallest integer such that $r^k = 1 \bmod p$.
- Note that $k \mid \phi(p)$ (p. 417).
- 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.
- 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.
    * It is at most $\sum_{i,j} d_{ij}$.
- 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).