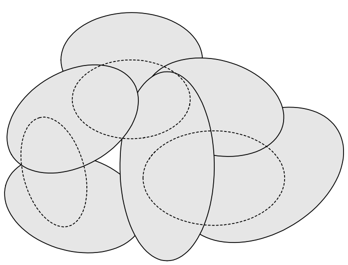
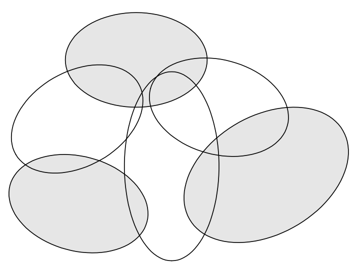
Set-Related Problems

- We are given a family $F = \{S_1, S_2, \ldots, S_n\}$ of subsets of a finite set U and a budget B.
- SET COVERING asks if there exists a set of B sets in F whose union is U.
- SET PACKING asks if there are B disjoint sets in F.
- Assume $|U| = 3m$ for some $m \in \mathbb{N}$ and $|S_i| = 3$ for all i.
- EXACT COVER BY 3-SETS asks if there are m sets in F that are disjoint and have U as their union.

Corollary 43 SET COVERING, SET PACKING, *and* EXACT COVER BY 3-SETS *are all NP-complete.*



SET COVERING



SET PACKING

The KNAPSACK Problem

- There is a set of n items.
- Item i has value $v_i \in \mathbb{Z}^+$ and weight $w_i \in \mathbb{Z}^+$.
- We are given $K \in \mathbb{Z}^+$ and $W \in \mathbb{Z}^+$.
- KNAPSACK asks if there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq K$.
  - We want to achieve the maximum satisfaction within the budget.

KNAPSACK Is NP-Complete

- KNAPSACK $\in$ NP: Guess an S and verify the constraints.
- We assume $v_i = w_i$ for all i and $K = W$.
- KNAPSACK now asks if a subset of $\{w_1, w_2, \ldots, w_n\}$ adds up to exactly K.
  - Picture yourself as a radio DJ.
  - Or a person trying to control the calories intake.
- We shall reduce EXACT COVER BY 3-SETS to KNAPSACK.

The Proof (continued)

- We are given a family $F = \{S_1, S_2, \ldots, S_n\}$ of size-3 subsets of $U = \{1, 2, \ldots, 3m\}$.
- EXACT COVER BY 3-SETS asks if there are m disjoint sets in F that cover the set U.
- Think of a set as a bit vector in $\{0,1\}^{3m}$.
  - 001100010 means the set $\{3, 4, 8\}$, and 110010000 means the set $\{1, 2, 5\}$.
- Our goal is $\overbrace{11\cdots1}^{3m}$.

The Proof (continued)

- A bit vector can also be considered as a binary *number*.
- Set union resembles addition.
  - $001100010 + 110010000 = 111110010$, which denotes the set $\{1, 2, 3, 4, 5, 8\}$, as desired.
- Trouble occurs when there is *carry*.
  - $001100010 + 001110000 = 010010010$, which denotes the set $\{2, 5, 8\}$, not the desired $\{3, 4, 5, 8\}$.

The Proof (continued)

- Carry may also lead to a situation where we obtain our solution $11\cdots1$ with more than m sets in F.
  - $001100010 + 001110000 + 101100000 + 000001101 = 111111111$.
  - But this "solution" $\{1, 3, 4, 5, 6, 7, 8, 9\}$ does not correspond to an exact cover.
  - And it uses 4 sets instead of the required 3.[a]
- To fix this problem, we enlarge the base just enough so that there are no carries.
- Because there are n vectors in total, we change the base from 2 to $n + 1$.

[a] Thanks to a lively class discussion on November 20, 2002.

The Proof (continued)

- Set v_i to be the $(n+1)$-ary number corresponding to the bit vector encoding S_i.
- Now in base $n + 1$, if there is a set S such that $\sum_{v_i \in S} v_i = \overbrace{11\cdots1}^{3m}$, then every bit position must be contributed by exactly one v_i and $|S| = m$.
- Finally, set

$$K = \sum_{j=0}^{3m-1} (n+1)^j = \overbrace{11\cdots1}^{3m} \quad \text{(base } n+1\text{)}.$$

The Proof (concluded)

- Suppose F admits an exact cover, say $\{S_1, S_2, \ldots, S_m\}$.
- Then picking $S = \{v_1, v_2, \ldots, v_m\}$ clearly results in

$$v_1 + v_2 + \cdots + v_m = \overbrace{11 \cdots 1}^{3m}.$$

- On the other hand, suppose there exists an S such that $\sum_{v_i \in S} v_i = \overbrace{11 \cdots 1}^{3m}$ in base $n+1$.
- The no-carry property implies that $|S| = m$ and $\{S_i : v_i \in S\}$ is an exact cover.

BIN PACKINGS

- We are given N positive integers $a_1, a_2, \ldots, a_N$, an integer C (the capacity), and an integer B (the number of bins).
- BIN PACKING asks if these numbers can be partitioned into B subsets, each of which has total sum at most C.
- Think of packing bags at the check-out counter.

Theorem 44 BIN PACKING *is NP-complete.*

INTEGER PROGRAMMING Is NP-Complete[a]

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.
  - SET COVERING can be expressed by the inequalities $Ax \geq \vec{1}$, $\sum_{i=1}^{n} x_i \leq B$, $0 \leq x_i \leq 1$, where
    * x_i is one if and only if S_i is in the cover.
    * A is the matrix whose columns are the bit vectors of the sets $S_1, S_2, \ldots$.
    * $\vec{1}$ is the vector of 1s.

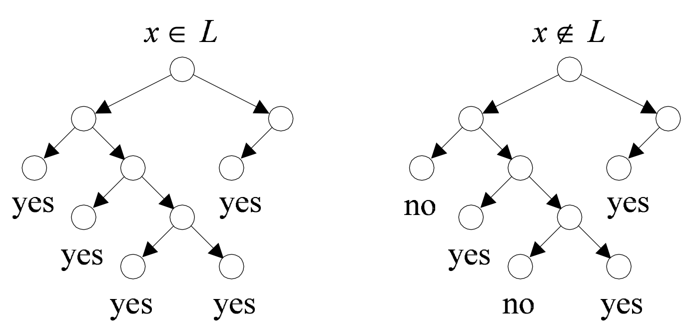
[a]Papadimitriou (1981).

coNP

- NP is the class of problems that have succinct certificates (recall Proposition 34 on p. 250).
- coNP is the class of problems that have succinct disqualifications:
  - A "no" instance of a problem in coNP possesses a short proof of its being a "no" instance.
  - Only "no" instances have such proofs.
- Clearly $\mathrm{P} \subseteq \mathrm{coNP}$.
- It is not known if $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.
  - Contrast this with $\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$.

coNP as Decision Problems

- Suppose L is a coNP problem.
- There exists a polynomial-time nondeterministic algorithm M such that:
  - If $x \in L$, then $M(x) =$ "yes" for all computation paths.
  - If $x \notin L$, then $M(x) =$ "no" for some computation path.



Some coNP Problems

- VALIDITY $\in$ coNP.
  - If ϕ is not valid, it can be disqualified very succinctly: a truth assignment that does not satisfy it.
- SAT COMPLEMENT $\in$ coNP.
  - The disqualification is a truth assignment that satisfies it.
- HAMILTONIAN PATH COMPLEMENT $\in$ coNP.
  - The disqualification is a Hamiltonian path.

An Alternative Characterization of coNP

Proposition 45 *Let $L \subseteq \Sigma^*$ be a language. Then $L \in$ coNP if and only if there is a polynomially decidable and polynomially balanced relation R such that*

$$L = \{x : \forall y\, (x, y) \in R\}.$$

- $\bar{L} = \{x : (x, y) \in \neg R \text{ for some } y\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in$ NP by Proposition 34 (p. 250).
- Hence $L \in$ coNP by definition.

coNP Completeness

Proposition 46 *L is NP-complete if and only if its complement $\bar{L} = \Sigma^* - L$ is coNP-complete.*

Proof ($\Rightarrow$; the $\Leftarrow$ part is symmetric)

- Let $\bar{L}'$ be any coNP language.
- Hence $L' \in$ NP.
- Let R be the reduction from L' to L.
- So $x \in L'$ if and only if $R(x) \in L$.
- So $x \in \bar{L}'$ if and only if $R(x) \in \bar{L}$.
- R is a reduction from $\bar{L}'$ to $\bar{L}$.

Some coNP-Complete Problems

- SAT COMPLEMENT is coNP-complete.
  - SAT COMPLEMENT is the complement of SAT.
- VALIDITY is coNP-complete.
  - ϕ is valid if and only if $\neg\phi$ is not satisfiable.
  - The reduction from SAT COMPLEMENT to VALIDITY is hence easy.
- HAMILTONIAN PATH COMPLEMENT is coNP-complete.

Possible Relations between P, NP, coNP

- P = NP = coNP.
- NP = coNP but P $\neq$ NP.
- NP $\neq$ coNP and P $\neq$ NP (current "consensus").

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.).

Primitive Roots in Finite Fields

Theorem 47 (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 $p-1$.*

- The above theorem can be used to test efficiently primes of the form $2^m + 1$.
- We will prove the theorem later.

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

Pratt's Theorem

Theorem 48 (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 47 (p. 327).
- Use recursive doubling to check if $r^{p-1} = 1 \bmod p$ in time polynomial in the length of the input, $\log_2 p$.
- 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.

The Proof (concluded)

- 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 49 *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 $k < \log_2 p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
- $C(p)$ 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

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

for $p \geq 4$.

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 47 (p. 327) is a primitive root of p.
- We now prove the existence of primitive roots and then Theorem 47.

[a]Carl Friedrich Gauss (1777–1855).

Euler's[a] Totient or Phi Function

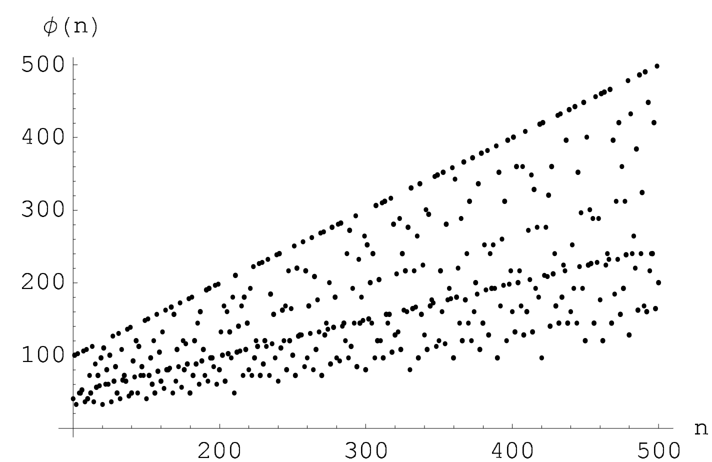
- Let

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

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).

eulerphi.nb



Two Properties of Euler's Function

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

Lemma 50 $\phi(n) = n \prod_{p|n} (1 - \frac{1}{p})$.

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

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

Corollary 51 $\phi(mn) = \phi(m)\phi(n)$ *if* $\gcd(m, n) = 1$.

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

A Key Lemma

Lemma 52 $\sum_{m|n} \phi(m) = n$.

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

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

- Equation (4) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 50.
- 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)

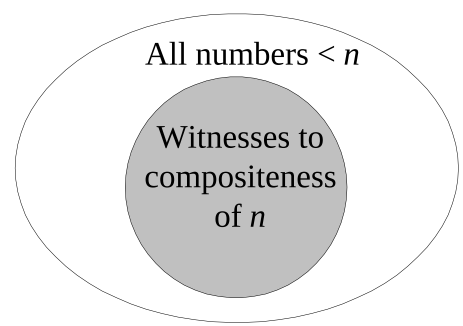
- By Corollary 51 (p. 335),

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

- 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

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

The Density Attack for PRIMES



- It works, but does it work well?

Factorization and Euler's Function

- The ratio of numbers $\leq n$ relatively prime to n is $\phi(n)/n$.
- When $n = pq$, where p and q are distinct primes,
$$\frac{\phi(n)}{n} = \frac{pq - p - q + 1}{pq} > 1 - \frac{1}{q} - \frac{1}{p}.$$
  - 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 53 *For all* $0 < a < p$, $a^{p-1} = 1 \bmod p$.

- Consider $a\Phi(p) = \{am \bmod p : m \in \Phi(p)\}$.
- $a\Phi(p) = \Phi(p)$.
  - 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.
- Hence $(p-1)! = a^{p-1}(p-1)! \bmod p$.
- Finally, $a^{p-1} = 1 \bmod p$ because $p \not| (p-1)!$.

[a] Pierre de Fermat (1601–1665).

The Fermat-Euler Theorem

Corollary 54 *For all* $a \in \Phi(n)$, $a^{\phi(n)} = 1 \bmod n$.

- 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, 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 55 *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 55 on p. 343.
- 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

$$(s^\ell)^{k/d} = 1 \bmod p.$$

- 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 51 on p. 335.

- 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 47 (p. 327).

A Few Calculations

- Let $p=13$.
- From p. 342, 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 47 (p. 327)

- Suppose p is not a prime.
- We proceed to show that no primitive roots exist.
- Suppose r is a primitive root.
- Suppose $r^{p-1}=1 \bmod p$, the 1st condition of the primitive root on p. 327 (note $\gcd(r,p)=1$).
- We will show that the 2nd condition must be violated.
- $r^{\phi(p)}=1 \bmod p$ by the Fermat-Euler theorem (p. 342).
- Because p is not a prime, $\phi(p)<p-1$.

The Other Direction of Theorem 47 (concluded)

- 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 of the primitive root on p. 327.

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 $\leq 2n$ calls to the algorithm for SAT.
- 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).

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 answer 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, ..., 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} ≤ 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.
- 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).