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

```
$t := \epsilon$; {Truth assignment.}
if $\phi \in$ SAT then
    for $i = 1, 2, \ldots, n$ do
        if $\phi[\, x_i = \texttt{true} \,] \in$ SAT then
            $t := t \cup \{\, x_i = \texttt{true} \,\}$;
            $\phi := \phi[\, x_i = \texttt{true} \,]$;
        else
            $t := t \cup \{\, x_i = \texttt{false} \,\}$;
            $\phi := \phi[\, x_i = \texttt{false} \,]$;
        end if
    end for
    return $t$;
else
    return "no";
end if
```

Analysis

- If SAT can be solved in polynomial time, so can FSAT.
  - 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.
- Hence SAT and FSAT are equally hard (or easy).
- Note that this reduction from FSAT to SAT is not a Karp reduction (recall p. 217).
- Instead, it calls SAT multiple times as a subroutine.

---

[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 (D) asks if there is a tour with a total distance at most B.
- TSP asks for a tour with the shortest total distance.
  - The shortest total distance is at most $\sum_{i,j} d_{ij}$.
    - Recall that the input string contains $d_{11}, \ldots, d_{nn}$.
    - Thus the shortest total distance is less than $2^{|\,x\,|}$ in magnitude, where x is the input (why?).
- 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).

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 34 (p. 273).
- 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. 426.
- The reduction of Cook's Theorem L to SAT is a Levin reduction (p. 277).
- So there is a polynomial-time computable function R such that $x \in L$ iff $R(x) \in$ SAT.
- In fact, the proof gives an efficient algorithm to transform a satisfying assignment of $R(x)$ to a certificate for x, too.
- 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.

What If NP $=$ coNP?[a]

- Can you say similar things?

---

[a] Contributed by Mr. Ren-Shuo Liu (`D98922016`) on October 27, 2009.

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).

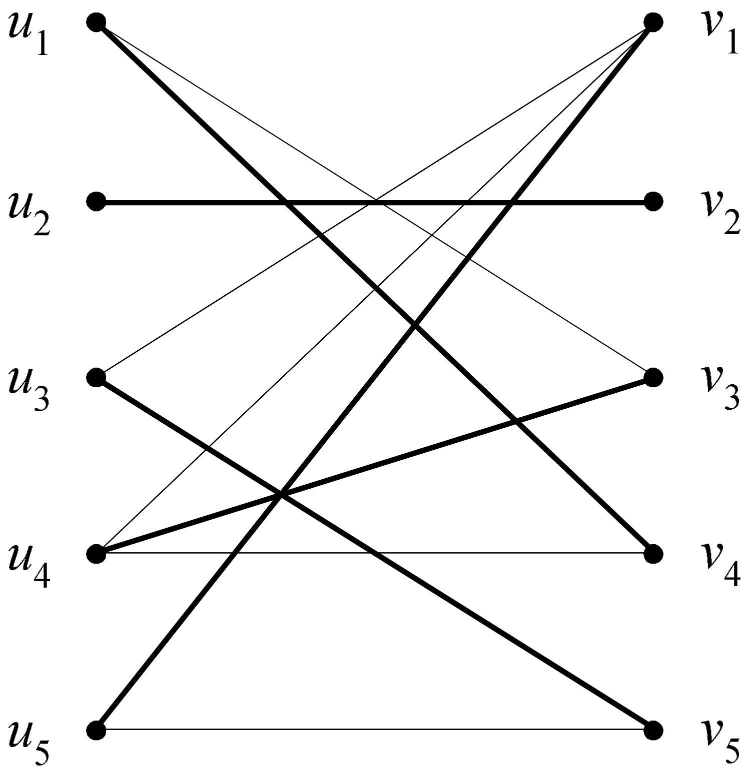
Bipartite Perfect Matching

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

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

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

  - π ranges over all permutations of n elements.
  - $\mathrm{sgn}(\pi)$ is 1 if π is the product of an even number of transpositions and -1 otherwise.
  - Equivalently, $\mathrm{sgn}(\pi) = 1$ if the number of (i, j)s such that $i < j$ and $\pi(i) > \pi(j)$ is even.[a]

---

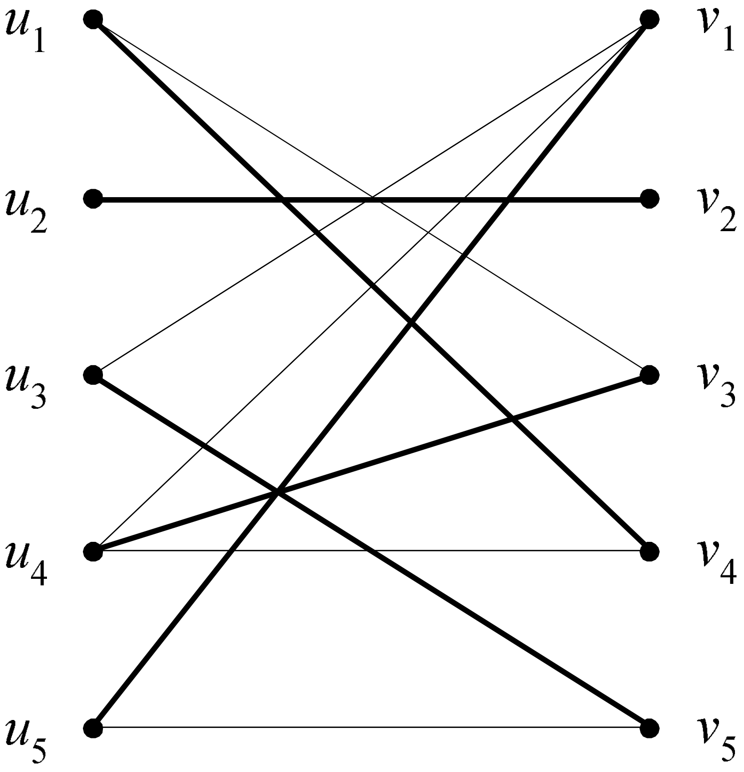
[a] Contributed by Mr. Hwan-Jeu Yu (`D95922028`) on May 1, 2008.

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 $i_1, \ldots, i_m$ 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
```

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. 447.
  - Run the algorithm *independently* k times and output "G has no perfect matchings" if and only if they all say no.
  - The error probability is now reduced to at most 2^{-k}.

Analysis (concluded)[a]

- Note that we are calculating

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

$$\text{prob}[\,\text{algorithm answers “yes”} \mid G \text{ has a perfect matching}\,].$$

- We are *not* calculating

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

$$\text{prob}[\,G \text{ has a perfect matching} \mid \text{algorithm answers “yes”}\,].$$

---

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

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.

Lószló Lovász (1948–)

An Intriguing Question[a]

- Is there an $(i_{11}, \ldots, i_{nn})$ that will always give correct answers for all bipartite graphs of $2n$ nodes?
- A theorem on p. 543 shows that such a witness exists!
- Whether it can be found efficiently is another question.

---

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