The Traveling Salesman Problem

- We are given n cities $1, 2, \ldots, n$ and integer distances d_{ij} between any two cities i and j.
- Assume $d_{ij} = d_{ji}$ for convenience.
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.
- The decision version TSP (D) asks if there is a tour with a total distance at most B, where B is an input.
- Both problems are extremely important but equally hard (p. 325 and p. 392).

A Nondeterministic Algorithm for TSP (D)

for $i = 1, 2, \ldots, n$ **do**
    Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}
end for
$x_{n+1} := x_1$;
{Verification stage:}
if $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n} d_{x_i, x_{i+1}} \leq B$ **then**
    "yes";
else
    "no";
end if

Time Complexity under Nondeterminism

- Nondeterministic machine N decides L **in time** $f(n)$, where $f : \mathbb{N} \to \mathbb{N}$, if
  - N decides L, and
  - for any $x \in \Sigma^*$, N does not have a computation path longer than $f(|x|)$.
- We charge only the "depth" of the computation tree.

Time Complexity Classes under Nondeterminism

- NTIME$(f(n))$ is the set of languages decided by NTMs within time $f(n)$.
- NTIME$(f(n))$ is a complexity class.

NP

- Define
$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k).$$
- Clearly $\text{P} \subseteq \text{NP}$.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
  - Hamiltonian path.
  - Graph colorability.
- The most important open problem in computer science is whether $\text{P} = \text{NP}$.

Simulating Nondeterministic TMs

Theorem 5 *Suppose language L is decided by an NTM N in time $f(n)$. Then it is decided by a 3-string deterministic TM M in time $O(c^{f(n)})$, where $c > 1$ is some constant depending on N.*

- On input x, M goes down every computation path of N using *depth-first* search (but M does *not* know $f(n)$).
  - As M is time-bounded, the depth-first search will not run indefinitely.

The Proof (concluded)

- If some path leads to "yes," then M enters the "yes" state.
- If none of the paths leads to "yes," then M enters the "no" state.

Corollary 6 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

NTIME vs. TIME

- Does converting an NTM into a TM require exploring all the computation paths of the NTM as done in Theorem 5?
- This is the most important question in theory with practical implications.

Nondeterministic Space Complexity Classes

- Let L be a language.
- Then
$$L \in \text{NSPACE}(f(n))$$
if there is an NTM with input and output that decides L and operates within space bound $f(n)$.
- NSPACE$(f(n))$ is a set of languages.
- As in the linear speedup theorem (Theorem 4 on p. 71), constant coefficients do not matter.

Graph Reachability

- Let $G(V, E)$ be a directed graph (digraph).
- REACHABILITY asks if, given nodes a and b, does G contain a path from a to b?
- Can be easily solved in polynomial time by breadth-first search.
- How about the nondeterministic space complexity?

The First Try in NSPACE$(n \log n)$

$x_1 := a$; {Assume $a \neq b$.}
for $i = 2, 3, \ldots, n$ **do**
    Guess $x_i \in \{v_1, v_2, \ldots, v_n\}$; {The ith node.}
end for
for $i = 2, 3, \ldots, n$ **do**
    if $(x_{i-1}, x_i) \notin E$ **then**
        "no";
    end if
    if $x_i = b$ **then**
        "yes";
    end if
end for
"no";

In Fact REACHABILITY $\in$ NSPACE$(\log n)$

$x := a$;
for $i = 2, 3, \ldots, n$ **do**
    Guess $y \in \{2, 3, \ldots, n\}$; {The next node.}
    if $(x, y) \notin E$ **then**
        "no";
    end if
    if $y = b$ **then**
        "yes";
    end if
    $x := y$;
end for
"no";

Space Analysis

- Variables i, x, and y each require $O(\log n)$ bits.
- Testing $(x, y) \in E$ is accomplished by consulting the input string with counters of $O(\log n)$ bits long.
- Hence
$$\text{REACHABILITY} \in \text{NSPACE}(\log n).$$
  - REACHABILITY with more than one terminal node also has the same complexity.
- REACHABILITY $\in$ P (p. 185).

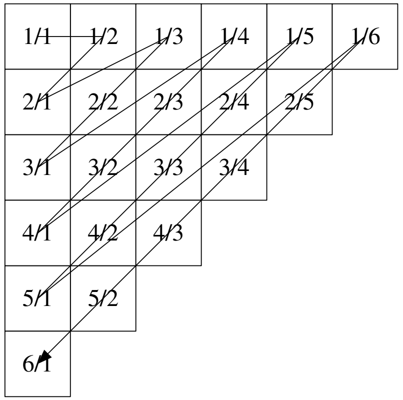
Undecidability

It seemed unworthy of a grown man
to spend his time on such trivialities,
but what was I to do?
— Bertrand Russell (1872–1970),
Autobiography, Vol. I

Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-one correspondence with $\mathbb{N}$, the set of natural numbers.
  - Set of integers $\mathbb{Z}$.
    - $0 \leftrightarrow 0, 1 \leftrightarrow 1, 2 \leftrightarrow 3, 3 \leftrightarrow 5, \ldots, -1 \leftrightarrow 2, -2 \leftrightarrow 4, -3 \leftrightarrow 6, \ldots$.
  - Set of positive integers $\mathbb{Z}^+$: $i - 1 \leftrightarrow i$.
  - Set of odd integers: $(i - 1)/2 \leftrightarrow i$.
  - Set of rational numbers: See next page.
  - Set of squared integers: $i \leftrightarrow \sqrt{i}$.

Rational Numbers Are Countable



Cardinality

- For any set A, define $|A|$ as A's **cardinality** (size).
- Two sets are said to have the same cardinality, written as
$$|A| = |B| \quad \text{or} \quad A \sim B,$$
if there exists a one-to-one correspondence between their elements.
- 2^A denotes set A's **power set**, that is $\{B : B \subseteq A\}$.
  - If $|A| = k$, then $|2^A| = 2^k$.
  - So $|A| < |2^A|$ when A is finite.

Cardinality (concluded)

- $|A| \leq |B|$ if there is a one-to-one correspondence between A and one of B's subsets.
- $|A| < |B|$ if $|A| \leq |B|$ but $|A| \neq |B|$.
- If $A \subseteq B$, then $|A| \leq |B|$.
- But if $A \subsetneq B$, then $|A| < |B|$?

Cardinality and Infinite Sets

- If A and B are infinite sets, it is possible that $A \subsetneq B$ yet $|A| = |B|$.
  - The set of integers *properly* contains the set of odd integers.
  - But the set of integers has the same cardinality as the set of odd integers (p. 102).
- A lot of "paradoxes."

Hilbert's[a] Paradox of the Grand Hotel

- For a hotel with a finite number of rooms with all the rooms occupied, a new guest will be turned away.
- Now let us imagine a hotel with an infinite number of rooms, and all the rooms are occupied.
- A new guest comes and asks for a room.
- "But of course!" exclaims the proprietor, and he moves the person previously occupying Room 1 into Room 2, the person from Room 2 into Room 3, and so on . . ..
- The new customer occupies Room 1.

[a]David Hilbert (1862–1943).

Hilbert's Paradox of the Grand Hotel (concluded)

- Let us imagine now a hotel with an infinite number of rooms, all taken up, and an infinite number of new guests who come in and ask for rooms.
- "Certainly, gentlemen," says the proprietor, "just wait a minute."
- He moves the occupant of Room 1 into Room 2, the occupant of Room 2 into Room 4, and so on.
- Now all odd-numbered rooms become free and the infinity of new guests can be accommodated in them.
- "There are many rooms in my Father's house, and I am going to prepare a place for you." (*John* 14:3)

Galileo's[a] Paradox (1638)

- The squares of the positive integers can be placed in one-to-one correspondence with all the positive integers.
- This is contrary to the axiom of Euclid[b] that the whole is greater than any of its proper parts.
- Resolution of paradoxes: Pick the notion that results in "better" mathematics.
- The difference between a mathematical paradox and a contradiction is often a matter of opinion.

[a]Galileo (1564–1642).
[b]Euclid (325 B.C.–265 B.C.).

Cantor's[a] Theorem

Theorem 7 *The set of all subsets of* $\mathbb{N}$ ($2^{\mathbb{N}}$) *is infinite and not countable.*

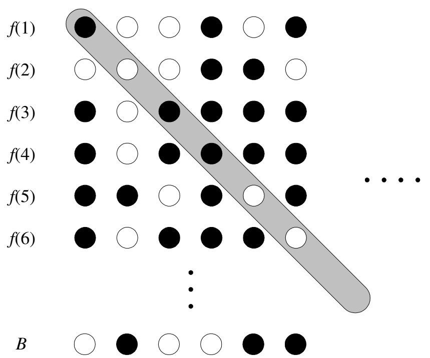
- Suppose it is countable with $f : \mathbb{N} \to 2^{\mathbb{N}}$ being a bijection.
- Consider the set $B = \{k \in \mathbb{N} : k \notin f(k)\} \subseteq \mathbb{N}$.
- Suppose $B = f(n)$ for some $n \in \mathbb{N}$.

[a]Georg Cantor (1845–1918). According to Kac and Ulam, "[If] one had to name a single person whose work has had the most decisive influence on the present spirit of mathematics, it would almost surely be Georg Cantor."

The Proof (concluded)

- If $n \in f(n)$, then $n \in B$, but then $n \notin B$ by B's definition.
- If $n \notin f(n)$, then $n \notin B$, but then $n \in B$ by B's definition.
- Hence $B \neq f(n)$ for any n.
- f is not a bijection, a contradiction.

Cantor's Diagonalization Argument Illustrated



A Corollary of Cantor's Theorem

Corollary 8 *For any set T, finite or infinite,*

$$|\,T\,| < |\,2^T\,|.$$

- The inequality holds in the finite A case.
- Assume A is infinite now.
- $|T| \leq |2^T|$: Consider $f(x) = \{x\}$.
- The strict inequality uses the same argument as Cantor's theorem.

A Second Corollary of Cantor's Theorem

Corollary 9 *The set of all functions on $\mathbb{N}$ is not countable.*

- Every function $f : \mathbb{N} \to \{0, 1\}$ determines a set
$$\{n : f(n) = 1\} \subseteq \mathbb{N}.$$
- And vice versa.
- So the set of functions from $\mathbb{N}$ to $\{0, 1\}$ has cardinality $|\,2^{\mathbb{N}}\,|$.
- Corollary 8 (p. 113) then implies the claim.

Existence of Uncomputable Problems

- Every program is a finite sequence of 0s and 1s, thus a nonnegative integer.
- Hence every program corresponds to some integer.
- The set of programs is countable.
- A function is a mapping from integers to integers.
- The set of functions is not countable by Corollary 9 (p. 114).
- So there must exist functions for which there are no programs.

Universal Turing Machine[a]

- A **universal Turing machine** U interprets the input as the *description* of a TM M concatenated with the *description* of an input to that machine, x.
  - Both M and x are over the alphabet of U.
- U simulates M on x so that
$$U(M; x) = M(x).$$
- U is like a modern computer, which executes any valid machine code, or a Java Virtual machine, which executes any valid bytecode.

[a]Turing (1936).

The Halting Problem

- **Undecidable problems** are problems that have no algorithms or languages that are not recursive.
- We knew undecidable problems exist (p. 115).
- We now define a concrete undecidable problem, the **halting problem**:
$$H = \{M; x : M(x) \neq \nearrow\}.$$
  - Does M halt on input x?

H Is Recursively Enumerable

- Use the universal TM U to simulate M on x.
- When M is about to halt, U enters a "yes" state.
- If $M(x)$ diverges, so does U.
- This TM accepts H.
- Membership of x in any recursively enumerative language accepted by M can be answered by asking
$$M; x \in H?$$

H Is Not Recursive

- Suppose there is a TM M_H that *decides* H.
- Consider the program $D(M)$ that calls M_H:

  if $M_H(M; M) =$ "yes" **then**
      $\nearrow$; {Writing an infinite loop is easy, right?}
  else
      "yes";
  end if

- Consider $D(D)$:
  - $D(D) = \nearrow \Rightarrow M_H(D; D) =$ "yes" $\Rightarrow D; D \in H \Rightarrow D(D) \neq \nearrow$, a contradiction.
  - $D(D) =$ "yes" $\Rightarrow M_H(D; D) =$ "no" $\Rightarrow D; D \notin H \Rightarrow D(D) = \nearrow$, a contradiction.

Comments

- Two levels of interpretations of M:
  - A sequence of 0s and 1s (data).
  - An encoding of instructions (programs).
- There are no paradoxes.
  - Concepts should be familiar to computer scientists.
  - Supply a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, etc.

Self-Loop Paradoxes

Cantor's Paradox (1899): Let T be the set of all sets.

- Then $2^T \subseteq T$, but we know $|2^T| > |T|$ (p. 113)!

Eubulides: The Cretan says, "All Cretans are liars."

Liar's Paradox: "This sentence is false."

Sharon Stone in *The Specialist* (1994): "I'm not a woman you can trust."

More Undecidability

- $\{M : M$ halts on all inputs$\}$.
  - Given $M; x$, we construct the following machine:
    - $M_x(y)$: if $y = x$ then $M(x)$ else halt.
  - M_x halts on all inputs if and only if M halts on x.
  - So if the said language were recursive, H would be recursive, a contradiction.
  - This technique is called **reduction**.

More Undecidability (concluded)

- $\{M; x : \text{there is a } y \text{ such that } M(x) = y\}$.
- $\{M; x : \text{the computation } M \text{ on input } x \text{ uses all states of } M\}$.
- $\{M; x; y : M(x) = y\}$.

Reductions in Proving Undecidability

- Suppose we are asked to prove L is undecidable.
- Language H is known to be undecidable.
- We try to find a computable transformation (or reduction) R such that

$$R(x) \in L \text{ if and only if } x \in H.$$

- This suffices to prove that L is undecidable.

Complements of Recursive Languages

Lemma 10 *If L is recursive, then so is $\bar{L}$.*

- Let L be decided by M (which is deterministic).
- Swap the "yes" state and the "no" state of M.
- The new machine decides $\bar{L}$.

Recursive and Recursively Enumerable Languages

Lemma 11 *L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.*

- Suppose both L and $\bar{L}$ are recursively enumerable, accepted by M and $\bar{M}$, respectively.
- Simulate M and $\bar{M}$ in an *interleaved* fashion.
- If M accepts, then $x \in L$ and M' halts on state "yes."
- If $\bar{M}$ accepts, then $x \notin L$ and M' halts on state "no."

A Very Useful Corollary and Its Consequences

Corollary 12 *L is recursively enumerable but not recursive, then $\bar{L}$ is not recursively enumerable.*

- Suppose $\bar{L}$ is recursively enumerable.
- Then both L and $\bar{L}$ are recursively enumerable.
- By Lemma 11 (p. 126), L is recursive, a contradiction.

Corollary 13 *$\bar{H}$ is not recursively enumerable.*

R, RE, and coRE

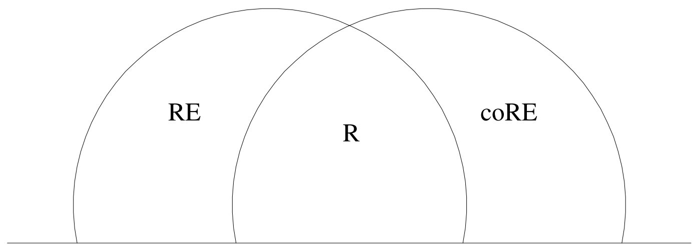
RE: The set of all recursively enumerable languages.

coRE: The set of all languages whose complements are recursively enumerable (note that coRE is not $\overline{\text{RE}}$).

R: The set of all recursive languages.

R, RE, and coRE (concluded)

- R = RE $\cap$ coRE (p. 126).
- There exist languages in RE but not in R and not in coRE.
  - Such as H (p. 118 and p. 119).
- There are languages in coRE but not in RE.
  - Such as $\bar{H}$ (p. 127).
- There are languages in neither RE nor coRE.



Notations

- Suppose M is a TM accepting L.
- Write $L(M) = L$.
  - In particular, if $M(x) = \nearrow$ for all x, then $L(M) = \emptyset$.
- If $M(x)$ is never "yes" nor $\nearrow$ (as required by the definition of acceptance), we let $L(M) = \emptyset$.

Nontrivial Properties of Sets in RE

- A property of a set accepted by a TM (a recursively enumerable set) is **trivial** if it is always true or false.
  - Is a recursively enumerable set accepted by a TM? Always true.
- It can be defined by the set $\mathcal{C}$ of recursively enumerable sets that satisfy it.
- The property is nontrivial if $\mathcal{C} \neq \text{RE}$ and $\mathcal{C} \neq \emptyset$.
- Up to now, all nontrivial properties of recursively enumerable sets are undecidable (pp. 122–123).
- In fact, Rice's theorem confirms that.

Consequences of Rice's Theorem

Corollary 14 *The following properties of recursively enumerative sets are undecidable.*

- *Emptiness.*
- *Finiteness.*
- *Regularity.*
- *Context-freedom.*

Boolean Logic

Boolean Logic[a]

Boolean variables: $x_1, x_2, \ldots$.

Literals: x_i, $\neg x_i$.

Boolean connectives: $\vee, \wedge, \neg$.

Boolean expressions: Boolean variables, $\neg\phi$ (**negation**), $\phi_1 \vee \phi_2$ (**disjunction**), $\phi_1 \wedge \phi_2$ (**conjunction**).

- $\bigvee_{i=1}^{n} \phi_i$ stands for $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$.
- $\bigwedge_{i=1}^{n} \phi_i$ stands for $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$.

Implications: $\phi_1 \Rightarrow \phi_2$ is a shorthand for $\neg\phi_1 \vee \phi_2$.

Biconditionals: $\phi_1 \Leftrightarrow \phi_2$ is a shorthand for $(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$.

[a] Boole (1815–1864) in 1847.

Truth Assignments

- A **truth assignment** T is a mapping from boolean variables to **truth values** `true` and `false`.
- A truth assignment is **appropriate** to boolean expression ϕ if it defines the truth value for every variable in ϕ.
  - $\{x_1 = \texttt{true}, x_2 = \texttt{false}\}$ is appropriate to $x_1 \vee x_2$.

Satisfaction

- $T \models \phi$ means boolean expression ϕ is true under T; in other words, T **satisfies** ϕ.
- ϕ_1 and ϕ_2 are **equivalent**, written

$$\phi_1 \equiv \phi_2,$$

  if for any truth assignment T appropriate to both of them, $T \models \phi_1$ if and only if $T \models \phi_2$.
  - Equivalently, $T \models (\phi_1 \Leftrightarrow \phi_2)$.

Truth Tables

- Suppose ϕ has n boolean variables.
- A **truth table** contains 2^n rows, one for each possible truth assignment of the n variables together with the truth value of ϕ under that truth assignment.
- A truth table can be used to prove if two boolean expressions are equivalent.
  - Check if they give identical truth values under all 2^n truth assignments.

A Truth Table

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

De Morgan's[a] Laws

- **De Morgan's laws** say that

$$\neg(\phi_1 \wedge \phi_2) = \neg\phi_1 \vee \neg\phi_2,$$
$$\neg(\phi_1 \vee \phi_2) = \neg\phi_1 \wedge \neg\phi_2.$$

- Here is a proof for the first law:

ϕ_1	ϕ_2	$\neg(\phi_1 \wedge \phi_2)$	$\neg\phi_1 \vee \neg\phi_2$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

[a] Augustus DeMorgan (1806–1871).

Conjunctive Normal Forms

- A boolean expression ϕ is in **conjunctive normal form** (**CNF**) if
$$\phi = \bigwedge_{i=1}^{n} C_i,$$
where each **clause** C_i is the disjunction of one or more literals.
- For example, $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$ is in CNF.
- Convention: An empty CNF is satisfiable, but a CNF containing an empty clause is not.

Disjunctive Normal Forms

- A boolean expression ϕ is in **disjunctive normal form** (**DNF**) if
$$\phi = \bigvee_{i=1}^{n} D_i,$$
where each **implicant** D_i is the conjunction of one or more literals.
- For example,
$$(x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2) \vee (x_2 \wedge x_3)$$
is in DNF.