$\sum_{i=k}^{n} 1 = n+1-k$ $\qquad \sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$

$\sum_{i=1}^{n} f(x) = n f(x)$ $\qquad \sum_{i=1}^{n} i^2 = \frac{1}{6}n(n+1)(2n+1)$

$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r} (r \neq 1)$ $\qquad \sum_{i=1}^{n} i^3 = \frac{1}{4}n^2(n+1)^2$

$\sum_{i=0}^{n} 2^i = 2^{n+1}-1$ $\qquad \sum_{i=0}^{n} \frac{1}{2^i} = 2 - \frac{1}{2^n}$ $\qquad \sum_{i=1}^{n} \log i = \log n!$

$$\frac{T(n)}{f(n)} \xrightarrow[n\to\infty]{} \begin{cases} \infty & T \in \omega(f) \\ \text{constant} > 0 & T \in \Theta(f) \\ 0 & T \in o(f) \end{cases}$$

$T \in o(f)$	$T \in O(f)$	$T \in \Theta(f)$	$T \in \Omega(f)$	$T \in \omega(f)$
$T < f$	$T \le f$	$T = f$	$T \ge f$	$T > f$

Monotonically increasing

$$\int_{m-1}^{n} dx\, f(x) \le \sum_{i=m}^{n} f(i) \le \int_{m}^{n+1} dx\, f(x)$$

$$\sum_{i=1}^{n} f(i) \in \Theta\left(\int_0^n dx\, f(x)\right)$$

Quotient-Remainder Theorem *

Given $n \in \mathbb{Z}$ and $d \in \mathbb{N}$, there is a unique quotient $q \in \mathbb{Z}$ and remainder r with $0 \le r < d$, such that $n = qd + r$ [$r = rem(n,d)$. if remainder is zero, $n = qd$ for some $q \in \mathbb{Z}$ and d divides n, $d \mid n$]

Bezout's Identity

The GCD of m and n is the smallest positive linear combination of m, n with integer coefficients. For some $x, y \in \mathbb{Z}$, $\gcd(m,n) = mx + ny$.

GCD Facts: $X < U$ $\sqrt{}$ $\cap$ 4

i) $\gcd(m,n) = \gcd(m, rem(n,m))$ $\qquad \gcd(km, kn) = k \cdot \gcd(m,n)$

ii) Every common divisor m, n divides $\gcd(m,n)$

iii) For $k \in \mathbb{N}$, ~~$\gcd(l,n)=1$, $\gcd(l,m)=1$ $\gcd(l,mn)=1$~~

iv) If $\gcd(l,m)=1$ and $\gcd(l,n)=1$ $\gcd(l,mn)=1$

v) If $d \mid mn$ and $\gcd(d,m)=1$, then $d \mid n$.

Theorem

Every integer $n \ge 2$ can be written uniquely as a product of primes.

Euclid's Lemma Generalized

Let P is prime. If $p \mid mn$ then $p \mid n$ or $p \mid m$.

Modular Equivalence Properties $\quad a \equiv b \pmod d$ $\quad r \equiv s \pmod d$

a) $ar \equiv bs \pmod d$ $\qquad$ b) $a + r \equiv b + s \pmod d$

c) $a^n \equiv b^n \pmod d$

Theorem

Suppose $ac \equiv bc \pmod d$ and $\gcd(c,d) = 1 \Rightarrow a \equiv b \pmod d$

Graph

Sum of degrees = 2 × number of edges

$$\boxed{Q(n,k) = \binom{n+k-1}{k-1} \quad \begin{matrix} k-1 \text{ delimiter} \\ k \text{ color / types} \\ n \text{ length} \end{matrix}}$$ ☆

同于 $x_1 + x_2 + x_3 + x_4 + \cdots$

Order matters no replacement *

$n \times (n-1) \times (n-2) \times \cdots (n-(k-1)) = \frac{n!}{(n-k)!}$

number of k-subsets $= \binom{n}{k} = \frac{n!}{(n-k)!k!}$ order doesn't matter

$\binom{n}{k} \Rightarrow x^k y^{n-k}$ Binomial coefficients

Structural Induction on recursive set

[Base case] $P(s_1), P(s_2), \ldots P(s_k)$ are T

[Induction step] For every constructor, If P is T for the parent elements, then P is T for new child created.

iv) $\gcd(l,m)=1$ $\quad \gcd(l,n)=1$

$1 = lx + my$ $\quad 1 = lx' + ny'$

$1 = (lx+my)(lx'+ny') = l(lxx' + nxy' + myx') + mn \cdot (yy')$ a positive linear combination

iii) $\gcd(km, kn) = kmx + kny = k(mx+ny)$

$k > 0$, $mx + ny$ must be minimum positive linear combination of m, n, which means that $mx + ny = \gcd(m,n)$.

Recursive Rooted Binary tree / full Binary tree

① The empty ε is RBT / A sigle root-node •

② T_1, T_2 are disjoint RBTs with roots r_1 and r_2, then linking r_1 and r_2 to a new root r gives a new RBT with root r.

$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

$(p \vee q) \vee r \equiv p \vee (q \vee r)$

$\neg(p \wedge q) \equiv \neg p \vee \neg q$

$\neg(p \vee q) \equiv \neg p \wedge \neg q$

$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

$p \to q \equiv \neg q \to \neg p$

$p \to q \equiv \neg p \vee q$

$\forall a \neq (\exists d : P(a,d)) < \exists d : (\forall a : P(a,d))$

$\overline{A \cup B} = \overline{A} \cap \overline{B}$

$\overline{A \cap B} = \overline{A} \cup \overline{B}$

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Perbutation order ✓

replacement $n \times n \times n \cdots$

no relacement $n \times (n-1) \times \cdots$

combination order ✗

$C^R(n,r) = \frac{(n+r-1)!}{r!(n-1)!}$ $\binom{n}{r}$

$\binom{n}{k}$	0	1	2	3
0	1			
1	1	1		
2	1	2	1	1
3	1	3	3	4
4	1	4	6	10

A ARDVARK

$\binom{8}{3}\binom{5}{2}\binom{3}{1}\binom{2}{1}\binom{1}{1}$

A R

$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$

$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|$

$(x+y+z)^n = \sum_{0 \le i+j \le n} \frac{n!}{i!j!(n-i-j)!} x^i y^j z^{n-i-j}$

(a) (i) The Turing machine copies the bits over one by one.

1: Move right to the first ␣ and write #.
2: Return to ∗.
3: Move right to first non-marked before #.
   Remember and mark the bit.
   If, instead, you reach #, return to ∗ unmarking all the ✓ and halt.
4: Move right to first ␣, write the remembered bit and GOTO step 2.

(ii) $\mathcal{L} = \{w\#w \mid w \in \Sigma^*\}$.

(b) (i) We use a ✗ to simulate the punctuation #.

1: Move right to the first ␣ and mark with ✗.
2: Return to ∗.
3: Move right to first non-marked before ✗.
   Remember and mark the bit with ✓.
   If you reach ✗, unmark the bit, return to ∗ unmarking all the ✓ and halt.
4: Move right to first ␣, write the remembered bit and GOTO step 2.

(ii) $\mathcal{L} = \{ww \mid w \in \Sigma^*\}$.

(c) (i) Write a 1 for every zero and repeat for every zero.

1: Move right to the first ␣ and mark with #.
2: Return to ∗.
3: Move right to first non ✗-marked 0 and mark with ✗.
   If you reach #, return to ∗ unmarking all 0's and halt.
4: Return to ∗.
5: Move right to first non ✓-marked 0 and mark with ✓
   If you reach #, return to ∗ unmarking ✓s (leaving the ✗s) and GOTO step 3.
6: Move right to first ␣ and write 0.
7: Move left to first ✓ and GOTO step 5.

(ii) $\mathcal{L} = \{0^{\bullet n}\#1^{\bullet n^2} \mid n \ge 0\}$.

(d) (i) Mark and replace the first with the last bit and *vice versa* and continue.

1: Move right to the first non-marked bit. Mark it and remember it.
   If you reach ␣, return to ∗, erasing all marks and halt.
2: Move right to the last non-marked bit.
   If there is none, return to ∗, erasing all marks and halt.
   Otherwise, remember it, replace it with the bit from step 1 and mark it.
3: Move left to the first marked bit.
   Replace the bit with the bit remembered in step 2 and GOTO step 1.

(ii) $\mathcal{L} = \{w\#w^R \mid w \in \Sigma^*\}$.

Theorem 27.3. IF $\mathcal{L}_{HALT}$ is Turing-decidable, THEN $\mathcal{L}_{TM}$ is Turing-decidable. That is, $\mathcal{L}_{TM} \le_R \mathcal{L}_{HALT}$.

Theorem 27.3 says $\mathcal{L}_{TM}$ is reducible to $\mathcal{L}_{HALT}$, that is $\mathcal{L}_{TM}$ is *easier* than $\mathcal{L}_{HALT}$. Theorem 27.3 asserts an implication, which doesn't prove anything useful until we add more information (recall our discussion of logical implication in Chapter 4 on page 43). The additional known information is that $\mathcal{L}_{TM}$ is undecidable.

Theorem 27.4. $\mathcal{L}_{HALT}$ is undecidable.

Proof. (Contradiction) Assume $\mathcal{L}_{HALT}$ is decidable. By Theorem 27.3, $\mathcal{L}_{TM}$ is decidable, a contradiction. ∎

Let's now prove Theorem 27.3 which establishes that $\mathcal{L}_{HALT}$ is "harder" than $\mathcal{L}_{TM}$.

Proof. (of Theorem 27.3) We use a direct proof of the implication. Assume that $\mathcal{L}_{HALT}$ is decidable, and let H_{TM} be a decider for $\mathcal{L}_{HALT}$. We use H_{TM} to construct A_{TM}, a decider for $\mathcal{L}_{TM}$. Here is the idea. Run H_{TM} to determine if M halts. If M does not halt, reject. If M does halt, then simulate M on w to determine accept or reject (by running a universal Turing Machine U_{TM} on $\langle M \rangle\#w$). Here is the sketch of A_{TM}.

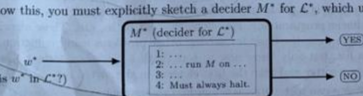
A_{TM} = Turing Machine derived from H_{TM} (the decider for $\mathcal{L}_{HALT}$)
INPUT: $\langle M \rangle\#w$ where M is a Turing Machine and w an input to M.
1: Run H_{TM} on input $\langle M \rangle\#w$. If H_{TM} rejects, then REJECT and halt.
2: Run U_{TM} on input $\langle M \rangle\#w$ and output the decision U_{TM} gives.

In step 1, the decider H_{TM} must halt. Step 2 only runs if H_{TM} accepts $\langle M \rangle\#w$, which means M halts on w, so U_{TM}, when it simulates M on w, must halt. Thus, A_{TM} always halts. Further, A_{TM} accepts $\langle M \rangle\#w$ if and only if U_{TM} accepts $\langle M \rangle\#w$ in step 2, which happens if and only if M accepts w. Therefore, A_{TM} decides $\mathcal{L}_{TM}$. ∎

We summarize the steps in our proof that $\mathcal{L}_{HALT}$ is unsolvable into a general method for proving undecidability.

To prove that a problem $\mathcal{L}$ is undecidable:

1: Find an undecidable problem $\mathcal{L}^*$ which you believe is easier than $\mathcal{L}$. Usually $\mathcal{L}^* = \mathcal{L}_{TM}$ or $\mathcal{L}_{HALT}$.
2: Show that $\mathcal{L}^* \le_R \mathcal{L}$, that is, $\mathcal{L}$ is indeed harder than an undecidable problem. You must show:
   IF there is a decider M for $\mathcal{L}$, THEN there is a decider M^* for $\mathcal{L}^*$.
   To show this, you must explicitly sketch a decider M^* for $\mathcal{L}^*$, which uses M as a subroutine.



Event of interest: subsets of outcomes where you win

Sample space: $\Omega = \{w_1, w_2, \cdots w_n\}$ is the set of possible outcomes

Uniform probality space: every outcome in the sample space has the same probality

$P(A|B) = \frac{P(A \cap B)}{P(B)}$

$P(A_1 \cap A_2 \cap A_3) = P(A_1 | A_2 \cap A_3) \times P(A_2 | A_3) \times P(A_3)$

$P(A) = P(A|B) \cdot P(B) + P(A|\bar{B}) \cdot P(\bar{B})$

$\left.\begin{array}{l} P(A \cap B) = P(A) \times P(B) \\ P(A|B) = P(A) \end{array}\right\}$ Independent

Focs twins:

$P(E_K | E_1 \cap E_2 \cap E_3 \cdots \cap E_{K-1}] = \left(\frac{B-K}{B-K+1}\right)^{N-K}$

$P(K\ L; p) = \begin{cases} \frac{\beta^L - \beta^K}{\beta^L - 1} & \beta = p/(1-p) \quad p \neq \frac{1}{2} \\ \frac{L-K}{L} & p = \frac{1}{2} \end{cases}$

P: closer to the goal <probality>

L-K: L-K steps away in opposite direction

K: steps to success

$B(K; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$

$P_X(t) = \beta(1-p)^t \quad t = 1, 2, 3\ldots \quad \beta = \frac{p}{1-p}$

$E(X)_B = np \quad E(X)_{wait} = n/p$

$E(X) = E(X|A) \cdot P(A) + E(X|\bar{A}) \cdot P(\bar{A})$

$E(X^2) = E[(1+Y)^2] = E[1 + 2Y + Y^2]$
$= 1 + 2E(Y) + E[Y^2]$

$W(K, \ell) = E($waiting time | boy $\times P($boy$)$
$+ E($waiting time | girl$) \times P($girl$)$
$= \{1 + W(K-1, \ell)\} \times P + \{1 + W(K, \ell-1)\} \times (1-P)$

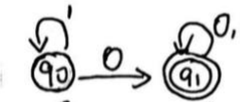
number of ways to assign k hats correct

$\frac{\binom{n}{k} \times (n-k)! \sum_{i=0}^{n-k} \frac{(-1)^i}{i!}}{n!} = \frac{1}{k!} \sum_{i=0}^{n-k} \frac{(-1)^i}{i!}$

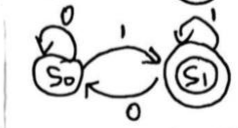
$|A| \leq |B|$ injection

$|A| = |B|$ Bijection

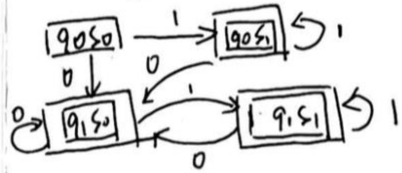
$|A| \leq |N|$ countable (injection)



string with a 0



String ending in 1.



$*1*1*1* = \{$string with at most two 1s$\}$

$\{0\}^* \cdot \{\varepsilon, 1\} \cdot \{0\}^* \cdot \{\varepsilon, 1\} \cdot \{0\}^*$

CFG:

repetition: $\{w\#w\}$ multiple-equality: $\{0^{\cdot n} 1^{\cdot n} 0^{\cdot n}\}$

squaring: $\{0^{\cdot n^2}\}$ $\{0^{\cdot n} 1^{\cdot n^2}\}$ exponentiation: $\{0^{\cdot 2^n}\}$ $\{0^{\cdot n} 1^{\cdot 2^n}\}$

$w \in \mathcal{L}(M) \leftrightarrow M(w) =$ halt and $\boxed{\text{Yes}}$

$w \notin \mathcal{L}(M) \leftrightarrow M(w) =$ halt and No or forever

---

$w \in \mathcal{L}(M) \leftrightarrow M(w) =$ halt and $\boxed{\text{Yes}}$

$w \notin \mathcal{L}(M) \leftrightarrow M(w) =$ halt and $\boxed{\text{No}}$

$|a| < 1 \quad \sum_{k=1}^{\infty} k a^k = \frac{a}{(1-a)^2}$

$P(\text{Home} | RL) = P(\text{Home})$

$P(\text{home}) = P(L) \cdot P(\text{home}|L) + P(RR) \cdot P(\text{home}|RR) + P(RL) \cdot P(\text{home}|RL)$

$E(X) = E(X|HH) \cdot P(HH) + E(X|HT) P(HT) + E(X|T) \cdot P(T)$

Hall's Theorem

Left X: Right N(X) if $|X| \leq |N(X)|\ \forall$, then it can be done

---

A reduced to B → A easier than B

A's decider convertible to B's decider

→ A harder than B

- A computing problem is decision problem -> Yes set (a set of finite binary strings)
- Every finite language is regular.
- The description of a Turing Machine is a finite a finite binary string
- Not all binary strings are Turing machine, but any Turing machine has a unique binary description
- Problems solved by Turing machine is countable
- There are uncountably computing problems
- UTM is **undecidable** … recognizer & a decider for language of UTM is a program that looks at another program and determines ahead of time whether terminated successfully.
- General languages are uncountable and so the non-recognizable languages must be uncountable.
- Every decidable language is recognizable.

Exercise 22.2. First we show f is 1-to-1. Suppose not. Let $n_1 \neq n_2$ and $f(n_1) = f(n_2)$. So,

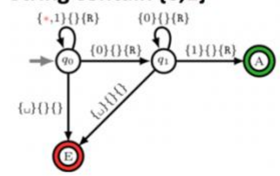
$$\tfrac{1}{4}(1 + (-1)^{n_1}(2n_1 - 1)) = \tfrac{1}{4}(1 + (-1)^{n_2}(2n_2 - 1)) \rightarrow (-1)^{n_1}(2n_1 - 1) = (-1)^{n_2}(2n_2 - 1).$$

The sign of both sides must be the same, so $(-1)^{n_1} = (-1)^{n_2}$ and we conclude $2n_1 - 1 = 2n_2 - 1$. That is, $n_1 = n_2$, a contradiction. So, f is an injection. Now we show that f is onto. Given $z \in \mathbb{Z}$, we must find n for which $f(n) = z$.

$$z > 0: n = 2z \quad \rightarrow f(n) = \tfrac{1}{4}(1 + (-1)^{2z}(4z - 1)) = z;$$
$$z \leq 0: n = 2|z| + 1 \rightarrow f(n) = \tfrac{1}{4}(1 + (-1)^{2|z|+1}(4|z| + 1)) = -|z| = z.$$

Therefore, f is onto, and hence a bijection from $\mathbb{N}$ to $\mathbb{Z}$.

String contain {0,1}



Repetition without punctuation

1: If the first symbol is ␣, ACCEPT (empty input).
2: Return to ∗.
// Mark the first half with ✓ and the second with ✗
3: Move right to the *first* unmarked bit and mark it ✓.
If none exists (you come to ✗), GOTO Step 5.
4: Move right to the *last* unmarked bit and mark it ✗.
If none exists (the first right symbol is ␣ or ✗) REJECT.
(the input has an odd number of bits)
Otherwise, after marking, GOTO Step 2.

After the loop involving steps 3 and 4, the input string is partitioned into two halves: the first is marked with ✓ and the second with ✗. We now compare ✓ bits with ✗ bits.

5: Return to ∗
// Match each ✓-bit with a corresponding ✗-bit
6: Move right to the first bit marked ✓.
If none exists (you come to ␣) ACCEPT
Otherwise remember the bit and unmark it.
7: Move right to the first bit marked ✗.
If the bit does not match the bit remembered, REJECT.
If it is a match, unmark the bit and GOTO Step 5.

Best → Worst Runtime

$\log n$ log
n linear
$n \log n$ loglinear
n^2 quadratic
n^3 cubic
$n^{\log n}$ super polynomial
2^n exponential
$n!$ factorial
n^n BAD