★ Randomized Algorithms:

① Application by Verifying Polynomial Identities.

$$f(x) = \prod_{i=1}^{n} (x - a_i)$$

$$g(x) = \sum_{i=0}^{n} c_i x^i$$

$$f(x) \stackrel{?}{\equiv} g(x)$$

→ Algo chooses integer r uniformly at random in range $\{1, \ldots, 100n\} : S$

∴ $f(r)$ in $O(n)$ (Multiply n numbers)

$g(r)$ in $O(n)$ (Horner's Rule)

If $f(x) \equiv g(x)$ & $f(r) = g(r)$ } Correct ans

$f(x) \not\equiv g(x)$ & $f(r) \neq g(r)$ }

If $f(x) \not\equiv g(x)$ & $f(r) = g(r)$ Incorrect ans

⇒ $f(x) - g(x) = 0$ has a root r

where deg is atmost n

∴ ∃ atmost n roots in S

∴ Error $\leq 1/100$

Run Algo for K times. Error $\leq \left(\frac{1}{100}\right)^k$

→ Monte Carlo Algo.

② Verifying Matrix Multiplication (All operations done modulo 2)

→ $A_{n\times n}$ $B_{n\times n}$ $C_{n\times n}$ (all entries are either 0 or 1)

$\bar{r} = (r_1, r_2, \ldots, r_n) \in \{0,1\}^n$

∴ Prob to pick 1 random vector $1/2^n$

$(A(Br)) \overset{?}{=} Cr \quad \rightarrow \Theta(n^2)$

$X \quad Y$

If $AB = C$ and $ABr = Cr$ } Correct

$AB \neq C$ and $ABr \neq Cr$ }

If $AB \neq C$ and $ABr = Cr$ then ans is wrong.

∴ $D = AB - C \neq 0$

∴ ∃ atleast one component of D not equal to zero

WLOG assume $d_{11} \neq 0$

$X = Y$

$\Rightarrow (AB)r = Cr$

$\Rightarrow (AB - C)r = 0$

$\Rightarrow Dr = 0$

$\therefore d_{11}r_1 + d_{12}r_2 + \ldots + d_{1n}r_n = 0$

$\Rightarrow r_1 = -\dfrac{(d_{12}r_2 + \ldots + d_{1n}r_n)}{d_{11}}$

* $(r_2, \ldots, r_n)$ is determined already then we have only one choice of r_1 for which the equality holds out of the 2 choices of r_1 i.e. 0 & 1.

$\therefore$ error $\leq 1/2$

after k runs error $\leq (1/2)^k$: Monte Carlo

→ Fingerprinting

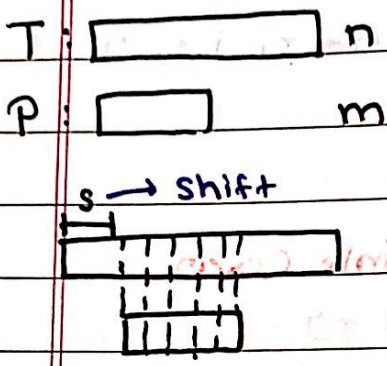
Try only at one sample point rather than at every point.

* Abundance of Witness
* Above Idea (Principle of deffered decisions)

If ∃ several random variables such as r_i of F some are set at one point in algo & other left random or deffered until some further point in analysis.

* LAS Vegas: Randomized algo that always returns a correct result. But running time may vary between executions.
* Monte Carlo: A randomized algo that terminates in polynomial time, but might produce erroneous result.

③ Pattern Matching (Only Binary strings considered)



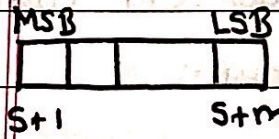
: First occurrance of P in T

★ Worst case T.C. : $O((n-m+1)m)$

→ Valid shift

Invalid Shift

★ fingerprint function



$$f(T[s+1 \ldots s+m]) = T[s+m] \times 2^0 + T[s+m-1] \times 2^1 + \vdots + T[s+1] \times 2^{m-1}$$

★ Horner's Rule:

$T[s+m] + 2(\ldots 2(T[s+2] + 2T[s+1]))$

∴ it takes $O(m)$ time to compute the fingerprint

∴ $f(T[s+1 \ldots s+m]) = t_s$

$f(P[1 \ldots m]) = p$

$f(T[s+2 \ldots s+m+1]) = t_{s+1}$

$\therefore t_{s+1} = 2(t_s - 2^{m-1} T[s+1]) + T[s+m+1]$

∴ Given t_s we can compute t_{s+1} in O(1) time

* Precompute powers of 2 till m

→ s is a valid shift ⇔ $t_s = p$

Hence to compute:

p : O(m)

$t_0, t_1 \ldots t_{n-m}$: O(n-m+1)

∴ Compute p : O(m) & t_0 : O(m)

& $t_1 \ldots t_{n-m}$: O(n-m)

∴ T.C. O(2m + (n-m)) = O(n+m)

* ∵ t_s & p can be ↑ calculate them w.r.t modulo q ∈ Prime

* Whenever string match we compare the substring with P

Fingerprints

∴ Worst case T.C. = O(2m + (n-m) + (n-m+1)m)

≈ O(nm)

Max possible no of checks

* Avg case:

f(P) = k where $k \in [0, 1, 2, \ldots, q-1]$

* Supurious hits:

Finger prints match but strings donot.

∴ Probability for ts to match with p modulo q is $1/q$

∴ Avg case: $O(2m + n - m + (\frac{n-m+1}{q})m)$

if $q > m$: $O(n+m)$

$Pr(f(T[s+1..s+m]) = f(P) \mid T[s+1..s+m] \neq P)$

$= Pr(ts - p \text{ is divisible by } q \mid T[s+1..s+m] \neq P)$

$T[s+1] \ldots T[s+m]$: max value $2^m - 1$ (# CP)

∴ $ts - p < 2^m$

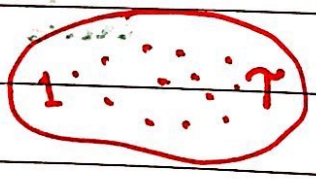
⇒ $ts - p$ has almost m prime divisor

If > m prime divisors:

$ts - p > 2^{>m} > 2^m$ (#)

* Choose q uniformly from $[1, T]$

∴ no. of prime nos $\frac{T}{\ln T}$



Only almost m of these can divide $ts - p$

∴ $Pr(ts \equiv p \pmod q \mid T[s+1..s+m] \neq P) = Pr(Z)$

$$\therefore Pr(Z) \leq \frac{m}{T/\ln T} = \frac{m \ln T}{T}$$

$$\Rightarrow T = n^2 m \lg(n^2 m)$$

$\ln T \approx \lg T$ (# only a const factor)

$$\therefore Pr(Z) \leq \frac{m \lg(n^2 m \lg(n^2 m))}{n^2 m \lg(n^2 m)}$$

$$\leq \frac{\lg(\alpha \lg(\alpha))}{n^2 \lg(\alpha)} \quad \text{where } \alpha = n^2 m$$

$$\lim_{\alpha \to \infty} \frac{\lg(\alpha \lg(\alpha))}{\lg(\alpha)} = \frac{\frac{1}{\alpha \lg(\alpha)}(1 + \lg \alpha)}{1/\alpha} = 1 + \frac{1}{\lg(\alpha)} \approx 1$$

$$\therefore Pr(Z) \leq 1/n^2$$

∴ Monte Carlo : $O(n+m)$

Error $\leq 1/n^2$

→ LAS VEGAS:

Whenever two fingerprint match explict check by naive

$O(2m + (n-m))\left(1 - 1/n^2\right) + O(nm)\left(\frac{1}{n^2}\right)$: Expected time

$= O\left(n+m\left(\frac{n^2-1}{n^2}\right) + \frac{m}{n}\right)$

$\approx O(n + m + m/n)$

✱	TC
Naive	$O((n-m+1)m)$
W.C.	$O(nm)$
Avg Case	$O(n+m)$
Monte Carlo	$O(n+m)$ Error $\leq 1/m^2$
Las Vegas	$O(n+m+m/n)$: Expected time

→ Fingerprinting Types:

① Matrix Mul. } fix field then choose point of eval for fingerprint.
Poly Check }

② Pattern matching } After fixing point of evaluation we fix 1 and 7 (i.e. field)

Determine whether a odd number is prime or not.

Naive: $O(\sqrt{n}\ div(n)) = O(2^{\lg n/2}\ div(n))$

↓ large n

↓ Psuedo polynomial time algo

Fermat's little theorem:

If n is prime then $\forall\ 1 \le a < n$

$a^{n-1} \equiv 1 \pmod{n}$

Contrapositive: $\exists\ a \in 1 \le a < n \ni a^{n-1} \not\equiv 1 \pmod{n}$

then n is not a prime

Possibility of error one-sided → If no. of maybe k then error k-sided

i.e. Said prime but is composite (maybe)

Said composite must be composite

Psuedoprime to base a:

A composite integer n is psuedoprime to base a whenever $a^{n-1} \equiv 1 \pmod{n}$ ○ $a > 1$

Absolute psuedoprime (Carmichael number)

A composite integer n is called absolute psuedoprime then ○ ~~~~ $a^{n-1} \equiv 1 \pmod{n}$

$\forall\ a \in [1, n) \ni \gcd(a, n) = 1$

* If composite $n \in \mathbb{Z}$ is not an absolute pseudoprime then $\exists\ b \in [1, n)\ \ni \gcd(b, n) = 1$ & $b^{n-1} \not\equiv 1 \pmod{n}$

* Note ∃ only 646 carmicheal numbers below 10^9. so we can safely assume n is not a carmicheal number

∴ $\exists\ b \in [1, n)\ \ni \gcd(b, n) = 1$ & $b^{n-1} \not\equiv 1 \pmod{n}$

n is composite & not a carmicheal number

then

If $a \in [1, n)$ & $a^{n-1} \equiv 1 \pmod{n}$

i.e. a passes fermat's test

then $\exists\ x \in [1, n)\ \ni x^{n-1} \not\equiv 1 \pmod{n}$

(Here $x \equiv ab \pmod{n}$)

∴ no. of numbers fail the test ≥ no. of no.s pass the test

∴ Pr(Passing the test | n is composite & not carmicheal) ≤ 1/2

→ Abundance of Witness paradigm applied to reduce the error probability to $(1/2)^k$

∴ T.C $O(k \log(n)\ \text{mult}(n))$ ⇒ weakly polynomial.

→ Miller Rabin error (1/4)

→ AKS algo T.C. $O((\log n)^5)$

Pattern matching revisited

q choosen from [1, T)

No. of prime numbers $\approx T/\ln T$

∴ probability to choose a prime $\frac{T/\ln T}{T} = \frac{1}{\ln T}$

∴ we need $\ln T$ iterations to find a prime

(i.e. choose a no. put it back if not prime)

∴ T.C. Monte Carlo: $O(\ln T + n + m)$: Expected Time

Monte Carlo algo usually donot have Expected Times but in this case it does.

★ String matching with DFA (Output all valid shifts)

T : n , P : m

T_i : first i chrs of T , $T_0 = \epsilon$

P_j : first j chrs of P , $P_0 = \epsilon$

$Q = \{0, 1, 2, \ldots m\}$

★ Suffix function : (σ)

$\sigma : \Sigma^* \to Q$

$\sigma(T_i)$: Length of longest prefix of P which is a suffix of T_i

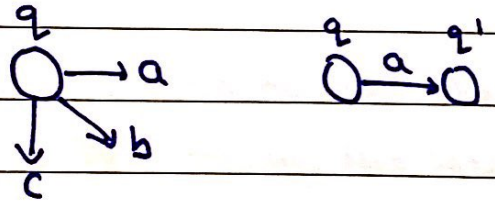
T_i : ababababa

P_i : ababacac

$\therefore \sigma(T_i) = 6$

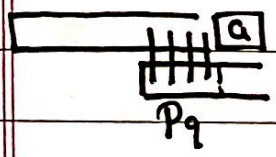
★ DFA/NFA has m+1 states.

i.e. $Q = \{0, 1, \ldots m\}$



$\delta(q, a) = q'$

→ For every state check all the symbols i.e. $|\Sigma|$



$\Delta\ \delta(q, a) = \sigma(P_q a)$

Def. of δ

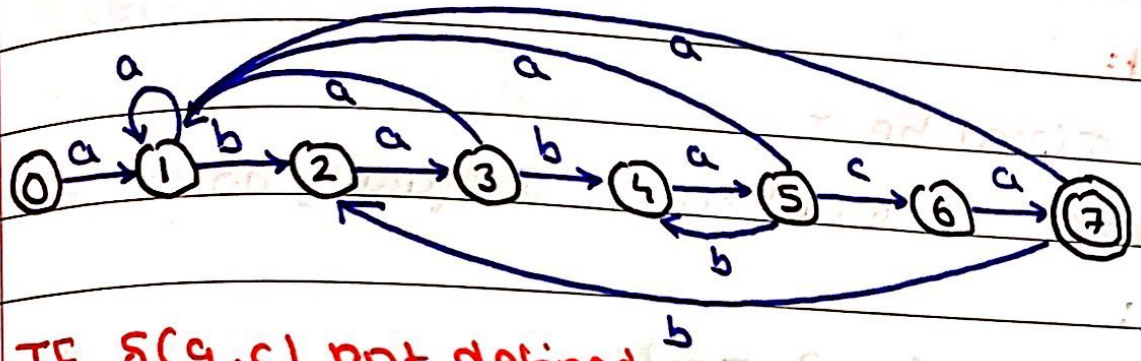
★ DFA in state j $\Leftrightarrow \sigma(T_i) = j$

★ DFA in state m $\Leftrightarrow \sigma(T_i) = m$

$\Leftrightarrow$ valid shift i−m

★ Consider P: ababaca

∴ m = 7 & Σ = {a, b, c}

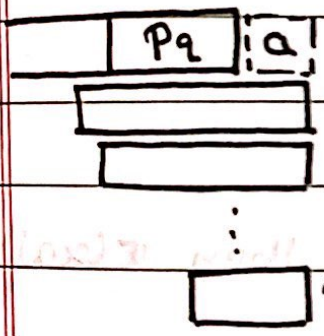


→ IF δ(q, c) not defined

δ(q, c) = 0 By default then

★ Preprocess the DFA

O(m) : States



No. of patterns : O(m)

for each pattern : O(m) checks

∴ Preprocessing Time : O(m³|Σ|)

∴ Space Complexity : O(m|Σ|)

(Transition Table for DFA)

★ Algo for matching given Preprocessed DFA:

```
For i = 1 to n
    q = δ(q, T[i])
    if q == m
        Print "Valid" i - m
```

* Suffix-function Lemma:

→ For $x \in \Sigma^*$ & $a \in \Sigma$ $\sigma(xa) \leq \sigma(x)+1$

Proof:

Let $\sigma(xa)$ be r

If $r=0$ ∵ $\sigma(x) \geq 0$ ∴ Inequality holds

* Note:

$\sigma(x) = \max\{k : P_k \sqsupset x\}$

P_k is suffix of x

∴ $P_r \sqsupset xa$

∴ $P_{r-1} \sqsupset x$

∴ $r-1 \leq \sigma(x)$ (# Def of σ)

∴ $\sigma(xa) \leq \sigma(x)+1$

* Suffix-function recursion Lemma:

→ For $x \in \Sigma^*$ and $a \in \Sigma$ if $q = \sigma(x)$ then $\sigma(xa) = \sigma(P_q a)$

Proof:

$P_q \sqsupset x \Rightarrow P_q a \sqsupset xa$ .. ① $\Rightarrow \sigma(P_q a) \leq \sigma(xa)$

Let $r = \sigma(xa) \Rightarrow P_r \sqsupset xa$ .. ②

∴ $\sigma(xa) \leq \sigma(x)+1$

$\Rightarrow r \leq q+1$

$\Rightarrow |P_r| \leq q+1 = |P_q a|$

① & ② $\Rightarrow P_r \sqsupset P_q a$

& ③ $\Rightarrow r \leq \sigma(P_q a)$ } Def

$\Rightarrow \sigma(xa) \leq \sigma(P_q a)$

∴ $\sigma(xa) = \sigma(P_q a)$

* Final State Function (ϕ)

$\phi : \Sigma^* \to Q$

$\phi(\epsilon) = q_0$

$\phi(wa) = \delta(\phi(w), a)$

* If DFA accepts string w then it ends up in state $\phi(w)$

* T.P.T $\phi(T_i) = \sigma(T_i) \quad \forall i \leq n$

→ Proof:

Induction Base : $i = 0 \quad \phi(T_0) = 0 = \sigma(T_0)$

Induction Hypo : $\phi(T_i) = \sigma(T_i) = q$

Induction Step: $\phi(T_{i+1}) = \phi(T_i a)$

$= \delta(\phi(T_i), a)$ (#Def)

$= \delta(q, a)$

$= \sigma(P_q a)$

$= \sigma(T_i a)$ (#Lemma)

$= \sigma(T_{i+1})$

	Preprocess time	space	Query Time
* DFA matching :	$O(m^3\lvert\Sigma\rvert)$	$O(m)$	$O(n)$
KMP :	$O(m)$	$O(m)$	$O(n)$