$$n_t \leq n_t \left(1 - \frac{1}{|I^*|}\right)^t$$

if $x < 1$,

$(1-x) \leq e^{-x}$

$$\therefore n_t \leq n\left(1 - \frac{1}{|I^*|}\right)^t \leq n e^{-t/|I^*|}$$

greedy will stop when

$$\frac{t}{|I^*|} \leq \log n$$

$$\frac{|I|}{|I^*|} \leq \log n \quad (\text{since } t = |I|)$$

## DYNAMIC PROGRAMMING

given a DAG with edge weights, find shortest path from u to v in the topological sorting.

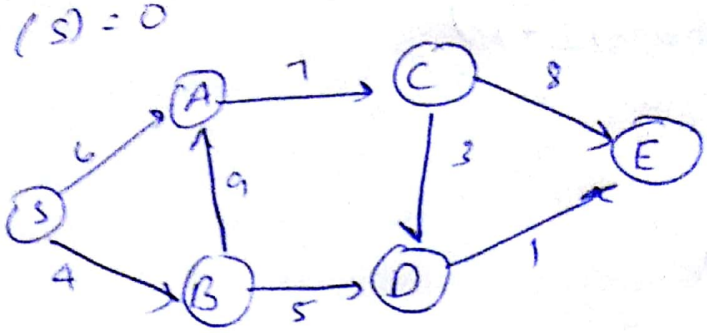
$$L(s,E) = \min_{(u,E) \in \text{Edgeset}} \{ L(s,u) + w(u,E) \}$$

i.e., for all neighbours of ending node E, the ~~for~~ shortest path till the neighbours + weight connecting to E and take minimum of them.

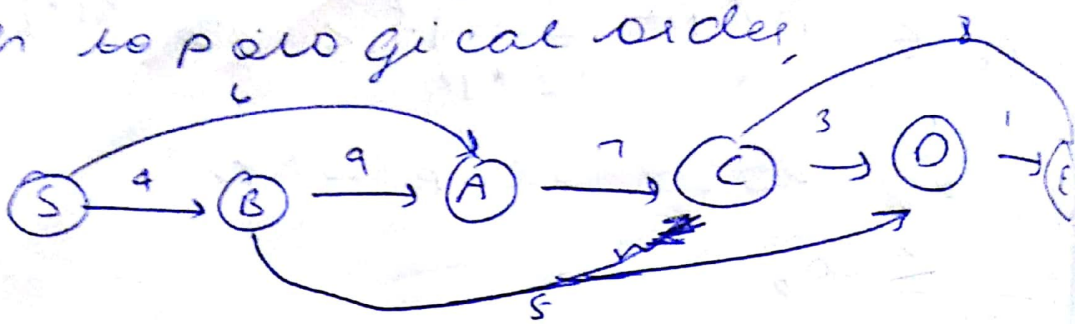
loop in the topological sorted order.

$$L(u) = \min_{(v,u) \in E} \{ L(v) + w(v,u) \}$$

$L(S) = 0$




In topological order,




$L(B) = 4$

$L(A) = \min \begin{cases} L(S) + 6 \\ L(B) + 9 \end{cases} = 6$

$L(C) = \min \begin{cases} L(A) + 7 \\ \cancel{L(B)} \end{cases} = 13$

$L(D) = \min \begin{cases} L(C) + 3 \\ L(B) + 5 \end{cases} = 9$

$L(E) = \min \begin{cases} L(C) + 8 \\ L(D) + 1 \end{cases} = 10$

## Longest increasing subsequence

Input: sequence of nos. $a_1, a_2, a_3, \ldots, a_n$

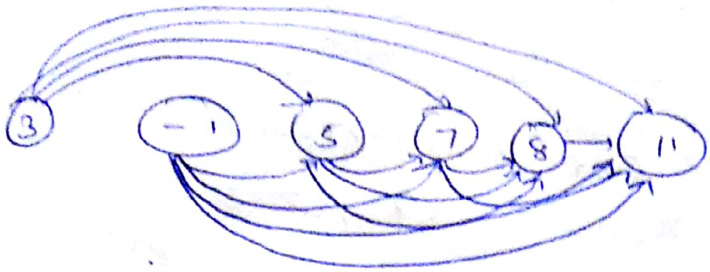
Output: $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_k}$

$i_1 < i_2 < \ldots < i_k$

if $j > t$, $a_{i_j} > a_{i_t}$

Create a DAG.

eg. 3 -1 5 7 8 11




LIS will be the longest ~~subsequence~~ path in the DAG

If suppose we want to find LIS ending at a defined $a_i$

$$L(i) = \max_{(j,i) \in E} \{ 1 + L(j) \}$$

If we don't specify an $a_i$, LIS will be max. of all $L(i)$'s

## Edit distance

~~Spell~~

given string operation edit, delete, overload

eg. EARTH → HEART (2)

Edit distance is min. ~~edit~~ no. of operations required to convert.

Align the 2 strings

_EARTH
HEART_

$x = x_1 x_2 x_3 \ldots x_n$

$y = y_1 y_2 \ldots y_m$

$E(i,j)$ = Edit distance between $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$

$E(i,0) = i$

$E(0,j) = j$

alignment could end as

(i) $\overline{y_j}$ OR(ii) $x_i$ — OR(iii) $x_i$ / $y_j$

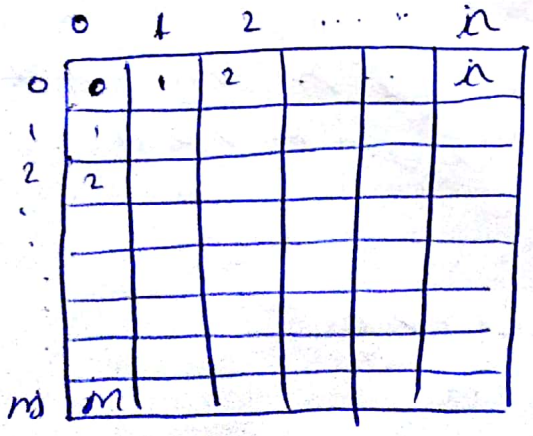
~~If (i) $E(i,j) =$~~

$$E(i,j) = \begin{cases} 1 + E(i, j-1) & \text{, in case (i)} \\ 1 + E(i-1, j) & \text{, in case (ii)} \\ E(i-1, j-1) + \text{diff}(x_i, y_j) & \text{in case (iii)} \end{cases}$$

↓

if $x_i$ is same as $y_j$, add 0, else add 1.

$$\text{diff}(x_i, y_j) = \begin{cases} 0, & \text{if } x_i = y_j \\ 1, & \text{otherwise} \end{cases}$$



eg. X = EARTH

Y = HEART

| | | E | A | R | T | H |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| H 1 | 1 | 1 | 2 | 3 | 4 | 4 |
| E 2 | 2 | 1 | 2 | 3 | 4 | 5 |
| A 3 | 3 | 2 | 1 | 2 | 3 | 4 |
| R 4 | 4 | 3 | 2 | 1 | 2 | 3 |
| T 5 | 5 | 4 | 3 | 2 | 1 | 2 |

# chain matrix multiplication

$M_1, M_2, M_3 \ldots M_N$

$M_i$ is a matrix $P_{i-1} \times P_i$

$M_{P_0 \times P_1}\ M_{P_1 \times P_2}\ M_{P_2 \times P_3} \cdots M_{P_{n-1} \times P_n}$

$\frac{1}{n}\binom{2n-2}{n-1}$ ways to parenthesize the product (catalan)

Number of ways to parenthesize n matrices

$$N(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} N(k)\, N(n-k) \end{cases}$$

(matrix multiplication is associative)

eg. $M_{100\times50}\ M_{50\times10}\ M_{10\times100}\ M_{100\times1}$

(i) $\left(\left(M_{100\times50}\ M_{50\times10}\right) M_{10\times100}\right) M_{100\times1}$

Total cost = 50,000 + 100000 + 10000
= 160,000 (using pg 2)

(ii) $\left(M_{100\times50}\left(M_{50\times10}\left(M_{10\times100}\ M_{100\times1}\right)\right)\right)$

Total cost = 1000 + 500 + 5000 = 6500

(iii) $\left(\left(M_{100\times50}\ M_{50\times10}\right)\left(M_{10\times100}\ M_{100\times1}\right)\right)$

50000 + 1000 + 1000 = 52,000

Let $C_{ij}$ be optimum cost for multiplying $M_i \ldots M_j$

We need $C_{1n}$

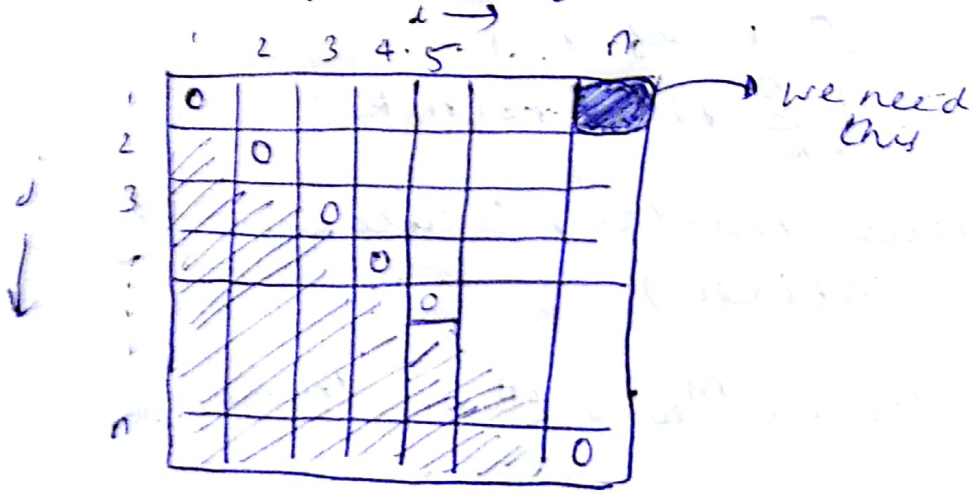
Base case:

$C_{ii} = 0$

$$C_{ij} = \min_{i \le k \le j} \{ C_{ik} + C_{(k+1)j} + P_{i-1} P_k P_j \}$$

$P_{i-1} P_k P_j$ : cost after 2 matrices are formed

matrice of i vs. j:




eg. $M_{100 \times 50}$ $M_{50 \times 10}$ $M_{10 \times 100}$ $M_{100 \times 1}$

$n = 4$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 50,000 | 1.5e5 | 6.5e3 |
| 2 | | 0 | 5e4 | 1500 |
| 3 | | | 0 | 1e3 |
| 4 | | | | 0 |

not needed

$P_0 = 100$
$P_1 = 50$
$P_2 = 10$
$P_3 = 100$
$P_4 = 1$

$$C_{12} = \min_{1 \le k \le 1} \{ C_{1k} + C_{(k+1)j} + P_{i-1} P_k P_j \}$$

$$= C_{11} + C_{22} + P_0 P_1 P_2$$

$$C_{23} = C_{22} + C_{33} + P_1 P_2 P_3$$

$$C_{24} = 50,000$$

---

$$C_{13} = \min \begin{cases} At\ k=1, & C_{11} + C_{23} + P_0 P_1 P_3 \\ At\ k=2, & C_{12} + C_{33} + P_0 P_2 P_3 \end{cases}$$

$$C_{24} = \min \begin{cases} At\ k=2, & C_{22} + C_{34} + P_1 P_2 P_4 \\ At\ k=3, & C_{23} + C_{44} + P_1 P_3 P_4 \end{cases}$$

---

$$C_{14} = \min \begin{cases} k=1, & C_{11} + C_{24} + P_0 P_1 P_4 \\ k=2, & C_{12} + C_{34} + P_0 P_2 P_4 \\ k=3, & C_{13} + C_{44} + P_0 P_3 P_4 \end{cases}$$

We need to fill $n^2$ cells. Filling one cell takes $O(n)$. Therefore, it is $O(n^3)$

We can get worst case by getting max. of all.

Max independent set in a tree (MIS)

$G = (V, E)$

$V' \subseteq V$

$E' \subseteq E$

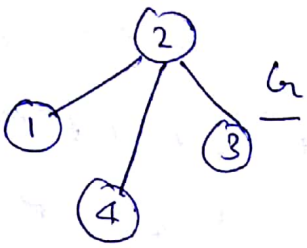
$(E' \subseteq V' \times V')$

$G' = (V', E')$ is a subgraph.

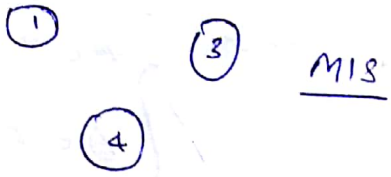
Spanning subgraph is a subgraph with all vertices.

Independent set in a graph is ~~an induced~~ are vertices that are not directly connected by an edge. (i.e., after inducing all edges, it is an empty set of edges.

eg.




G




MIS

consider a tree.

Let $C_u$: size of MIS ~~rooted~~ rooted at u

we need $C_{root}$

Now the entire MIS can either contain $C_u$ or not.

$$C_u = \text{Max}\begin{cases} \left[1 + \sum_{v \in \text{grandchildren}(u)} C_v\right], & \text{if MIS contain } u \\ \sum_{v \in \text{children}(u)} C_v, & \text{if MIS does not contain } u. \end{cases}$$

Termination condition, $C_{leaf} = 1$

LINEAR PROGRAMMING

maximize $(x+5y)$

subject to :

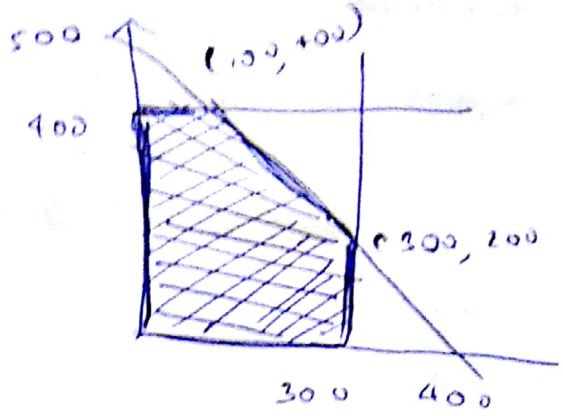
$x \geq 0$

$y \geq 0$

$y \leq 300$

$y \leq 400$

$x+y \leq 500$




Let $x+5y = c$ where $c$ is max. Call this is ||el to $x+5y = 0$.

Plot $x+5y = 0$ and the line ||el to it that intersects with the area defined by the constraints (found by moving the line). You will find the max defined by how much you moved the line. It will always hit a vertex in the ~~concave~~ polygon defined by the graph of the constrained area. (Primal version of LP)

Also,

$x+5y = (x+y) + 4y \leq 2100$

$\leq 500 \quad \leq 1600$

(Dual version of LP)

Dual version of LP : Defining ~~constraint~~ function in terms of constraints $\lambda_1, \lambda_2, \ldots \lambda_n$, to get

max. upper bound. Find coeff. $\lambda_i$ where the bound is least.

eg. $\lambda_1$ $x \leq 300$
$\lambda_2$ $y \leq 400$
$\lambda_3$ $x + y \leq 500$

$\lambda_1 = 1$
$\lambda_2 = 5$
$\lambda_3 = 0$

$+ 0(x+y)$
$\Rightarrow \quad 1(300) + 5(400) = 2300 = 2300$

$\therefore \leq 2300$

Dual is minimalization problem while primal is to maximize.

eg. $x^2 + y^2$ (maximise)

Let $x^2 + y^2 = c^2$

Is a circle ~~through~~ with centre at origin with radius c.

so we need to draw concentric circles ~~until~~ as long as we stay in constraints.

eg. $x_1 + 2x_2 - x_3$

$x_1, x_2, x_3 \geq 0$

$\lambda_1$ $x_1 + x_2 \leq 100$
$\lambda_2$ $x_1 + x_3 \leq 50$
$\lambda_3$ $x_2 + x_3 \leq 80$

$\lambda_1 + \lambda_2 = 1$
$\lambda_1 + \lambda_3 = 2$
$\lambda_2 + \lambda_3 = -1$

$\Rightarrow \lambda_1 = 2$

$\lambda_2 = -1$

$\lambda_3 = 0$

- we $\lambda$'s alone help.

eg. max. $x_1 + 2x_2 + x_3$

$x_1, x_2, x_3 \geq 0$

$\lambda_1$ $x_1 + x_2 \leq 100$

$\lambda_2$ $x_1 + x_3 \leq 150$

$\lambda_3$ $x_2 + x_3 \leq 50$

$\lambda_1 + \lambda_2 = 1$

$\lambda_1 + \lambda_3 = 2$

$\lambda_2 + \lambda_3 = 1$

$\Rightarrow \lambda_1 = 1$

$\lambda_2 = 0$

$\lambda_3 = 1$

$\therefore x_1 + 2x + x_3 \leq$ ~~150~~ $\overset{\lambda_1}{1}(100) + \overset{\lambda_2}{0}(150) + \lambda_3(50)$

$\leq 150$

# RANDOMIZED ALGORITHM

consider a family of algorithm
$\{A_1, A_2, \ldots, A_k\}$

consider inputs $\{x_1, x_2, \ldots, x_k\}$ which correspondingly are inputs that solve $A_1, A_2, \ldots, A_k$, say that $X$ is the set of all input possible

if $|x_i| \leq \frac{2}{3}|x|$

if $x \in X \Rightarrow x \in x_i$ for $\frac{2}{3}$ i's

choose $j_1, j_2, \ldots j_k \in_R [1 \ldots R]$
Execute $(A_{j_1}(x), A_{j_2}(x), \ldots, A_{j_k}(x))$
and choose majority

## Primality testing

Input : $N > 2$
Output : yes if N is prime
no if N is composite.

naive algo ; test divisibility
$\forall i \leq \sqrt{N}$, $i/N$ ?
$O(\sqrt{N})$

however it isn't efficient because if N is a 512 bit no., $N \sim 2^{512}$
$O(\sqrt{N}) \approx \sqrt{2^{512}} = 2^{256}$

$O(\sqrt{N}) = O(2^{\frac{\log N}{2}})$ not efficient (not polynomial)
we need $O(\log N)$

## Miller Rabin

step 1: Write $(N-1) = 2^q s$, $s$ is odd (if it was even, stop)

step 2: Choose $k$ random numbers say $a_1, a_2, \ldots a_k \in [2, \ldots, N-1]$

step 3: Create a 2D array $M_{k \times (q+1)}$
Fill with natural nos.,
$M_{ij} = a_i^{s \cdot 2^j} \bmod N$

step 4: Read ~~the array M~~ each row of the array M from right to left till a number that is not 1 is encountered.
If ∃ a row s.t. the first no. from right to left that is $\neq 1$ is not $N-1$, output ("COMPOSITE")
else output ("PRIME").

Each row is a separate algorithm and is hence primality test.

If no. is prime, it always outputs prime. If no. is composite, probability of being right is $1 - \frac{1}{2^k}$

Theorem: If no. N is prime, it always outputs prime

Proof: $\forall a \in [2 \ldots (N-1)]$, the no. you get by a will always have the property that the first number that is not 1 will be $N-1$

The last element will always be 1 because it is

$$a^{s \cdot 2^r} \bmod N = a^{N-1} \bmod N$$
$$= 1 \quad \text{if } N \text{ is prime due to Fermat's theorem}$$

By induction, show that a number that is being read from the row from right to left, the next no. will be 1 or $N-1$.

Let the next no. being read be $x$

$$x = a^{s \cdot 2^{r-1}} \bmod N$$

$x^2 \bmod N = 1$ since every cell after one cell is the number squared mod N.

If $x^2 \bmod N = 1$

$$\Rightarrow x^2 - 1 \equiv 0 \pmod N$$
$$(x+1)(x-1) \equiv 0 \pmod N$$

$x$ is a no. between 0 and $N-1$ since every no. in the cell is

We that ~~[illegible]~~ (due to mod N)

~~$(x+1)(x-1)$ is in range~~

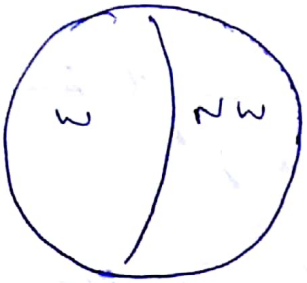
therefore ~~[illegible]~~ $x = 1$ or $x = N-1$

since $(x+1)(x-1) \equiv 0 \pmod N$

~~[illegible]~~ implies one term has to be zero. (or $(x+1)(x-1)$ is a multiple of N)

Theorem: If it is composite, it outputs COMPOSITE with probability $\geq 1 - \frac{1}{2^k}$

Proof:




W → witness (values of 'a' which won't force algo to output composite)

NW → non witnesses

If we show that $|W| \geq |NW|$ then it ~~p~~ is proved because ~~if~~ since there is a probability of 1/2 ~~of [illegible]~~ choosing an 'a' that lies in W or NW.

Choosing 'k' elements 'a_i' all of which ~~[illegible]~~ ~~[illegible]~~ lie in NW is $\frac{1}{2^k}$

Show one to one mapping from NW to W

$f: NW \rightarrow W$

because it implies there are

atleast as many w as NW.

To show 1-1 mapping:

~~$\exists t$ s.t. $t \in NW$~~

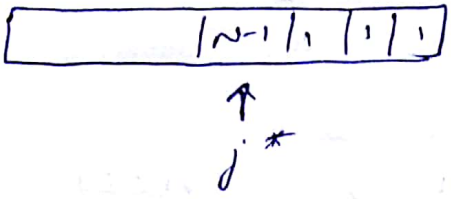
$\rightarrow \exists t$ s.t $t \in W$

$\rightarrow \forall a \in NW$, $at \in W$

This will be a 1-1 mapping.

1 is a non witness.

Take any non-witness




(~~Let $h \in NW$~~ Let $h \in NW$ and ~~$h = ?$~~

$h^{s 2^{j*+1}} \bmod N$)

~~$h$~~ $h$ is no. at index $j^*$

$h^{s 2^{j*+1}} \bmod N = 1$ } (separate statements)

$h^{s 2^{j*}} \bmod N = N-1$ }

Assume N is composite

$N = pq$ (where $GCD(p,q) = 1$)

Let $t \equiv h \pmod p \equiv (p-1) \pmod p$

and $t \equiv 1 \pmod q$

t exists due to chinese remainder theorem, and t exists between p and q $\Rightarrow$ t exists between 0 and N-1.

$t \equiv s.2^{j*+1} \bmod N = ?$

$t^{s \cdot 2^{j*+1}} \bmod p = 1$ ⟶ *

$t^{s \cdot 2^{j*+1}} \bmod q = 1$

since p and q are coprime

and $t = m_1 p + 1$ and $t = m_2 q + 1$

$\Rightarrow p$ divides $m$

$\Rightarrow t = m p q + 1$

$\Rightarrow t^{s \cdot 2^{j*+1}} \bmod N = 1$

$t^{s \cdot 2^{j*}} \bmod N = ?$

Now $t^{s \cdot 2^{j*}} \equiv h^{s \cdot 2^{j*}} \pmod p$

$\equiv N \pmod p$

$\equiv p-1 \pmod p \longrightarrow (a)$

$t^{s \cdot 2^{j*}} = 1 \pmod q \longrightarrow (b)$

$t^{s \cdot 2^{j*}} \bmod N \neq 1$

(Proof :

Contradiction :

If $t^{s \cdot 2^{j*}} \bmod N = 1$

$\Rightarrow t^{s \cdot 2^{j*}} \bmod p = 1$

violates (a) )

$t^{s \cdot 2^{j*}} \bmod N \neq N-1$

(Proof :

violates (b) )

$\therefore t \in W$ .

If $a \in NW$

at ?

$(at)^{s \cdot 2^{i+1}}$ mod $N = 1$ since a is a non witness.

$(at)^{s \cdot 2^{i}}$ mod $N$ = ?

$a^{s \cdot 2^{i}} = 1$ or $N-1$ since $a \in NW$

$\Rightarrow (at)^{s \cdot 2^{i}}$ mod $N$ = (1 or $(N-1)$) * ($\neq 1$, $\neq N-1$)

[since $(N-1)^2$ mod $N = 1$]

$\Rightarrow (at)^{s \cdot 2^{i}}$ mod $N \neq 1$, $\neq N-1$

Therefore, $at \in W$

Hence proved. —

$\therefore |W| \geq |NW|$ since 1-1 mapping established.

# POLYNOMIAL TIME REDUCTIONS

A problem is NPC (NP complete) if we are able to show that

$A \leq_p B$

if $\exists$ a function $f: \Sigma^* \longrightarrow \Sigma^*$ s.t.

$\forall x \quad A(x) \iff B(f(x))$

↳ iff

if $\exists$ a $\lambda$function $f: \Sigma^* \longrightarrow \Sigma^*$ s.t.

$\forall x, \quad x \in A \iff f(x) \in B$

then

$A \leq_m B$

↓

reduced mapping

then $A \leq_p B$ efficiently computable → polynomial time reducible

eg. A: satisfiability problem (or a variant of it)

You will be given a boolean formula $\phi$ in conjuctive normal form (CNF) such that it is satisfiable with exactly 3 literals per clause ~~and $\phi$ is satisfias~~

$3\text{-SAT} = \{ \phi \mid \phi$ is in CNF, with exactly 3 literals per clause and $\phi$ is satisfiable $\}$

CNF is product of sums

$\phi : (x_1 \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$

$\wedge (\quad) \wedge \quad \wedge (\quad)$

Input : 3-CNF $\phi$

Output : yes if $\phi$ is satisfiable
NO otherwise

i.e., if you create a truth table, atleast one row is true

eg. (i) $\phi = (x_1 \vee x_2 \vee x_3)$

if $x_1$ : true, then $\phi$ is satisfiable

(ii) $\phi = (x_1 \vee x_1 \vee x_1) \wedge (\bar{x_1} \vee \bar{x_1} \vee \bar{x_1})$

is not satisfiable

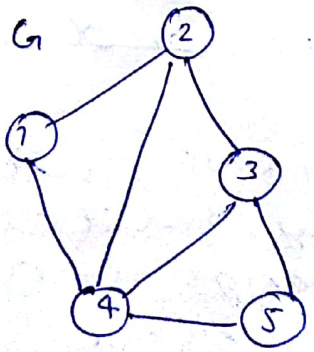
In (i) $\phi \in$ 3-SAT

In (ii) $\phi \notin$ 3-SAT

Consider problem B: Clique problem
Language is graph G and a +ve integer k such that G has a clique of size k (G has complete ~~graph~~ subgraph of size k)

CLIQUE = $\{ \langle G, k \rangle \mid$ G has a clique of size k $\}$

eg.




$\langle G, 1 \rangle \in$ Clique

$\langle G, 2 \rangle \in$ Clique

$\langle G, 3 \rangle \in$ "

$\langle G, 4 \rangle \notin$ "

Write a program that outputs yes if there exists clique of size k.

If you can write a solution to B, you can solve A (satisfiability problem)

theorem: $3\text{-SAT} \leq_p$ CLIQUE

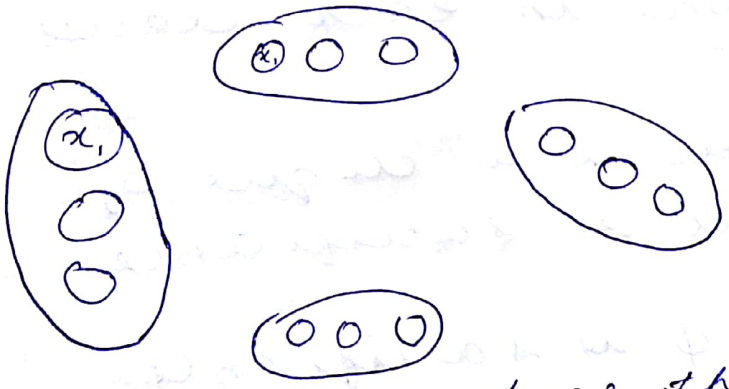
convert $\phi$ to $\langle G, k \rangle$

such that $\phi \in 3\text{-SAT} \iff \langle G, k \rangle \in$ CLIQUE

consider $\phi : (x_1 \vee x_2 \vee \bar{x_3}) \wedge (\ . \vee . \vee \ ) \wedge ( \ ) \ldots\ldots$

with $k$ clauses and $n$ variables

create $G$ on $3k$ vertices (due to $3k$ literals)




first define edges that should not be connected:

(i) There should be no edge between 2 vertices of the same clause

(ii) No edges between vertices that are complimentary to each other. i.e., no edge connects $x$ & $\bar{x}$

all other edges exist

set $k = k$ and ask whether a clique of size $k$ exists

Prove that such a clique exists iff $\phi$ is satisfiable.

~~Suppose there is a size k~~

You can have maximum size clique of size k. where one in each clause is connected. All of those will not be complimentary. Make them all true.

This has to imply that $\phi$ is satisfiable since atleast one in each clause is true.

∴ If there is a clique of size k, it is satisfiable.

Suppose $\phi$ is satisfiable, ~~you can assign~~ each clause is satisfiable and atleast one literal in every clause is true. Pick the first literal that is true in every clause. You would have picked k literals. So you would have picked k literals in the graph. These k literals have to form a clique because there can't be a missing edge and satisfies (a) and (b)

$\therefore \phi \in 3\text{-SAT} \iff \langle G, k \rangle \in$ CLIQUE

$\forall L \in NP$, if $L \leq_p 3\text{-SAT}$

$\Rightarrow$ 3-SAT is NP Hard.

A is NP-Hard if $\forall L \in NP$, $L \leq_p A$ (Hardness of L is upper bounded by Hardness of A)

A is NP complete ~~also~~ and NP Hard

$\Rightarrow$ A is NP complete.

$3\text{-SAT} \leq_p$ CLIQUE

$L \leq_p 3\text{-SAT} \leq_p$ CLIQUE

(It is transitive)

If you take an NP complete program and reduce it to your problem, then even that is NP hard.

~~it has proof p~~

A problem x is interesting if ~~and~~ it has an efficient algorithm A such that $A(x,p) = 1$ iff $x \in L$

~~$\exists L$ is in p~~ $\Rightarrow L \in$ ~~NP~~ P

~~$\exists x$ $\forall x$ $\exists$ proof p~~

$\forall x \exists p \mid \exists$ efficient algo A s.t. $A(x,p) = 1$ iff $x \in L$

$\exists) L \in NP$

# NUMBER THEORETIC ALGORITHMS

- Primality testing (done)
- GCD, Zib.
- Chinese remainder theorem

## G.C.D

Input: a, b (a ≥ b)
Output: g = gcd(a, b)

Euclid's GCD:

Recursive algorithm

Euclid(a, b): → (a ≥ b)
  if (b == 0) return a;
  else return (Euclid(b, a %. b));

Ipy:

% g = gcd(a, b)
d = gcd(b, a %. b)
then
~~then~~ g = d

g = gcd(a, b) ⇒ g | a & g | b

⇒ g | a %. b

since a %. b = a − mb
= g·a₁ − mgb₁ (where a = g a₁, b = g b₁)
= g(a₁ − mb₁)

∴ g | a %. b

⇒ g is gcd(b, a %. b)

$T(n) = 1 + T(m)$

where $n = \log_2 a$, $m = \log_2 (a \% b)$

But this is tough to solve.

$a \% b \leq \frac{a}{2}$

Proof:

If $b \leq a/2$, then it is true

else if $b \geq a/2$, then $a \% b = a - b \leq a/2$ (true)

$O(\log_2 a)$ time.

For a tighter bound,

if Euclid-algo takes $k$ (recursion) steps, then $a \geq F_{k+2}$, $b \geq F_{k+1}$

At $k = 0$, $F_1 = 0$, $F_2 = 1$

assume theorem is true upto $k-1$

So if it is true till Euclid $(b, a \% b)$

$\Rightarrow b \geq F_{(k-1)+2} = F_{k+1}$

$a \% b \geq F_{(k-1)+1} = F_k$.

Is $a \geq F_{k+2}$

Now $a \geq b + (a \% b)$

(since $a = mb + r(a \% b)$ and $m \geq 1$)

$\Rightarrow a \geq F_{k+1} + F_k$ (from induction hypothesis)

$\Rightarrow a \geq F_{k+2}$

8) show that this is right
i.e., if we store since
$a = F_{k+2}$, $b = F_{k+1}$, then it
takes k recursion.

Fibonacci no:

Input : n
Output : $Fib_n$

```
Fib(n)
    if n == 0 return 0
    if n == 1 return 1
    else return ( Fib(n-1) + Fib(n-2))
```

$T(n) = T(n-1) + T(n-2) + 1$

This is exponential

```
Fib(n)
    if n == 0 , F[0] = 0
    if n == 1 , F[1] = 1
    for i = 2 to n
        F[i] ← F[i-1] + F[i-2]
Print F[n]
```

This is linear time.
To do better use linear algebra.

$F_0 = 0$

$F_1 = 1$

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$$\begin{bmatrix} F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$$\begin{bmatrix} F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^3 \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} F_{n} \\ F_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$O(\log n)$

(since we can calculate $(\text{matrix})^2$, $(\text{matrix})^4 \ldots$)

But the nos: can be very large and can't be stored in that case.

~~Algo~~

$Fib(0) = 0$, $Fib(1) = 1$

$Fib(n) = Fib(n-1) + Fib(n-2)$

Assume $fib(n) = \rho^n$

$\rho^n = \rho^{n-1} + \rho^{n-2}$ where $\rho > 0$

~~In the lowest case~~

$\Rightarrow \rho^2 = \rho + 1$ (dividing by $\rho^{n-2}$)

$\rho^2 - \rho - 1 = 0$

$\Rightarrow \rho = 1 \pm \frac{\sqrt{5}}{2}$

$\therefore fib(n) = \lambda_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \lambda_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$

$\therefore fib(0) = \lambda_1 + \lambda_2 = 0 \Rightarrow \lambda_2 = -\lambda_1$

$fib(1) = \lambda_1 \left(\frac{1+\sqrt{5}}{2}\right) + \lambda_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1$

$\Rightarrow \lambda_1 \left(\frac{1+\sqrt{5}}{2}\right) - \lambda_1 \left(\frac{1-\sqrt{5}}{2}\right) = 1$

$\Rightarrow \lambda_1 = \frac{1}{\sqrt{5}}, \quad \lambda_2 = -\frac{1}{\sqrt{5}}$

$\therefore fib(n) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$

## Chinese remainder theorem

$x \equiv a_1 \pmod{n_1}$

$x \equiv a_2 \pmod{n_2}$

$x \equiv a_k \pmod{n_k}$

solve for $x$ given $n_i$ and $a_i$

eg. $x \equiv 1 \mod 3$

$x \equiv 2 \mod 5$

$x \equiv 3 \mod 7$

$x = 5\,2 + m\,105$.

$\therefore x = M + m \prod_{i=1}^{k} n_i$

now to find $M$?

Let $x \equiv 1 \mod n_1$

$x \equiv 0 \mod n_2$

$\vdots$

$x \equiv 0 \mod n_k$

$\left(\prod_{j=2}^{k} n_j\right) V$

$V = \left[\left(\prod_{j=2}^{k} n_j\right)^{-1} \mod n_1\right]$

$[1, 0, \ldots 0] = M_1$

$[0, 1, \ldots .0] = M_2$

$\vdots$

$[0, \ldots\ldots, 1] = M_k$

so solution will be

$a_1 M_1 + a_2 M_2 + \cdots + a_k M_k \mod \prod_{i=1}^{k} n_k$

$= \left[\sum_{i=1}^{k} \left\{ a_i \left\{ \left[\prod_{j=1}^{k} n_j\right] / n_i \right] \left[ \left( \frac{\prod_{j=1}^{k} n_j}{n_i} \right)^{-1} \mod n_i \right] \right\} \right] \mod \prod_{i=1}^{k} n_k$

inverse must exist, so there must be no common factor with $n_i$.

MID II

1. Dynamic Programming
2. Linear "
3. Polynomial time reduction
4. Primality testing & number testing algo.

A. Longest common subsequence (LCS)
B. Polygon triangulation
C. Optimal rod cutting
D. Longest path in a weighted tree.

B) Polygon triangulation:
Find optimum triangulation of a polygon (smallest perimeter)

C) eg: Rod of length 5


(i) Cutting first at 4 then at 2, cost = 9
(ii) Cutting first at 2, then at 4, cost = 8

E. 3-SAT $\leq_p$ Vertex cover
Given a graph, vertex cover is a subgraph where every edge is incident on

inverse must exist, so there must be no common factor with $n_i$.

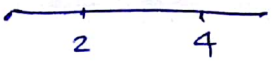
MID II

1. Dynamic Programming
2. Linear "
3. Polynomial time reduction
4. Primality testing & number testing algo.

A. Longest common subsequence (LCS)
B. Polygon triangulation
C. Optimal rod cutting
D. Longest path in a weighted tree.

B) Polygon triangulation:
Find optimum triangulation of a polygon (smallest perimeter)

C) eg: Rod of length 5




(i) cutting first at 4 then at 2, cost = 9
(ii) cutting first at 2, then at 4, cost = 8

E) 3-SAT $\leq_p$ Vertex cover
Given a graph, vertex cover is a subgraph where every edge is incident on

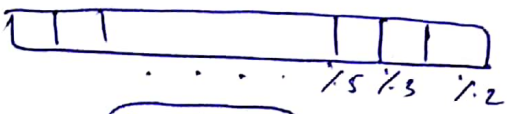
at least one vertex is the cover.
find vertex cover of min size ~~if you know~~
(Binary search)

(F) 3-SAT $\leq_p$ Subset sum
whether ~~a~~ there exists a subset in a given set that add up to a given no.

~~$\{\langle x_1, x_2, x_3, \ldots, x_n \rangle, T\}$~~

(h) Residue number system (RNS)
store ~~no~~ ~~no. i/2, no. i/3~~,
$n \% 2, n \% 3, \ldots$ ~~$n \% i$~~

~~you~~ n digit unique representation.




Sum

| 7 | 5 | 3 | 2 | |
|---|---|---|---|---|
| 5 | 2 | 0 | 1 | |
| 4 | 0 | 0 | 0 | |
| 2 | 2 | 0 | 1 | add digits and mod by respective |

multiply

| 13 | 11 | 7 | 5 | 3 | 2 | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 5 | 2 | 0 | 1 |
| 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| 0 | 0 | 0 | 6 | 0 | 0 | 0 | multiply and mod by respective |

Hence sum and multiply are linear
However to compare, you need Chinese remainder theorem

| Representation | < | + | * |
|---|---|---|---|
| Biary | Fast | Fast | Medium |
| RNS | M | F | F |
| Product of primes | S | S | F |

(H) GCD

give a divide and conquer algo to get GCD (.O(log n)

(split by LSB)