933271081, Chih-Hsiang Wang

1. Reductions

(a) $a \times b = \frac{(a+b)^2 - a^2 - b^2}{2}$ $\Rightarrow$ multiplication $\leq$ squaring

(b) $a^2 = a \times a$ $\Rightarrow$ squaring $\leq$ multiplication

(c) $\frac{1}{x} - \frac{1}{y} = \frac{y-x}{xy}$, let $y = x+1$ $\Rightarrow$ $\frac{1}{x} - \frac{1}{x+1} = \frac{x+1-x}{x(x+1)} = \frac{1}{x^2+x}$ $\Rightarrow$ $x^2 = \frac{1}{\frac{1}{x} - \frac{1}{x+1}} - x$

$\Rightarrow$ squaring $\leq$ reciprocal

(d) b-j, (a).(b) $\Rightarrow$ multiplication and squaring are equivalent

however, we can not reduce reciprocal from squaring

2. Lucas Numbers

$L_n = L_{n-1} + L_{n-2}$, $L_0 = 2$, $L_1 = 1$ $\Rightarrow$ $L_2 = 3$, $L_3 = 4$, $L_4 = 7$ ...

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $F_n$ | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 ... |
| $L_n$ | 2 | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 47 | 76 | 123 ... |

we can see that $L(n) = F(n-1) + F(n+1)$, for $n \in N$

can get $L(n)$ from adding two Fibonacci number,

so $T_L(n) = 2T_F(n) + 1$

$L_n$ is reducible to $F_n$, while $F_n$ is a problem in P ( $O(1.618)^n$ )

$\Rightarrow$ $L_n$ is in P, the algorithm runs in polynomial time in K

3. Roots

(a) $x^2 - x - 1 = 0$ | $x(x-1) = 1$, assume $x$ is a positive integer.

it follows that $x = 1$ and $x-1 = 1$ or $x = -1$, $x-1 = -1$,

however, we can not find such $x$ to fullfill the equation $\Rightarrow$ contradicting with assumption

$\Rightarrow$ no positive integer solutions

(b) assume $x = \frac{p}{q}$, $(p \geq q)$, where $(p,q) = 1$ $\Rightarrow$ $\frac{p^2}{q^2} - \frac{p}{q} - 1 = 0$ $\Rightarrow$ $p^2 - pq - q^2 = 0$ $\Rightarrow$ $(p-q)(p+q) = pq$

because $(p,q) = 1$, p and q cannot be both even

① if p is odd, q is odd : $(p-q)(p+q) = pq$ $\Rightarrow$ contradiction
(even) (even) (odd)

② if p is odd, q is even : $(p-q)(p+q) = pq$ $\Rightarrow$ contradiction
(odd) (odd) (even)

③ if p is even, q is odd : $(p-q)(p+q) = pq$ $\Rightarrow$ contradiction
(odd) (odd) (even)

$\Rightarrow$ no rational solutions

4. Average Case

$A(n) = A(n-1) + 1 + \frac{2}{n}$ ($\frac{2}{n}$ is the probability for next comparison), $A(2) = \frac{3}{2}$

$\Rightarrow A(1) = 0$

$= A(n-2) + (1+1) + (\frac{2}{n} + \frac{2}{n-1})$

$= A(n-3) + (1+1+1) + (\frac{2}{n} + \frac{2}{n-1} + \frac{2}{n-2})$

$= \underbrace{A(1)}_{0} + (n-1) + \sum_{j=2}^{n} \frac{2}{j}$ $\Rightarrow$ for large $n$, $A(n) \to n-1 + 2(\log n + \text{const})$

best case is $n-1$, worst case is $2(n-1)$

the average case is nearer to best case

5. Lower Bound

Procedure Land S (X, n, Max, Min)

if $n = 1$, Max = X[0]

Min = X[0]

if $n = 2$, if X[0] > X[1],

Max = X[0]

Min = X[1]

else, Max = X[1]

Min = X[0]

if $n > 2$, splie X into A, B

Land S (A, $\frac{n}{2}$, Big A, Small A)

Land S (B, $\frac{n}{2}$, Big B, Small B)

Compare (Big A, Big B)

Compare (Small A, Small B)

$T(n) = 2T(n/2) + 2$

$= 2^k T(n/2^k) + \frac{2(1-2^k)}{1-2}$

$= 2^k T(n/2^k) + 2 \cdot 2^k - 2$

let $2^k = \frac{n}{2}$, $2^{k+1} = n$, $k = \log n - 1$

$\Rightarrow T(n) = \frac{n}{2} \cdot T(2) + 2 \cdot \frac{n}{2} - 2$, $T(2) = 1$

$\Rightarrow T(n) = \frac{3}{2}n - 2$

6. Boolean Expression

(a)

FIND ( D(x1, ⋯, xn) )

set X = 0 0 ⋯ 0 .
(n binary number in total)

for 1 to $2^n$

if YS(D) == "Yes"

return D

else

add 1 to binary number X

(b)

FIND ( D(x1, ⋯, xn) ) = $2^n$ · YS ( D(x1, ⋯, xn) )

$\Rightarrow 2^n \cdot O(n^k)$

$\Rightarrow O(2^n \cdot n^k)$

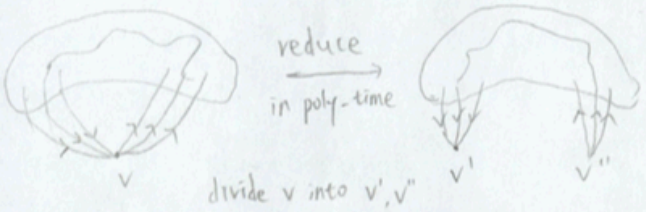
7. Platonic Hamiltonian Circuits

Tetrahedron Cube Octahedron Dodecahedron Icosahedron

There are only five Platonic solids and we can find a Hamiltonian circuit for each

8. s-t Hamiltonian Path

(a) HC − A  HP − B




divide v into v', v''

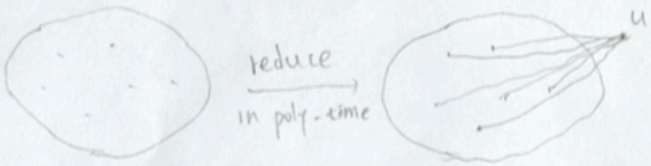
① Hamiltonian Path ∈ NP

in Yes/No problem, can get a non-deterministic alg to check it in polynomial time

② HC ≤ HP  (R)

A − G = (V, E) , B − G' = (V', E') , A(G) = B(G')

(b) HP − G  HC − J




u connects to every vertice

① Hamiltonian circuit ∈ NP (same as above)

② HP ≤ HC

G : $v_1, v_2, \cdots v_m$

J : $v_1, v_2, \cdots v_m, u$

(c)

① TSP ∈ NP

⇒ non-deterministic alg is O(n)

for i = 1, 2, ..., n do

  Guess $x_i \in \{1, 2, \cdots, n\}$ ; {The $i^{th}$ city}

end for

$x_{n+1} := x_1$ ;

{verification stage:}

If $x_1, x_2, \cdots x_n$ are distinct and $\sum_{i=1}^{n} d(x_i, x_{i+1}) \le$ total distance

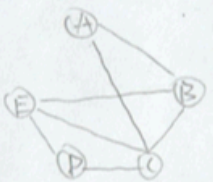
  then "yes" ;

else

  "no"

end if

② TSP ≤ HC (assume HC is NP-complete)




G = (V, E)


G' = (V, E')

Take any instance G = (V, E) for the HC problem

convert it into an instance

G' = (V, E' = V × V, d), total distance of TSP = 0

d(i, j) = { 1 if edge (i, j) ∈ E , 2 otherwise }

⇒ O(n²) as there n(n−1)/2 edges on a complete graph

(reference: https://www.slideshare.net/emrecankucukoglu/tsp-isnpcomplete)

9.

Goal : show that the language GRAPH ISOMORPHISM can be verified in polynomial time

Let the input $x$ be two graphs $G_1$ and $G_2$ and let the certificate $y$ be the indices $\{i_1, i_2, \cdots, i_n\}$. An algorithm $A(x, y)$ verifies GRAPH ISOMORPHISM by executing the following steps :

(a) if the certificate $y$ is a permutation of $\{1, 2, \cdots, n\}$

continue

else $\Rightarrow O(V^2)$

return false

(b) permute the vertices of $G_1$ as given by the given permutation.

Verify that the permuted $G_1$ is identical to $G_2$ $\Rightarrow O(V+E)$

$\Rightarrow$ the verification algorithm $A$ runs in $O(V^2)$

$\Rightarrow$ GRAPH ISOMORPHISM $\in$ NP

(reference : cs.jhu.edu/~cs363/fall2013/assign9_sin.pdf)

10.

(a)

$V_1 - V_2 - V_3$

| | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|
| $V_1$ | 0 | 1 | 0 |
| $V_2$ | 1 | 0 | 1 |
| $V_3$ | 0 | 1 | 0 |

$V_1 - V_3 - V_2$

| | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|
| $V_1$ | 0 | 0 | 1 |
| $V_2$ | 0 | 0 | 1 |
| $V_3$ | 1 | 1 | 0 |

$V_2 - V_1 - V_3$

| | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|
| $V_1$ | 0 | 1 | 1 |
| $V_2$ | 1 | 0 | 0 |
| $V_3$ | 1 | 0 | 0 |

canonical number is the smallest value for the binary number

$\Rightarrow 001001110 \Rightarrow 78$

(b) if we can find the canonical number easily, we can easily plot a specific graph in canonical form, meaning we can check if the graphs are isomorphic by checking the number

(c) because GRAPH ISOMORPHISM $\leq$ finding canonical number

besides, GRAPH ISOMORPHISM $\in$ NP by proof of #9

however, we can not make sure to find a "Yes" in decision problem of Is-canonical in polynomial time, but we can find "No" that

I $\neq$ the canonical number of G in polynomial time because I is $n^2$ $\Rightarrow$ Is-canonical is Co-NP

11. Tautology

① Tautology $\in$ co-NP

construct a TM M with a read-only tape and a work tape

1. Copy the Boolean formula on to the work tape. (linear time)
2. Replace each variable with its value. (linear time)
3. Recursively, replace every pair-wise operation with its value (i.e. $(0 \wedge 1)$ is replaced with 0). Consider that the first replacement phase reduces the size of the formula from $|x|$ to $\log(|x|)$ in $O(|x|)$ steps. Each replacement phase continues to halve the size of the formula to be evaluated in the next phase. (polynomial time)
4. If the final replacement phase yields a 0, halt and reject
5. If the final replacement phase yields a 1, halt and accept.

$\Rightarrow$ M runs in poly-nomial time

② $\overline{SAT} \leq$ Tautology

Given a non-deterministic TM N which decides Tautology in polynomial time

construct a non-deterministic TM M which decides $\overline{SAT}$ in polynomial time as follows

1. On input x, create $x' = \neg x$, (linear time)
2. Run $N(x')$ (polynomial time)
3. If N accepts, halt and accept.
4. Otherwise, halt and reject

$\Rightarrow$ M runs in polynomial time. since $\overline{SAT}$ is coNP-complete. Tautology $\in$ coNP. $\overline{SAT} \leq$ Tautology

$\Rightarrow$ Tautology is coNP-complete

(reference : cs-www.bu.edu/faculty/homer/535-10/homework/hw5-sol-2010.pdf)

* 3-SAT $\in$ NP

a non-deterministic algorithm need only guess an assignment of values and check if it works in polynomial time ( $\leq 3 \times$ number of clauses)

* SAT $\leq$ 3-SAT

convert a cnf-formula F into a 3cnf-formula F', with F is satisfiable $\Leftrightarrow$ F' is satisfiable

Let $c_1, c_2, \cdots, c_k$ be the clauses in F. If F is a 3cnf-formula, just set F' to be F.

Otherwise, the only reasons why F is not a 3cnf-formula are:

• some clauses $C_i$ has less than 3 literals, • some clauses $C_i$ has more than 3 literals

For each clause that has one literal, say $L_1$, change it into $(L_1 \vee L_1 \vee L_1)$

$\Rightarrow$ if F' is satisfiable, the value of $L_1$ must be 1

For each clause that has two literals, say $(L_1 \vee L_2)$, change it into $(L_1 \vee L_2 \vee L_1)$

$\Rightarrow$ if F' is satisfiable, the value of $(L_1 \vee L_2)$ must be 1

For each clause that has more than three literals, say $(L_1 \vee L_2 \vee \cdots \vee L_m)$, use new variables $z_i$, and replace the clause by $(L_1 \vee L_2 \vee z_1) \wedge (\bar{z}_1 \vee L_3 \vee z_2) \wedge (\bar{z}_2 \vee L_4 \vee z_3) \wedge \cdots \wedge (\bar{z}_{m-3} \vee L_{m-1} \vee L_m)$

$\Rightarrow$ if F' is satisfiable, the value of $(L_1 \vee L_2 \vee \cdots \vee L_m)$ must be 1

Finally, for each clause that has three literals, no change.

By construction of F', F is satisfiable $\Leftrightarrow$ F' is satisfiable

Besides, the above conversion takes polynomial time

$\Rightarrow$ SAT is polynomial time reducible to 3-SAT

$\Rightarrow$ 3-SAT is NP-complete

(reference : www.cs.nthu.edu.tw/~wkhon/toc07-lectures/lecture21.pdf)