<10_16_정리본>

2018320161_컴퓨터학과_송대선

- algorithm, Turing Machine, decider, language
- analysis of algorithm, time complexity, space complexity
- quantified boolean formula, generalized geography problem
- Turing reducion, polynomial time reduction
- dealing with hardness, heuristics
- NP Complete problems
- computing with boolean circuits [non-uniform complexity]
- oracle and relativization
- why is the P vs NP question difficult?
- proving P is not equal to NP
- quantum algorithms [Deutsch Josza algorithm, Grover algorithm]
- conditions under which P = NP

Traveling Salesman Problem의 정의 (input, output)

input : undirected graph G = (V, E), distinct nodes in V에 각각 할당된 양수인 실수들 (traveling cost), 양수인 실수 B.

output : yes, if node x에서 모든 node들을 한번씩만 거쳐서 다시 x로 돌아올 때, traveling cost들의 합이 B를 넘지 않는다.
no, otherwise.

input, output, instance의 개념을 알아야 한다.

우리는 이것을 efficiently하게 푸는 방법을 모른다.

brute force방식으로 모든 가능성을 체크하는 방법이 있다.
그러나 이는 n이 커지면 급격히 time complexity가 증가하고, efficient하다는 증명이 없다.

Partition Problem의 정의 (input, output)

input : 공집합이 아닌, 양수인 정수들의 set A, 양수인 정수 x.
output : yes, 만약 A의 subset인 B의 합이 x와 동일하면
no, otherwise

우리는 다만 이것의 verification이 효율적이라는 것만 알고 있다.

TSP와 Partition Problem의 structural similarity

1. 둘 다 solution을 찾기 위한 searching의 과정이 필요하다.
2. 만약에 Correct solution을 찾으면, 그것을 verify하는 것이 쉽다.

a shortest path from Chicago to New York through Pittsburgh
1. a shortest path from Chicago to Pittsburgh
2. a shortest path from Pittsburgh to New York
이 두가지 요소로 이루어져 있다.
->
문제의 structure를 파악함으로 경우의 수가 많아도,
일일이 그 가능성을 살필 필요가 없어진다.

이것은 efficient한 알고리즘이 있다.

optimal solution을 찾기 위해 subproblem들을 많이 제시할 수 있는 것이다.

k-clique problem의 정의
input : an undirected graph G = (V, E)
output : yes, if there is a k-clique in G
no, otherwise

TSP, partition problem, k-clique problem은 전부 NP class에 속한다.

P = NP 이면 어떻게 되는가?
- everything in a world is easy to solve, verify
- easily computable

algorithm과 Turing machine은 equivalent하다.
다만 Turing machine에는 두가지 종류가 존재한다 dTM, nTM이다.
deterministic algorithm, non-deterministic algorithm.

transition function은 computer program에 대응되기도 한다.

Turing machine이 작동하기 위한 init이 있다.
internal configuration은 string으로 표현가능하다.
표현이 가능한가?

universal turing machine :
a turing machine that can simulate any other turing machine for arbitrary input

dTM, nTM
nTM : it`s possible to move different possibilities

dTM은 nTM의 특이 케이스이다.

accepter, decider
accepter(recognizer)
: if w a member of L, it holds at accepting states
  but we don`t know about what happen if w is not a member of L.

decider :
 if w a member of L, it holds at accepting states
 if w is not a member of L, it holds at non-accepting states

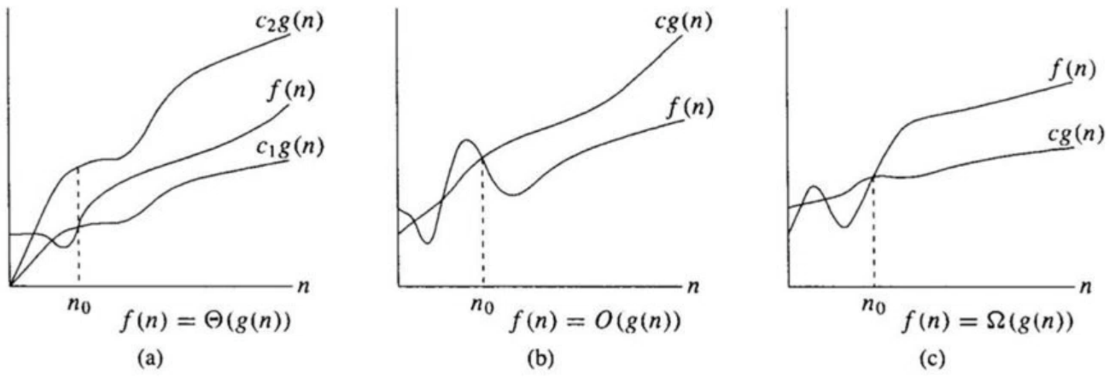
decider = algorithm

- analysis of algorithm, time complexity, space complexity

running time of TM
둘다 longest search path를 선택한다.

사실 running time이라는 개념은 decider(끝이 나야한다.)에만 적용한다.

Asymptotic notations



 P is defined as an infinite union of TIME(n^k)
$P = \cup_k TIME(n^k)$

P is the class of language that are decidable in polynomial time on a deterministic Turing Machine

PATH is the set of all directed graphs in which there is a path from s to t
PATH is example of P

BFS를 통해 PATH문제를 쉽게 해결가능하기 때문이다.

verifier should exist for a certain language
-> need a target language to be verified
verifier is an algorithm V
where this language A consists of strings
such that verifier V accepts <w, c> for some string c

verifier for a language A is an algorithm V,
where A={w|V accepts <w, c> for some string c}

이 V가 polynomial 한 시간내로 끝나면 이 V는 polynomial time verifier이다.

NP : class of a language that have polynomial time verifier
class of a language that do not have polynomial time verifier exists

$NP = \cup_k NTIME(n^k)$
NTIME(t(n))
= {L|L is a language decided by a O(t(n)) time nondeterministic Turing machine}

----
a language L belongs to NP if and only if
there exists a two-input polynomial algorithm A(a verifier)
            and a positive constant c
L is a set of strings in which
there exists a certificate y, with |y| = O($|x|^c$) such that A(x, y) = 1
x is a member of L

y: short certificate(withiness, proof)
----
nondeterministic turing machine whose runtime is polynomialiy bounded
<-> polynomial time verifier

verifier는 각각의 c는 nTM의 움직임을 가이드해준다.
nTM의 움직임은 n^k안에 종료되며, V가 올바르게 작동할 것을 보장한다.

P : the set of language for which membership can be decided polynomialiy
NP : the set of language for which membership can be verified polynomialiy

space complexity : define by counting number of cells used by TM
number of cells != number of steps

SPACE(f(n)) = {L|L is a language decided by an O(f(n)) space dTM}
NSPACE(f(n)) = {L|L is a language decided by an O(f(n)) space nTM}

$PSPACE = \cup_k SPACE(n^k)$
$NPSPACE = \cup_k NSPACE(n^k)$

- **quantified boolean formula, generalized geography problem**

Satisfiablility problem
: is given boolean formula is satisfiable?

SAT = {<ϕ>|ϕ is a satisfiable Boolean formula}
TQBF = {<ϕ>|ϕ is a true fully quantified Boolean formula} (앞에 조건자들이 붙는다.)

SAT는 TQBF의 특이 케이스이다. (SAT는 조건자가 0인 TQBF이다.)
SAT는 PSPACE의 member이다.
SAT를 풀기위한 cell의 수는 polynomial하다.

Theorems / lemmas / facts
- A shortest path from A to B through C - a structural property
- A configuration (internal configuration) is a string that...
- A computation can be represented as a finite sequence of strings.
- A Turing machine can be represented as a string
  (cf. a computer program, an algorithm, etc)
- A Turing machine is a structure The set of languages accepted (decided) by deterministic Turing machines
  = the set of languages accepted (decided) by nondeterministic Turing machines
- The language PATH belongs to P
- 2 definitions for NP (1 in terms of a polynomial verifier 2 in terms of a nTM) are equivalent
- PSPACE = NPSPACE
- The language SAT belongs to NP
- The language TQBF belongs to PSPACE SAT is a special case of TQBF
- P is a subset of PSPACE
- NP is a subset of PSPACE – why?
- P is a subset of NP – why?
- If SAT belongs to P,
  then a certificate of an arbitrary YES instance of SAT can be found efficiently
- Matching problem belongs to P

matching problem
a matching on an undirected graph G=(V, E) is a subset M of E
such that no two edges in M shares a node in V

independent set problem
Given an undirected graph G = (V, E),
a set of nodes S that is a subset of V is independent
if no two nodes in S are joined by an edge

an augmenting path
Given a matching M, a node v is called matched
if there is an edge e in M such that e is incident on v.

Given an undirected graph G = (V, E)
and a matching M on G,
a path p is called an augmenting path for M
if
(1) two end points of p are unmatched
(2) edges of p alternate between edges of M edges not in M

a matching M is maximum if and only if has no augmenting path
 a matching M is maximum, does not exists augmenting path. 두 개를 잘 구분하자

perfect matching :
for every nodes, there exits incident edge in a matching
일종의 1대1대응
선호도는 상관 X

stable matching
선호도가 필요

instability : (m,w), (m`,w`)
서로의 파트너를 버릴 것임 (선호도에 의하여!)

(1) Hamiltonian cycle problem, (2) Euler cycle problem
input : an uindirected graph G = (V, E)
output : if there a cycle such that
    (1) each node is visited exactly once, then YES otherwise NO
        -> efficient algorithm이 없다.
    (2) each edge is visited exactly once, then YES otherwise NO
        -> solvable efficiently

coP, coNP
coP = the set of languages L such that the complement of L is a member of P
coNP = the set of languages L such that the complement of L is a member of NP

the complement of PATH
: the language that consists of all NO instances of the PATH problem
(that asks whether there is a directed path from s to t in a directed graph G)

the complement of SAT
: the language that consists of all UNSATIFIABLE boolean formulas
(i.e., NO instances of SAT)

P is a subset of the intersection of NP and coNP – why?
- for YES instances
- for NO instances

Open problem: P = intersection of NP and coNP?
If P = NP then NP = coNP = P.

TAUTOLOGY
: the set of all boolean formulas that are satisfiable by every assignments
TAUTOLOGY belongs to coNP

complement of TAUTOLOGY
: some assignments can make coTAUTOLOGY unsatisfiable
and that can be easily verified

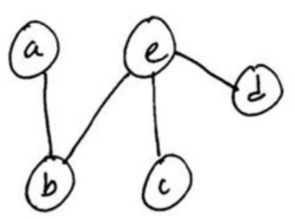
coNP의 새정의 ;
if there exists a polynomial p : N -> N. and a polynomial time TM M,
$x \in L \leftrightarrow \forall_u \in 0, 1^{P(x)}$, M(x, u) = 1

protein folding problem
belongs to NP

NP complete problems
: hardest in NP

Dominating set, partition into triangles
Dominating set G=(V, E)에서 V의 subset인 D는 D에 속해있지 않은 다른 모든 노드들이 최소 한 개의 D의 member와 adjacent해야한다.

domination number γ(G)는 가장 적은 숫자의 Dominating set이다.



$D_1 = \{a, e\}$

$D_2 = \{b, c, d\}$

#P
a set of counting problems

a problem is in #P
if it counts th number of distint solutions such that
(i) every possible candidate for a solution is of polynomial length.
(ii) given a candidate for a solution,
it can be checked in polynomial time whether the candidate is a solution.

NP is a subset of #P
NP에 속한 problem의 속성이 무엇인가?
최소 하나의 solution이라도 존재한다면 그 solution이 certificate로 작동한다.
#P는 solution의 개수들을 묻는 것이다.

what if P = NP
모든 problem들이 efficiently solvable and verifiable

SAT belongs to NP
SAT belongs to PSPACE
for possible evaluation, we can use polynomial amount of space and re-use it again again.

P is a subset of PSPACE
Proof)
if a language L is a member of P,
then by definition, there exists a dTM M
that decides L in O(n^k) for some positive integer k.
M can not see more than O(n^k) cells.
therefore, L must be a member of PSPACE

(in fact, TIME(O(n^k)) is a subset of SPACE(O(n^k)))

HW
if SAT belongs to P(if there exist dTM that decide SAT),
then every problem in NP can be solvable efficiently.

$Q = \exists, A \vee B$
$Q = \forall, A \wedge B$

recursive call can re-use the same space
TQBF belongs to PSPACE
TQBF is one of the hardest problem of PSPACE

Formula Game Problem
E wants to make formula TRUE
A wants to make formula FALSE

a winning strategy?
Quantified Boolean Formula가 주어졌을 때,
player E want to make ϕ TRUE
만약에 E가 어떤 선택을 함으로써 항상 그 formula를 TRUE로 만들 수 있다면,
그것이 E의 winning strategy이다.

(1) We have a oracle that decide TQBF in one step
(2) to decide any language L that belongs to P or NP,
we can use X freely.
In other words, to decide L, a TM can call X
if that can help deciding L.
X replies to the call with a correct answer immediately

TQBF belongs to PSPACE,
PSPACE = NPSPACE
TQBF belongs to NPSPACE

TQBF is one of the hardest problems in PSPACE
-> every problems in PSPACE can be efficiently convertible to TQBF.
No Hamiltonian path problem
Input : A graph G
Question : is it true that G has no Hamiltonian path?

-> coNP

P와 NP의 관계
NP is the class of problems A of the following from:

x is a yes-instance of A if and only if there exists a certificate ‘w’
such that (x, w) is a yes-instance of B
where B is a decision problem in P
regarding pairs (x, w), and where |w| = poly(|x|)

always any problems belongs to NP can be decided in polynomial time
by TM

NP is the class of properties A of the form

$$A(x) = \exists_w : B(x, w)$$

where B is in P, and where |w| = poly(|x|)

if P = NP, then the entire hierarchy collapses to P
(-> PH = P)

if SAT belongs to P,
then a certificate of an arbitrary YES instance of SAT can be found efficiently
-> decision problem과 search problem의 관계가 무엇인가?

-> decision problem을 효율적으로 solve하는 것이 가능하다면,
search problem을 효율적으로 solve하는 것이 가능하겠는가? 에 대한 질문이다.

P는 polynomial decider를 가지고 있는 decision problem의 집합이다.
우리가 polynomial length의 certificate를 가지고 있다면,
그것을 efficiently하게 verify가능하다

f can be converted into 2 sub-formulas with OR.
if f is a yes instance, then either one of 2 sub-formulas should be satisfiable
Yes instance인지를 확인하면, 2개로 쪼개서 또 확인해보는 것이다.
2n+1만큼의 evaluations이 필요하다.

#P
a set of counting problems, a set of functions

#P is the class of functions A(x) of the form

$$A(x) = |\{w: B(x, w)\}|$$

where B(x, w) is a property that can be checked in polynomial time,
and where |w|=poly(|x|) for all w such that B(x, w) holds

Proof Checking VS Short Proof

Proof Checking
input : a statement S and a proof P
Question : is P a valid proof of S?
belongs to P

Short Proof
input : a statement S and an integer n given in unary
Question : Does S have a proof of length n or less
belongs to NP

P, NP 문제와 관련이 있다.

Elegant Theory
Input : a set E of experimental data and an integer n given in unary
Question : Is there a theory T of length n or less that explains E?
Elegant Theory belongs to NP

verification is done efficiently

Graph Isomorphism
2개의 undirected graph의 structural equivalence를 생각하는 것이다.
이것은 NP에 속하지만 NP-Complete에도 속하는지는 모른다.

NP is a subset of PSPACE
proof)
let L in NP.
L can be decided in polynomial amount of space
problem의 size가 n이라고 할 때, witness는 p(n)의 크기이고
(p(n) is some polynomial in n)
$2^{p(n)}$개의 witness를 일일이 test해야한다.
다만 working space를 재활용하면 polynomial amount of space를 사용할 수 있음으로
NP는 PSPACE에 속한다.
-reducibility-
polynomial time reduction, many-one reduction:

$$A \le_P B:\ w \in A \leftrightarrow f(w) \in B$$

if $A \le_P B$ and $B \in P$, then $A \in P$.
proof)
Let M be polynomial time algorithm deciding B.
Let f be polynomial time reduction from A to B.
then, there exists polynomial algorithm N deciding A as follows,
N: On input w,
 (1) Compute f(w)
 (2) Run M on input f(w) and outputM outputs.

B is NP-Complete if
(1) B is in NP
(2) every A in NP is polynomial time reducible to B.

if B is NP-Complete, and $B \le_P C$ for C in NP,

then C is NP-Complete

3SAT = 3CNF
(1) literal : boolean variable x or ~x
(2) clause : literals with OR
(3) formula : clauses with AND
(4) if each clause consists of 3 literals, and all clauses are combined with ANDs

-> then, that formula is 3CNF

SAT is in NPC
if SAT is in P, then P = NP

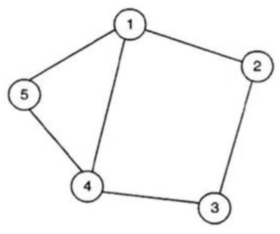
SAT can be reduced to 3 SAT
(1) clause의 literal이 1개인 경우, c=l
$c' = (l \vee l' \vee l'') \wedge (l \vee \sim l' \vee l'') \wedge (l \vee l' \vee \sim l'') \wedge (l \vee \sim l' \vee \sim l'')$
new variable l`, l``
(2) a clause c with two literals
$c = (l_1 \vee l_2)$
$c' = (l_1 \vee l_2 \vee l') \wedge (l_1 \vee l_2 \vee \sim l')$
(3) a clause with more than 4 literals
c = $(l_1 \vee l_2 \vee \cdots \vee l_n)$

$$c' = (l_1 \vee l_2 \vee V_1) \wedge (l_3 \vee \sim V_1 \vee V_2) \wedge (l_4 \vee \sim V_2 \vee V_3) \wedge \cdots \wedge (l_{n-2} \vee \sim V_{n-4} \vee V_{n-3}) \wedge (l_{n-1} \vee l_n \vee \sim V_{n-3})$$

Graph isomorphism
one-to-one function $\phi : V \rightarrow V$
for a edge (vi, vj) in G, a (φ(vi), φ(vj)) in G`.
this belongs to NP
it is a special case of permutation (matching과 관련있음)
V(G, H, c) verifies that c is really an isomorphism between G and H
1. verify c us a permutataion of nodes in G
2. permute node of G as given in c: check whether the resulting graph is H

subgraph isomorphism
for given 2 graph G and H,
determine whether G contains a subgraph that is isomorphic to H
this belongs to NPC

vertex cover
instance : a graph G = (V, E) and a positive integer $k \le \|V\|$
question : is there a vertex cover of size less or equal than k in G?



{1,3,5} – a vertex cover
{1,2,4} – a vertex cover
{2,5} – not a vertex cover

promise problem
instance x
promise P(x)
question Q(x)

P, Q : deciable predicates
(predicates are statements that contain variables
if we change that variables to constants, then predicates become propositions)
dTM M halts on every input solves (P, Q) if

$$\forall x[P(x) \rightarrow [Q(x) \leftrightarrow M(x) = \text{"}yes\text{"}]]$$

decision problem is a special case of promise problem
decision problem is promise problem whose promise is universal
-> holds for every problem instance

Given a directed acyclic graph, determine if the graph has a path of length 10.
yes-instance : directed acyclic graph with 10 length path.
no-instance : directed acyclic graph with no 10 length path.
promise : set of directed acyclic graph

Unique - SAT
instance : a formula F of propositional logic
promise : F has either 0 or 1 satisfying assignments
question : Does F have a satisfying assignment?

Problems (so far)
- Perfect matching, stable matching
- Hamiltonian cycle problem, Euler cycle problem
- TAUTOLOGY
- Protein folding problem
- Dominating set, partition into triangles
- the complement of PATH, the complement of SAT
- No Hamiltonian Path
- Proof checking, Short proof, Elegant theory
- Graph isomorphism, subgraph isomorphism
- Vertex cover
- Unique-SAT
- Promise problem

Definitions/terms
- augmenting path
- coP, coNP, #P, PH
- oracle, oracle Turing machine
- P^A, NP^A, P^{SAT}
- Random oracle hypethesis

Theorems / lemmas / facts
- Characterization of a maximum matching using an augmenting path
- P is a subset of the intersection of NP and coNP
- If P = NP then NP = CONP = P
- P = intersection of NP and coNP? (open)
- TAUTOLOGY is a member of coNP
- NP is a subset of sharp P
- if P = NP then the entire polynomial hierarchy collapses to P
- Proof checking belongs to P
- Short proof belongs to NP
- Elegant theory belongs to NP
- NP is a subset of PSPACE
- GRAPH ISOMORPHISM NP-Complete? (open)
- SUBGRAPH ISOMORPHISM is NP-complete!

- **oracle and relativization**

oracle anwsers questions immediately

M^A: oracle Turing Machine for language A

P^A: the class of languages decidable
with a polynomial time oracle Turing machine that uses oracle A.

P^{SAT} : whether SAT formula is satisfiable of not.

SAT is in NPC, so $P^{SAT} = P^{NP}$.

$\overline{SAT}$: the language of unsatisfiable formula.

$\overline{SAT} \in P^{SAT}$(왜냐하면 P^{SAT}의 답의 반대가 $\overline{SAT}$이기 때문이다.)

P^A이라면?

(1) any instance of A를 single step안에 solve하는 new command를 가지는 것이다.

(2) there is a program that solves A in polynomial time
with just ordinary commands

world?

Physics : theory -> calculation -> predictions for observations

Mathematics : axioms -> reasoning -> theorems

Computing : program -> execution on computer -> output

programmable world VS not programmable world

random oracle hypothesis

hypothesis : States believed to be true

oracle에 random 요소를 부여한다.

oracle A is created in a way that probability that x belongs to A is 1/2.

Pr[$P^A = NP^A$] = 0, Pr[$P^A \neq NP^A$] = 1

oracle이 없을 때? $P \neq NP$이라는 것이다. :사실 이는 false로 들어났다.

HW #3
Is there a binary relation on the set of computational problems mentioned in this class? If the answer is YES, give a concrete relation. Otherwise, explain the reason why there CANNOT exist a relation on this set.

{Traveling salesman problem, Partition problem, Shortest paths problem, K-clique problem, PATH, SAT, TQBF, Formula game problem, Matching problem - maximum matching problem, Independent set problem, Perfect matching, stable matching, Hamiltonian cycle problem, Euler cycle problem, TAUTOLOGY, Protein folding problem, Dominating set, partition into triangles, the complement of PATH, the complement of SAT, No Hamiltonian Path, Proof checking, Short proof, Elegant theory, Graph isomorphism, subgraph isomorphism, Vertex cover, Unique-SAT}

pairs with complements
(vertex cover, independent set)
(proof checking, short proof)
can be a relation

theory is a set
true proposition
true/false를 말하려면 일종의 domain/model/interpretation이 필요하다.

only last one is a sentence

1. $R_1(x_1) \wedge R_2(x_1, x_2, x_3)$
2. $\forall x_1 \left[R_1(x_1) \wedge R_2(x_1, x_2, x_3) \right]$
3. $\forall x_1 \exists x_2 \exists x_3 \left[R_1(x_1) \wedge R_2(x_1, x_2, x_3) \right]$.

logical consequence 1 - a relation between:
U : set of propositions
A : a formula
A is a logical consequence of U, denoted $U \models A$
iff every model of U is a model of A.

logical consequence 2 - a closure property:
theory is set of propositions,
formulas that is closed under a logical consequence(operation)

BPP (Bounded error Probabilistic Polynomial time)
BPP can be defined using only dTM.
language L is in BPP iff there exists polynomial p and dTM M such that
- M runs for polynomial time on all inputs
- For all x in L, the fraction of string y of length p(|x|) which satisfy
  M(x, y) = 1 is greater than or equal to 2/3
- For all x not in L, the fraction of strings y of length p(|x|) which satisfy
  M(x, y) = 1 is less than or equal to 1/3

BPP can be defined in PTM(Probabilistic TM)
(PTM is a special case of nTM)

M halts in T(|x|) steps
Pr[M(x) = L(x)] ≥ 2/3, where we denote L(x)=1 if x∈L/L(x)=0 if x∉L
BPTIME(T(n)) : class of languages decided by PTMs in O(T(n)) time
BPP = $\bigcup_c BPTIME(n^c)$
relation on set B : subset of BXB
ex) B = {a, b, c}
BXB = {(a,a), (a,b), (a,c), (b,a), (b,b), (b,c), (c,a), (c,b), (c,c)}
subset of BXB : {(a,b), (b,c)}

PTM can end with 3 states
accepting, rejecting, undetermined

Reducibility:
mapping reducibility = many-one reducibility = Karp reducibility

if A is mapping reducible to B, and B is acceptable by TM
then A is acceptable by TM

if A is mapping reducible to B, and A is not acceptable by TM
then B is not acceptable by TM

Turing reducibility is more general than mapping reducibility

if $A \leq_T B$, and B is decidable, then A is decidable
if A is mapping reducible to B, then A is Turing reducible to B.
proof) if A is mapping reducible to B,

$$w \in A \leftrightarrow f(w) \in B$$

f is called the polynomial time reduction of A to B.

mapping reduction can be used by oracle for B,
so that A can be decided relative to B

$$A \leq_m^p B \Rightarrow A \leq_{k\text{-}tt}^p B \Rightarrow A \leq_{k+1\text{-}tt}^p B \Rightarrow A \leq_{tt}^p B \Rightarrow A \leq_T^p B.$$

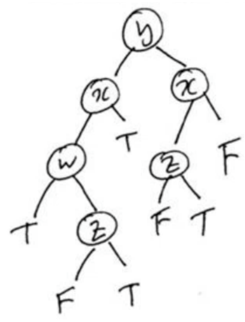
independent set, vertex cover, clique는 어떠한 관계가 있다.

Boolean query
A is truth-table reducible to B if
(1) there is computable function f such that
(2) f(n) is a Boolean query to B
    that consists of a finite # of membership questions against B
(3) n belongs to A <-> f(n) belongs to B

if P = NP, then SAT is truth-table reducible to a P-selective set

Any language L in P is P-selective

binary decision tree for a boolean function



$f = (\sim x \wedge y) \vee$
$(x \wedge \sim z) \vee$
$(y \wedge w)$

a path y, x, w - y:T, x:T, w:T then f:T
a path y, x, z – y:F, x:T, z:T then f:F

show coNP.
this belongs to BPP

problem:
input : 2 binary decision tree A, B
output : if A and B represent the same boolean function, then YES
          otherwise NO.
this problem belongs to coNP

if there exists a certificate that make NO, that is easily verified.
This problem belongs to BPP

NP is a subset of P
counting problem is in #P
if it asks for the number of certificate of an NP problem

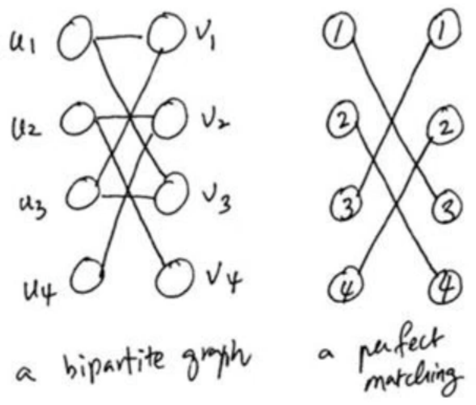
if L belongs to NP, then
1. for all x in L, there exists a short certificate c such that
2. polynomial time verifier V returns yes
   when (x, c) is given as input
therefore, for L, the existence of such a certificate is guaranteed

#P is a subset of PSPACE
if a problem belongs to #P, then
there are certain number of short certificate for corresponding NP problem
re-use polynomial space one by one

any language L in P is P-selective
proof)
if L is a member of P then, there is a polynomial decider M that decides L
therefore, we use M to define a p-selector for L

1. if there is an NP Complete P-selective set, then P = NP
2. if each member of NP is P-selective, then P = NP

perfect matching in a bipartite graph : a permutation



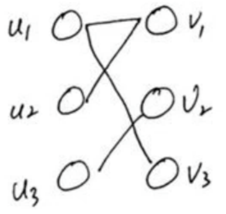
$1 \to 3$ $\sigma(1)=3$
$2 \to 4$ $\sigma(2)=4$
$3 \to 1$ $\sigma(3)=1$
$4 \to 2$ $\sigma(4)=2$

σ : a permutation of $\{1,2,3,4\}$

problem of counting the total number of perfect matchings in a bipartite graph is a member of #P

Tutte matrix of a bipartite graph nxn matrix M such that

M(i, j) = $\begin{cases} x_{ij} \text{ if } (u_i, v_j) \in E \\ 0\, otherwise \end{cases}$



$$\begin{bmatrix} x_{11} & 0 & x_{13} \\ x_{21} & 0 & 0 \\ 0 & x_{32} & 0 \end{bmatrix}$$

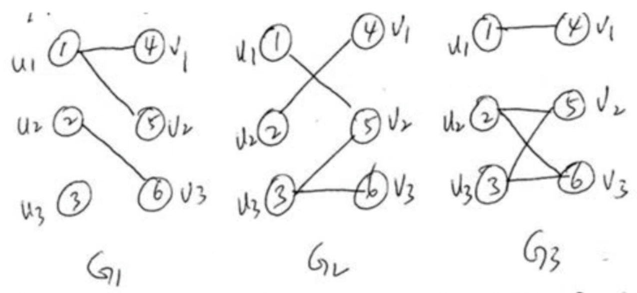
if the determinant of the Tutte matrix G is not zero,
then a bipartite graph G has a perfect matching

$$\det(A) = \sum_{\sigma \in S_n} sign(\sigma) \prod_{i=1}^{n} A_i \sigma(i)$$

sign(σ) : number of pairs (i, j) such that i<j, but j precedes I

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

σ			sign(σ)		
1	2	3	even	+	
1	3	2	odd	−	3, 2
2	1	3	odd	−	2, 1
2	3	1	even	+	2, 1 3, 1
3	1	2	even	+	3, 1 3, 2
3	2	1	odd	−	3, 2 2, 1 3, 1



$$\begin{bmatrix} x_{11} & x_{12} & 0 \\ 0 & 0 & x_{23} \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & x_{12} & 0 \\ x_{21} & 0 & 0 \\ 0 & x_{32} & x_{33} \end{bmatrix} \quad \begin{bmatrix} x_{11} & 0 & 0 \\ 0 & x_{22} & x_{23} \\ 0 & x_{32} & x_{33} \end{bmatrix}$$

$\det M_{G_1} = 0$ $\quad$ $\det M_{G_2} = - x_{12} x_{21} x_{33}$ $\quad$ $\det M_{G_3} = x_{11} x_{22} x_{33} - x_{11} x_{23} x_{32}$

HW #1
if SAT belong to P,
then a certificate of an arbitrary YES instance of SAT can be found efficiently.

SAT : decision problem, language that all satisfyable boolean formulas

f can be converted into 2 sub-formulas combined with OR operation

because SAT belongs to P,
there exists a dTM M that decides SAT in polynomial tme.

1. use M to determine whether f belongs to SAT or not.
2. if f belongs to SAT,
   convert it into a formula that consists of 2 sub-formulas combined with OR
3. use M to determine any one of these 2 sub-formulas belongs to SAT,
   let`s call the sub-formula belongs to SAT “f1“

repeat the same procedure for f1 until we know truth values for all variables

Each formula is rewritten as two formulas combined by OR
Each of 2 sub-formulas needs to be evaluated by M
Total number of evaluation : 2n+1
Therefore it is possible to find a certificate in polynomial time

HW #2
Explain the reason why P = NP if following is true:
(1) We have a magical device called X that can decide TQBF in one step
(2) To decide any language L that belongs to P or NP, we can use X freely
To decide L, a TM can call X if that can help deciding L.
X replies to the call with a correct answer immediately.

TQBF is one of the hardest problems in PSPACE in that
every problem in PSPACE can be efficiently convertible to TQBF

->
proof $P^{TQBF} = NP^{TQBF}$!
TQBF ⊆ PSPACE
PSPACE = NPSPACE

proof)
1. $P^{TQBF} \subseteq NP^{TQBF}$: trivial (because P is subset of NP)
2. $NP^{TQBF} \subseteq P^{TQBF}$

Given a language L ∈ NP^{TQBF}: there are two possibilities
(1) L ∈ NP (L ∈ NPSPACE, because NP ⊆ NPSPACE)
(2) an oracle X that decides TQBF is used (L is in TQBF)
: this can be simulated by nTM that uses a polynomial amount of space
the same space can be re-used for a next query
therefore, in both cases, L is a member of NPSPACE

NP^{TQBF} ⊆ NPSPACE = PSPACE ⊆ P^{TQBF}
(∵any language in PSPACE can be reduced to TQBF in polynomial time)