OneClass find more resources at www.oneclass.com

Nov 13 Lec 27

So far there's been
- problems solvable by simple algorithms (greedy)
- problems efficiently solvable by complex algorithms (network flow, DP, LP)
- problems not solvable efficiently (polytime) by any known algo.

WANT: a general method to prove some problems have no efficient algo.

INPUT SIZE + RUNNING TIME

Input size = amount of space used to store it
bitsize for integers —— each int x needs $\lfloor \log(x+1) \rfloor$ bits

- ex. int array A of n elements, $size(A) \leq n \cdot \log(\max(a_1, \ldots, a_n))$
  assume there's a max # bits k $size(A) \leq n \cdot \log(k) \in O(n)$

- ex. a graph stored as a list of m edges for n vertices
  assume vertices are labeled from 1, ..., n
  $\Rightarrow size(v_i) \in O(\log n)$
  assume edge is a vertex pair
  $\Rightarrow size(e_i) \leq 2\log(n)$
  Then $size(V) = n \cdot \log n$, $size(E) = 2m \cdot \log n$ } $Size(G) \in O((n+m)\log n) \in (n^2 \log n)$

- ex. an adj matrix of a graph G with n vertices
  $Size(G) = O(n^2)$

any polynomial algorithm as a function of inputed size is also polytime
if uses only reasonable primitive operations
comparisons, additions, subtractions
Not reasonable operations b/c can grow
multiplication, exponentiation

ex. suppose input is int n, $size(n) = \log n \Rightarrow n \in O(2^{size})$
algorithm loops from 1, ..., n $\Rightarrow$ 1, ..., 2^{size}
then the algorithm does not run polytime wrt input size

When input is n vertices, m edges then $size = n^2 \log n$
$\Rightarrow n \in O(size)$
then looping over n would remain in polytime wrt input size

CATEGORISING PROBLEMS — for a problem X
Search Problem —— find a solution that satisfies X
Optimization Problem —— find a optimal (min/max) solution that satisfies X
Decision Problem —— determine if any solution for X exists
output is "Yes" or "No"
- ex. Given a graph G, and vertex s,t is there a path $s \to t$
- use decision problems to prove negative results
  ie. the problem has no efficient solution
- show a decision hard ($\nexists$ efficient algorithm) then the general problem is also hard

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Nov 15 — Lec 28

Complexity Theory — show ∄ a good algorithm for a problem
⇒ if ∄ a good algorithm then no one can solve the problem efficiently (in polynomial time)
note: these are not efficient polytime: n^{1000} or 10^{1000n}

result — 1. can use the limitations to build new things
ex. in crypto encryption work algo unsolvable

2. if known some problem does have an efficient algorithm then will not waste time finding one

To prove a problem has no efficient algorithm
1. prove a lowerbound on any solution (difficult)
2. through reduction — there's a set of problems where ∄ efficient algorithm for them
an efficient solution to one is a solution to all

P = class of all decision problems with polytime solutions
-ex. maxflow, matching, MST

NP = class of decision problems with polytime verifier
Problem Q can be verified in polynomial time if ∃ algorithm V, s.t.
1. for all instances of input x to Q returns Yes,
∃ certificate c such that $V(x,c)$ = Yes in polytime

2. for all instances of input x to Q returns No,
∀ c such that $V(x,c)$ = No

$Q(x)$ = Yes $\iff$ $\exists c \in$ polysize, $V(x,c)$ = Yes in polytime

Problems ∈ NP

Composite Decision Problem ∈ NP
INPUT — Integer $x \in \mathbb{N}$, $n = size(x) = \log(x)$
OUTPUT — Yes if x has a factor; No otherwise
Proof — let certificate $c = (c_1, c_2)$ s.t. $c_1 \cdot c_2 = x$
Then $size(c) = size(c_1) + size(c_2)$
$\leq 2 \cdot size(x) = 2 \cdot n$
$\Rightarrow size(c) \in O(n)$ (in polysize)
Let $V(x,c)$: if $c_1 \cdot c_2 = x$ & $1 < c_1, c_2 < n$:
return Yes
return No
$V \in O(1)$ (in poly time)

To prove correctness, must show
if x is a composite number then there is some certificates accepted by the verifier
if x is not a composite number then there are no certificates

Note: the follow V is not a verifier in polytime
$V(x, c)$: for $i = 2, \ldots, x-1$; do
if $x \% i == 0$; then
return Yes
return No
$n = \log x \Rightarrow x = 2^n$ then loop $1, \ldots, 2^n$ would be in exponential time wrt size ⇒ V not polytime

Integer Program Feasibility Problem
INPUT — constraints: $A\vec{x} \leq \vec{b}$
OUTPUT — is there an $\vec{x} \in \{0,1\}$ s.t. if $A\vec{x} \leq \vec{b}$
Certificate = feasible $\vec{x}$; V = check $A\vec{x} \leq \vec{b}$

Factoring Decision Problem
INPUT — $x, k \in \mathbb{N}$
OUTPUT — Does x have a nontrivial factor $\leq k$
Proof — let certificate c be a list of prime numbers, s.t
$1 < c_1, c_2, \ldots, c_m < x$ and
$c_1 \cdot c_2 \cdots c_m = x$
Since the product of all numbers is x then $c \in$ polysize
let V: check if they're prime $\in$ polytime
check if they're $\leq n$
check if they're product is x
check if any is $\leq k$

requirements for NP
1. polytime verifier algorithm } w.r.t input size
2. polysize certifiate

Theorem: any problem P is also in NP
proof: let A be a polytime algorithm for the problem P
let certificate $c = \emptyset$
let verifier be $V(x, c)$: if $A(x)$ = Yes
return Yes
return No

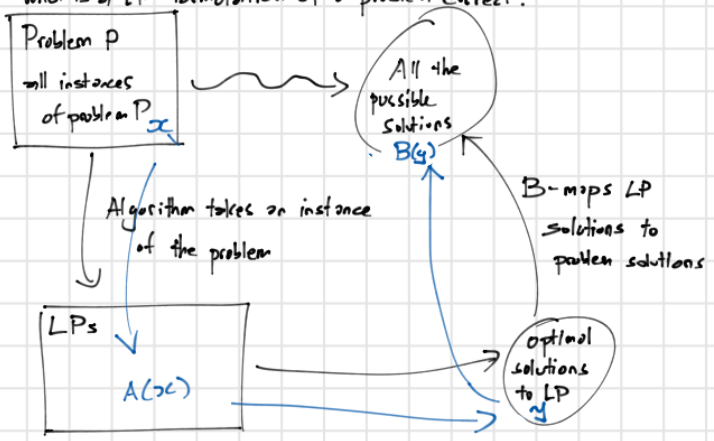
P is a subset of NP (as shown above)
Not known if P = NP, likely false

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Proving LP

when is an LP formulation of a problem correct?



For all instances of x of the problem P,
let y be the optimal solution of the LP $A(x)$
Then the algorithm B applied to y produces an optimal solution to x

ex. Fractional Knapsack

Input: A list of items $I = \{I_1, I_2, \ldots, I_n\}$
where for $j = 1, \ldots, n$ the item $I_j = (v_j, w_j) \in \mathbb{R}^2$
Also a knapsack bound $B \in \mathbb{R}$

Problem: find a fraction p_j (with $0 \leq p_j \leq 1$)
such that $\sum_{j=1}^{n} p_j w_j \leq B$ and $\sum_{j=1}^{n} v_j p_j$

Fractional knapsack $\longrightarrow$ LP

obj: $\max \sum_{j=1}^{n} b_j \cdot v_j$

subject to: $\sum_{j=1}^{n} b_j \cdot w_j \leq B$

$0 \leq b_j \leq 1, \quad \forall j = 1, \ldots, n$

var: b_j fraction of item j taken

$b_j \mapsto p_j$

Suppose $y = b_1', b_2', \ldots, b_n'$ is an optimal solution
Applying B to y you need to get $p_1', p_2', \ldots, p_n'$
These satisfy the constraints
And optimality follows from the optimality of simplex

How does this relate to complexity theory

P — all problems solvable by poly time algorithms
NP — all problems which have "certificates" that can be verified in poly time

Co-NP —

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Integer Programming:

Integer programming: more restricted version where all constants and variables are integers. NP-complete (no efficient algorithm).
Example: Minimum Vertex Cover: Given an undirected graph $G = (V, E)$, Identify a subset of vertices C that *covers* every edge (i.e., each edge has at least one endpoint in C), with minimum size.

We represent this problem as an integer program: use variable x_i for each vertex $v_i \in V$
minimize: $x_1 + x_2 + ... + x_n$
subject to:

- $x_i + x_j \geqslant 1$ for all $(v_i, v_j) \in E$
- $x_i \in \{0, 1\}$ for all $v_i \in V$

This 0-1 integer program is completely equivalent to original problem, through correspondence: v_i in cover iff $x_i = 1$. In more detail:

- Any vertex cover C yields feasible solution $x_i = 1$ if $v_i \in C$, 0 if $v_i \notin C$ because each constraint $x_i + x_j \geqslant 1$ satisfied (C must include one endpoint of each edge).
- Any feasible solution to LP yields vertex cover $C = \{v_i \in V : x_i = 1\}$ because for each edge (v_i, v_j), constraint $x_i + x_j \geqslant 1$ ensures C contains at least one of v_i, v_j.

Unfortunately, Integer Programming (IP) is NP-hard, so the problem cannot be solved in polytime this way. In the next lecture we will propose an algorithm to approximate the solution.

Linear relaxation method: remove restriction of x_i to integer values
minimize: $x_1 + x_2 + ... + x_n$ subject to:

- $x_i + x_j \geqslant 1$ for all $(v_i, v_j) \in E$
- $0 \leqslant x_i \leqslant 1$ for all $v_i \in V$

Solution can be found in polytime, but may include fractional values of x_i's.
Example: $G = (V, E)$ with $V = \{v_1, v_2, v_3\}$, $E = \{(v_1, v_2), (v_2, v_3), (v_1, v_3)\}$ becomes
minimize: $x_1 + x_2 + x_3$
subject to:

- $x_1 + x_2 \geqslant 1$
- $x_2 + x_3 \geqslant 1$
- $x_1 + x_3 \geqslant 1$
- $0 \leqslant x_1, x_2, x_3 \leqslant 1$

with solution $x_1 = x_2 = x_3 = 1/2$.

Then how to solve this problem? Suppose the solution to the relaxed LP is x_i^* for each variable x_i. Create cover as follows: for each $v_i \in V$, put $v_i \in C$ iff $x_i^* \geqslant 1/2$. (C is a cover because constraint $x_i + x_j \geqslant 1$ guarantees at least one of x_i^*, and x_j^* is $\geqslant 1/2$ for each edge (v_i, v_j).)
Approximation ratio?
Consider minimum vertex cover C'. For $i = 1, ..., n$, let $x_i' = 1$ if v_i in C'; $x_i' = 0$ otherwise. $x_1', ..., x_n'$ is a

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

solution to linear program that satisfies all constraints with 0-1 values so $|C'| = \sum x'_i \geqslant \sum x^*_i$ where x^*_i is optimal solution to linear program with no restriction on values, so guaranteed to be at least as small as any other solution, including those with additional restrictions. For $i = 1, ..., n$, let $\hat{x}_i = 1$ if $x^*_i \geqslant 1/2$; $\hat{x}_i = 0$ otherwise. Then, for each i, $\hat{x}_i \leqslant 2x^*_i$ so $|C| = \sum \hat{x}_i \leqslant 2 \sum x^*_i \leqslant 2|C'|$ (by equation above) Hence, $|C|$ is no more than twice the size of a minimum vertex cover.

Q: In general, how can we compute ratio without knowing OPT?
A: Use a lower bound. Find another value LB that's easy to compute and for which you can prove $LB \leqslant OPT$ and $A \leqslant r \times LB$, as in the above example for vertex cover.

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Nov 18 — LEC 29

NP: polytime certificates
polytime verifier
if the answer to the problem is Yes
there is some certificate for it
if the answer to the problem is No
there are no certificates / proofs

ex. 0/1 knapsack problem
given a list of objects with size $w_1, \dots, w_n$ and values $v_1, \dots, v_n$
and the size of a knapsack m and a number k
can we select items such that their total size $\le m$ and
and their total values $\ge k$

certificates: a list of items I → $len(I) = n$
verifier: $V(x, I)$ → $[w_1, \dots, w_n], [v_1, \dots, v_n], m, k$
x is the input for the original problem
if $\sum_{i \in I} w_i \le m$ && $\sum_{i \in I} v_i \ge k$ then
Yes
else No

ex. longest Path problem: is there a simple path of total weight atleast k?

ex. Hamiltonian path problem: given a graph G, is there a simple path using all vertices?

ex. Hamiltonian cycle problem: given a graph G, is there a simple path using all vertices and end at the first vertex of the path

ex. SAT: satisfiability
input: a propositional formula $\varphi(\vec{x})$
ex. $(x_1 \vee x_2 \Rightarrow x_3) \wedge \neg x_4 \wedge (x_4 \Rightarrow x_1)$ (T, T, F, T)
question: is there a truth assignment to the propositional variables $\vec{x}$ which makes $\varphi(\vec{x})$ true?

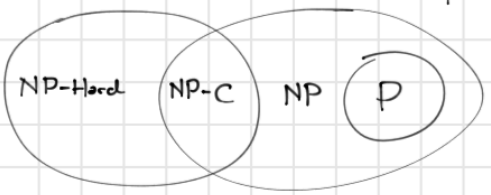
Nov 20 — LEC 30

NP Hard = the class of decision problems that all NP problems are reducible to them in polynomial time
NP Hard problems do not have to be in NP
ex. Halting Problem: given Program P and Input x
will P(x) halt?

$Q \in \text{NP-Hard} \Rightarrow \forall Q' \in NP,\ Q' \le_T^P Q$ (don't need to preserve NP)

NP Complete = the class of decision problems in NP that all NP problems are reducible to them in polynomial time

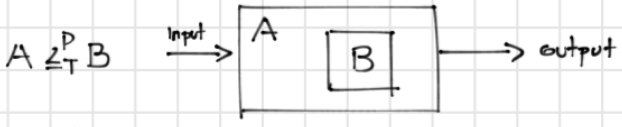
$Q \in \text{NP-C} \Rightarrow Q \in NP \wedge \forall Q' \in NP,\ Q' \le_m^P Q$ (need to preserve NP)



Polynomial time reductions → a technique to formalize the notion that one problem is not "harder" than another

Polytime Blackbox (Turning) reductions
use a blackbox however you want, but must be in polytime
ex. Problem A, B use a black box (B) to solve A

$A \le_T^P B$



Turning reductions don't preserve NP-ness

Polytime Many-to-one (Karp) Reduction
given Problem A, B,
$A \le_m^P B \Rightarrow \exists R \in \text{polytime},\ \forall \text{input } x,\ A(x) = Yes \iff B(R(x)) = Yes$

Karp reductions do preserve NP-ness
Theorem: if $B \in NP \wedge A \le_m^P B \Rightarrow A \in NP$
proof — $B \in NP \Rightarrow$ B has a polynomial verifier V_B
$B(y) = Yes \iff \exists c, V_B(y, c) = Yes$
$A \le_m^P B \Rightarrow$ there is a polytime reduction R such that $A(x) = Yes \iff B(R(x)) = Yes$
$V_A(x, c) = V_B(R(x), c)$

Cor: $B \in P \wedge A \le_m^P B \Rightarrow A \in P$

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

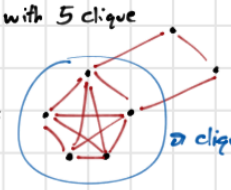
ex. k-Clique $\leq_m^P$ k-Independent Set

k-Clique Problem:

Input — Graph G, integer $k \in \mathbb{N}$

Output — Yes iff G has a clique of size $\geq k$

ex. G with 5 clique

G = 

a clique - all v are connected

k-Clique $\in$ NP

let certificate c be a list of vertices in the clique

```
V(x, c): for v∈c; do
           for u∈c−{v}; do
             if uv ∉ E; then
               return No
         return Yes
```

polytime: $V \in O(c^2) \in O(k^2 m)$

polysize: $c \in O(V)$

k-independent set

Input — Graph G, integer $k \in \mathbb{N}$

Output — Yes iff G has k indepent vertices

ex. G with 3 independent set

G = 

Independent set - no edges btwn them

k-independentset $\in$ NP (andyous proof)

Reduction Idea k-clique $\leq_m^P$ k-independent set

let $x = G, k$ be input to k clique

$R(x) = y = (H, \ell)$

$\ell = k$, $V_H = V_G$

for any vertex pair $u, v \in V_G$ add an edge uv to E_H if $uv \notin E_G$, else don't add edge uv to E_H

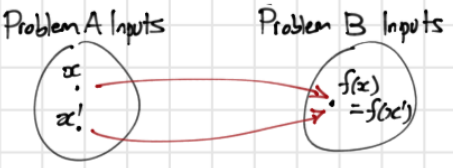
Nov22 — LEC 31

Recall — Cook Reduction (polytime black box)

Karp Reduction (polytime many-to-one)

$A \leq_m^P B \quad R: \Sigma_A^* \to \Sigma_B^*$

$A(x) = Yes \iff B(R(x)) = Yes$



NP is closed under many-to-one reductions

$A \leq_m^P B \wedge B \in NP \Rightarrow A \in NP$

Subset Sum

Input — a list of numbers $a_1, \ldots, a_n$

an integer t

Output — Yes iff. $\exists I \subseteq \{1, \ldots, n\}$ s.t $\sum_{i \in I} a_i = t$

Two-partition

Input — a list of numbers $a_1, \ldots, a_n$

Output — Yes iff it is possible to partition the list into two lists A and B such that

$$\sum_{x \in A} x = \sum_{x \in B} x$$

Show: Subset Sum $\leq_m^t$ two-partition

subsetSum(x) = Yes $\iff$ twoPart(R(x)) = Yes

$x = [a_1, \ldots, a_n], t$ $\quad R(x) = b_1, \ldots, b_m$

IDEA: $c = \sum_{i=1,\ldots,n} a_i$

$2t = a_1 + \cdots + a_n + (2t - c)$

then $\{a_1, a_3, a_4\} = t$ and $\{a_2, a_5, 2t - c\} = t$

$R(a_1, \ldots, a_n, t) = (a_1, \ldots, a_n, 2t - \sum_{1,\ldots,n} a_i)$

R can be calculated in polynomial time

SubsetSum$(a_1, \ldots, a_n, t)$ = Yes $\iff$

2-Partition$(a_1, \ldots, a_n, 2t - \sum_{1 \leq i \leq n} a_i)$ = yes

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Reduction correctness proof

show $SubsetSum(a_1,\dots,a_n,t) = Yes$

$\iff twoPart(a_1,\dots,a_n, 2t - \sum_{1,\dots,n} a_i) = Yes$

($\Rightarrow$) let I be such that $\sum_{i\in I} a_i = t$

let $A = \{a_i \mid i \in I\}$

$B = \{a_i \mid i \notin I\} \cup \{2t - \sum_{1,\dots,n} a_i\}$

$\sum_{x\in A} x = t$ and $\sum_{x\in B} = \sum_{i\notin I} a_i - \sum_{1,\dots,n} a_i + 2t$

$= -\sum_{i\in I} a_i + 2t = -t + 2t$

$= t$

hence $twoPart(R(x)) = Yes$

($\Leftarrow$) let A, B be a 2 part partition of

$a_1,\dots,a_n, 2t - \sum_{1,\dots,n} a_i$

wlog assume $2t - \sum_{1,\dots,n} a_i \in A$ and

let $I = \{i \mid a_i \in B\}$

$\sum_{i\in I} a_i = \sum_{x\in B} x$

$= \dfrac{\sum_{1,\dots,n} a_i + \left(2t - \sum_{1,\dots,n} a_i\right)}{2} = t$

hence $SubsetSum(x) = Yes$

Vertex Covers:

input → a graph size G, a number k

question → does G have vertex cover of size atmost k?

a vertex cover is a list of vertices such that every edge has an end point among them

ex.

show is NP → give a certificate & verifier

Integer Programming

input: a list of variables $\vec{x} \in \mathbb{Z}$

a list of linear constraints $A\vec{x} \le \vec{b}$

question is there an $\vec{x}$ such that $A\vec{x} \le \vec{b}$

hard to show is NP b/c ints for $\vec{x}$ hard to prove in poly size of n×m matrix

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

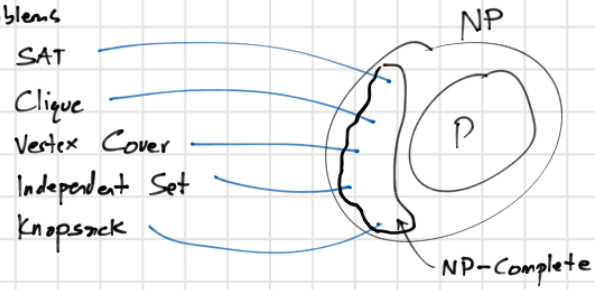
Nov 21 — Test

Definitions

P — all decision problems that can be computed by poly time algorithm

NP — all problems such that there exists a polynomial time verifier V for all instances of the problem there is a certificate x such $V(P, x) = 1$

Problems

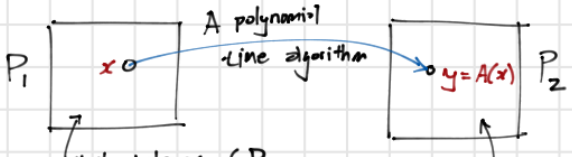


P is in NP b/c V can just solve the problem

A problem is NP-hard if there's a polynomial time Karp reduction from every problem in NP to it

NP-Complete = NP-Hard AND in NP

Karp Reduction $P_1 \leq_p P_2$



all the instances of P_1 -ex. clique problem this would be all graphs with finite $|V|$

all instances of P_2 -ex. Independent Set

Set Cover Input: A set $U = \{x_1, \ldots, x_n\}$

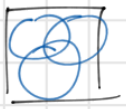
$L = \{S_1, S_2, \ldots, S_m\}$

$\forall i: S_i \subseteq U$, $k \in \mathbb{Z}$

output: 1 if there is a subcollection $C \subseteq L$ such that $|C| \leq k$ and $\bigcup_{S \in C} S = U$

Hitting Set Input: U, L, $k \in \mathbb{Z}$

output: 1 if there is a subset $H \subseteq U$, $|H| \leq k$, s.t $H \cap S_i \neq \emptyset$, for all i

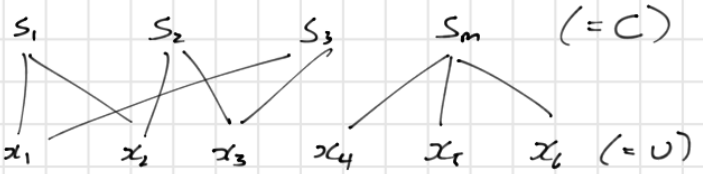


Show Set cover $\leq$ Hitting Set

Sets $S_1, S_2, \ldots, S_m \subseteq U$ — Sets $T_1, T_2, \ldots, T_y$

Universe U — Universe Z

Bound $k \in \mathbb{Z}$



both problems are instance of Covering the edges with $\leq k$

Vertex Cover Problem

Input: undirected graph G and integer k

output 1 if there is a vertex cover of at most k

↳ all the edges are adj.

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Nov 25 Lec 32

Types of Problems — ex. VertexCover

Search — Input: Graph G, integer k
Output: Find a vertex cover of size $\leq k$ or None if not possible

Optimization — Input: Graph G
Output: Find a vertex covers of minimum size

Decision — Input: Graph G, integer k
Output: Yes iff G has a vertex covers of size $\leq k$; No otherwise

Relations:

Decision $\leq_T^P$ Search — run Search, if None, return No else return Yes

Search $\leq_T^P$ Optimization — run Opt, if size $\leq k$, return Yes else return No

$\Rightarrow$ Decision $\leq_T^P$ Optimization

Optimization $\leq_T^P$ Search — for k=1 to n; run Search if answer != None, return answer
return None

Search $\leq_T^P$ Decision — extend like greedy

Define Extension — Input: G, k, partial sol $C = \{v_1, \ldots, v_c\}$
Output: Yes iff $\exists$ a vertex covers extending C

Then Decision $\Longleftrightarrow$ Ext, need to show
Decision $\leq_T^m$ Ext — run Ext(G, c, $\emptyset$)
Ext $\leq_T^m$ Decision (Notes)

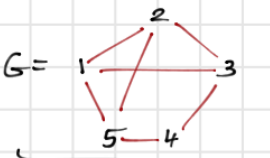
If Ext(G, c, $\emptyset$) return !None; then
S = $\emptyset$
for k=1 to n; do
for v $\in$ V
if Ext(G, k, S $\cup$ {v})
S $\cup$ {v}
return S

$\Rightarrow$ Optimization $\leq_T^P$ Decision

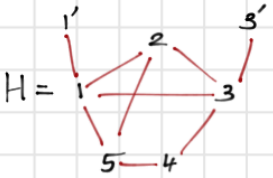
Show Ext $\leq_m^P$ Decision

graph G, C, k —R→ graph H, ℓ

ex. $C = \{1, 3\}$ $k = 3$

G = 

$R(G, C, k)$
$\ell = k$
$V_H = V_G \cup \{v' \mid v \in C\}$
$E_H = E_G \cup \{vv' \mid v \in C\}$

H = 

Ext(G, C, k) = Yes $\Longleftrightarrow$ Decision($\underbrace{R(G, C, k)}_{H, \ell}$) = Yes

($\Rightarrow$) let $S \supseteq C$ be a vertex cover of size $\leq k$ for G. S is a vertex cover for edges of H coming from G.
edge vv' is covered by S because $v \in C \subseteq S$
S is a vertex cover of size $\leq k$ for H

($\Leftarrow$) let S be a vertex cover of size $\leq k$ for H
let S' be S with v' replace with v for all $v \in C$
S' is a vertex covers for G of size $\leq k$
$C \subseteq S'$ because vv' is covered in H either by v or by v'

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Nov 27 — Lec 33

Recall: NP-Hard = the class of decision problems all NP problems are reducible to them in polynomial time

NP Hard problems do not have to be in NP

$Q \in \text{NP-Hard} \Rightarrow \forall Q' \in NP,\ Q' \leq_T^P Q$

don't need to preserve NP

NP Complete = the class of decision problems in NP that all NP problems are reducible to them in polynomial time

$Q \in \text{NP-Complete} \Rightarrow Q \in NP \wedge \forall Q' \in NP,\ Q' \leq_m^P Q$

need to preserve NP

Theorem: $A \in \text{NP-hard} \wedge A \leq_m^P B \Rightarrow B \in \text{NP-hard}$

Proof: Since $A \in \text{NP-hard}$, for all $D \in NP$, $D \leq_P A$

we know $A \leq_m^P B$, the all $D \in NP$, $D \leq_P B$

$\Rightarrow B \in \text{NP-hard}$

Cook's Theorem: CircuitSAT $\in$ NP-Complete

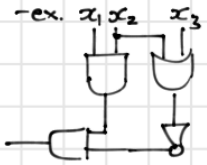
↳ given a boolean circuit, $\exists$ an input x such that the circuit outputs True

1. CircuitSAT $\in$ NP

let certificate c be a set of inputs

V is the circuit. poly time b/c check at most the $*$ gates

$\in O(n^2)$ n = # inputs, n choose 2 gates

ex. 

2. CircuitSAT $\in$ NP-hard

Show $\forall A \in NP,\ A \leq_m^P$ CircuitSAT

$A \in NP \Rightarrow \exists$ a polytime verifier algorithm V_A with certificates of size $p(|x|)$

Turn $V_A(x,c)$ into a polysize circuit

given there's poly # of steps, emulate each step as a gate

For any x, hard code it into the circuit $D_x(\vec{c})$ (set of inputs $x_1, x_2, \ldots$)

D_x is satisfiable iff. $V_A(x,c) =$ Yes for some c

ex. circuit for $i = 1, \ldots, 5$

if $i^2 = 4$; then return Yes

return No

→ if $1^2 = 4$; return Yes

if $2^2 = 4$; return Yes

if $3^2 = 4$; return Yes

⋮

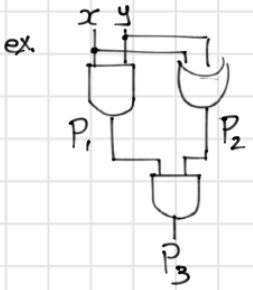
Nov 29 — Lec 34

Show SAT $\in$ NP-Complete

Know CircuitSAT $\in$ NP-C and SAT $\in$ NP

SAT $\leq_m^P$ CircuitSAT $\Rightarrow$ SAT $\in$ NP-C

IDEA: represent a circuit in its boolean form by introducing new variables for intermediate values in the circuit

ex. 

$(P_1 \iff x \wedge y) \wedge (P_2 \iff x \vee y) \wedge (P_3 \iff P_1 \wedge P_2) \wedge P_3$

ex. $x \iff y \wedge z$

$(x \Rightarrow y \wedge z) \wedge (x \Leftarrow y \wedge z)$

$\iff (\neg x \vee (y \wedge z)) \wedge (x \vee \neg y \vee \neg z)$

$\iff (\neg x \vee y) \wedge (\neg x \vee z) \wedge (x \vee \neg y \vee \neg z)$

Also: CircuitSAT $\leq_m^P$ 3SAT, CNF-SAT

3SAT $\leq_m^P$ Clique, Subset Sum

Clique $\leq_m^P$ Vertex Cover

} All $\in$ NP complete

Given Clique $\in$ NP-C, show Vertex Cover $\in$ NP-hard

Clique — Input: graph G, integer k

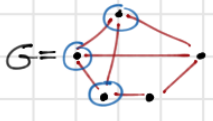
Output: Yes iff there is a set of vertices S of size $\geq k$ such that every vertex pair in S are adjacent. No otherwise.

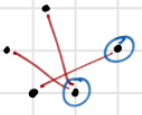
Vertex Cover — Input: graph H, integer l

Output: Yes iff there is a set of vertices S of size $\geq l$ such that for every edge, one of its end points belongs in S

$\Rightarrow$ Show Clique $\leq_m^P$ Vertex Cover

IDEA: if a clique exists, then there's atleast k vertices strongly connected. Those k vertices might be connected to the other $|V|-k$ vertices. In an inverted graph, all edges must have an end pin in the $|V|-k$ vertices

ex. $G =$  $k = 3$

$H =$  $l = 2$

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

$R(G,k) = (H, \ell)$ where $H = \overline{G}$ $\ell = |V| - k$

Prove $Clique(G,k) = Yes \iff VertexCover(R(G,k)) = Yes$

($\Rightarrow$) let S be a clique of size $\geq k$ in G
let $S' = V - S$, S' is a vertex cover of size $\leq |V| - k$ in $\overline{G}$
let vu be an edge in $\overline{G}$ $\Rightarrow$ vu is not an edge in G
$\Rightarrow$ either v or u is not in S
$\Rightarrow$ either v or u is in S
$\Rightarrow$ vu is covered by S'

($\Leftarrow$) let S' be a vertex cover of size $\leq |V| - k$
let $S = V - S'$. S is a clique of size $\geq k$ in G

Show $3SAT \leq_m^p Subset\ Sum$

SubsetSum — Input: $a_1, \ldots, a_n, t$
Output: Yes iff there $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = t$

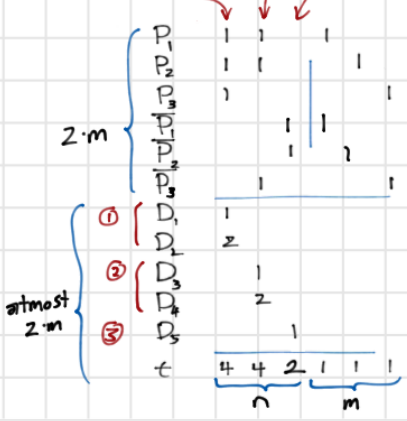
3SAT — Input: formula $F = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \ldots \wedge (a_n \vee b_n \vee c_n)$
where $a_i, b_i, c_i \in \{x_1, \overline{x_1}, \ldots, x_m, \overline{x_m}\}$
Output: Yes iff there $\exists$ a set of input such that F returns True.

IDEA: $a \vee b \vee c \iff x_1 + x_2 + \overline{x_3} \geq 1$ where $x_i \in \{0,1\}$
then $\underbrace{x_1 + x_2 + \overline{x_3} + D_1 + D_2 = 4}_{\text{Subset Sum input}}$ where $D_1 \in \{0,1\}$, $D_2 \in \{0,2\}$

Also need to enforce $x_i \wedge \overline{x_i}$ does not happen

$\underbrace{x_i + \overline{x_i} = 1}_{\text{Subset sum input}}$

ex. $(P_1 \vee P_2 \vee P_3) \wedge (P_1 \vee P_2 \vee \overline{P_3}) \wedge (\overline{P_1} \vee \overline{P_2})$
① ② ③



OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

Nov 28 Tue

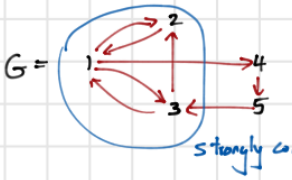
Cook vs. karp reduction
↳ for search problems

karp reduction —— straight encoding

Cook reduction —— use oracle multiple times

ex. k-clique search $\leq_T$ Decision version of k-clique

ex.



tarjan's finds the strongly connected component

cook reduces to BFS calls on all vertices

cook reduction ← need to prove entire algorithm

3 SAT ——

Input: a set of m clauses
$c_1, \ldots, c_m$ over n vars
$x_1, x_2, \ldots, x_n$

Output: 1 if there is an assignment that satisfies all clauses. $\emptyset$ otherwise.

note Conjunctive normal form
CNF = set of clauses

variables: $x_i \in \{0,1\}$ or

clauses: $x_{i_1} \vee \overline{x_{i_2}} \vee x_{i_3} \vee \overline{x_{i_4}}$

negation of var x_{i_4}

$(x_2 \vee x_1 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

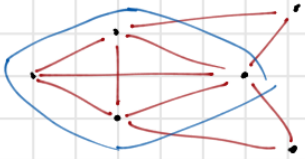
Solution: $x_2=1$, $x_1=X$, $x_2=X$
$x_3=0$, $x_2=1$, $x_1=0$
$x_3=0$, $x_2=0$, $x_1=1$

k-Clique

Input: an undirected graph $G=(V,E)$, an integer $k>0$

Output: 1 if there is a subset of vertices $V' \subseteq V$ with $|V'|=k$ and every pair of vertices $v,v \in V'$ has an edge

4-Clique

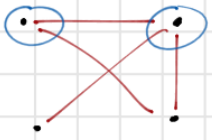


k-vertex cover

Input: an undirected graph $G'=(V',E')$ and an integer $k>0$

output: 1

2-VC

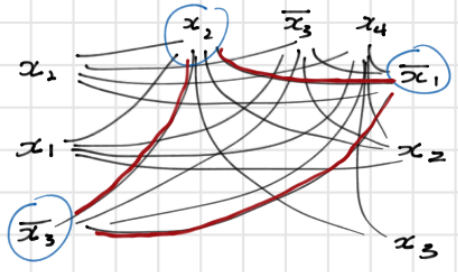


3 SAT $\leq_p$ Clique

given a 3-CNF formula $c_1, c_2, \ldots, c_m$ on variables $x_1, x_2, \ldots, x_n$. We want to construct a graph $G=(V,E)$ and choose k such that $c_1, \ldots, c_m$ are all satisfiable by a single assignment iff. the graph G has a k-clique

satisfying Assignment —— defined as an assignment to each var, or a collection of literals $l_1, l_2, \ldots, l_n$ (one for each clause) such that for no pair of literals l_i, l_j, it will happen that $l_i = \overline{l_j}$

ex. $x_2 \wedge x_1 \wedge \overline{x_3}$ $x_2 \vee \overline{x_3} \vee x_4$ $\overline{x_1} \vee x_2 \vee x_3$



For each clause $C = l_1, l_2, l_3$ add 3 vertices.
Add an edge between two vertices if the literals are consistent and they don't appear in the same clique

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

1. Show that the INDEPENDENT-SET decision problem is NP-hard.

   Input: An undirected graph $G = (V, E)$, positive integer k.

   Question: Does G contain an independent set of size at least k? (Recall that an independent set is a subset of vertices with **no** edge between any two members of this subset.)

   Solution: First, choose decision problem D for reduction. How to choose? Pick problem "close" to IS, if possible, to make reduction easier. We choose VERTEX-COVER (VC) and show VC $\leqslant_p$ IS:

   On input (G, k) (for VC), construct (G', k') (for IS) as follows:
   Set $G' = G$ and $k' = n - k$ (where $n = |V|$ in G).

   Clearly, (G', k') can be computed from (G, k) in polytime. Also, if G contains a vertex cover C of size k or less, then $V - C$ is an independent set in G of size $n - k$ or more: since every edge of G has at least one endpoint in C, no edge has both endpoints in $V - C$. Finally, if G contains an independent set I of size $n - k$ or more, then $V - I$ is a vertex cover of size k or less: since no edge of G has both endpoints in I, every edge of G has at least one endpoint in $V - I$.

2. Show that CLIQUE is NP-hard

   Input: An undirected graph $G = (V, E)$ and a positive integer k.

   Question: Does G contain a *clique* of size at least k, *i.e.*, a subset of k or more vertices such that G contains **every** possible edge between the vertices in the clique?

   Solution: We show that Clique is *NP*-hard by proving IS $\leqslant_p$ Clique (where IS is the IndependentSet problem).

   On input (G, k) (for IS), where $G = (V, E)$, construct (G', k') (for Clique) as follows:
   Set $k' = k$ and $G' = (V, \bar{E})$, where $\bar{E}$ is the *complement* of E, *i.e.*, for all $x, y \in V$, $(x, y) \in \bar{E} \Leftrightarrow (x, y) \notin E$.

   Clearly, (G', k') can be computed from (G, k) in polytime (in linear time, in fact).

   Also, if G contains an independent set I of size k or more, then I forms a clique in G': since G contains no edge between any two vertices of I, G' contains every edge between any two vertices of I.

   Finally, if G' contains a clique C of size k or more, then C forms an independent set in G: since G' contains every edge between any two vertices of C, G contains no edge between any two vertices of C.

3. Show that LargeSAT is NP-hard.

   Input: Propositional formula F in CNF, positive integer k

   Question: Is there an assignment of values to the variables of F that makes at least k clauses of F True?

   Solution:CNF-SAT $\leqslant_p$ LargeSAT
   Reduction:

   On input F, output (F, m) where m is the number of clauses in F.

   Clearly, (F, m) can be computed from F in polytime.
   Also, F is satisfiable iff there is an assignment of values to the variables of F that makes at least m clauses True (by definition of "satisfiable").

OneClass find more resources at www.oneclass.com

OneClass find more resources at www.oneclass.com

4. Show that Restricted3SAT is NP-hard.

   Input: Propositional formula F in 3CNF, where no variable appears in more than three clauses.

   Question: Is there an assignment of values to the variables of F that makes F True?

   Solution: 3SAT $\leqslant_p$ Restricted3SAT

   On input F, construct F' as follows:

   Start with $F' = F$.
   Scan F' and for each variable x that appears in more than three clauses, replace the first occurrence of x with a new variable x_1, the second occurrence with x_2, ..., the k-th occurrence with x_k. Then add clauses

   $$(\sim x_1 \vee x_2 \vee x_2) \wedge (\sim x_2 \vee x_3 \vee x_3) \wedge ... \wedge (\sim x_{k-1} \vee x_k \vee x_k) \wedge (\sim x_k \vee x_1 \vee x_1)$$

   F' can be constructed from F in polytime: the replacement of variables is done in linear time, and at most a linear number of new clauses need to be added.
   Also, if F is satisfiable, then it is possible to set the variables of F' to match (make all the new variables have the same value as the old variable they correspond to), and this will satisfy F'.
   Finally, if F' is satisfiable, then every new variable corresponding to some old variable x must be set to the same value (to make the new clauses True), so the old variable x can be set to this value and doing this for each variable will satisfy F.

OneClass find more resources at www.oneclass.com