Complexity: P: the class of decision problem that can be solved in poly time. NP: verified. CoNP: reject.

NP complete: ① $B \in NP$ ② every Problem A in NP is P-reducible to B / some NP complet problem A, $A \le_p B$. (P)(NP)(NP hard) NP complete

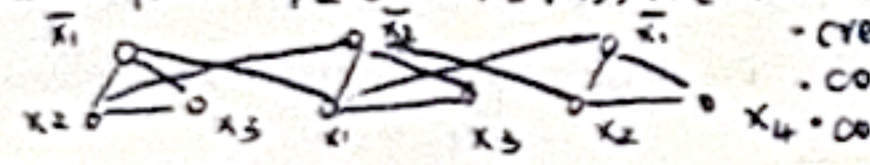
Conjuctive normal form (CNF): · variable: $x_1 x_2 \dots x_n$
· literal l: variable or its negation · clause: $C = l_1 \vee l_2 \vee \dots$
· CNF formula: $C_1 \wedge C_2 \wedge \dots$ · kCNF: each c has at most k literal (exact)

SAT & exact 3SAT is NP complete!
Given an exact 3CNF φ, is it satisfiable? → Truth values to variable s.t. φ = True.

Cook Levien theorem (main idea): if a prob is in NP, then ∃ TM $T(x,y)$ which · takes as input a prob instance x and an advice y of size $P(|x|)$ · verify in $q(|x|)$ time whether x is a Yes instance · both p and q are polynomials · x is Yes instance iff ∃ y $T(x,y)$ = Accept. ($A \le_p B$ if ∀x $x \in A \iff f(x) \in B$)

Karp reduction: Take an arbitrary instance of A, and in poly time, construct a single instance of B with the same answer (A: Yes ⇒ B: Yes ; A: NO ⇒ B: NO or B: Yes ⇒ A: Yes)

Independent Set: Given an undirected graph $G=(V,E)$, integer k
Q: Does there exists a subset of vertices $S \subseteq V$ with $|S|=k$ s.t. for each edge, at most one of its endpoint is in S? (vertex cover)
① ∈NP: Advice: an actual independent set
Algo: check if S is an independent set and $|S|=k$.
② EXACT 3SAT $\le_p$ Independent Set: Given a formula φ of exact 3 SAT with k clauses, construct an instance of (G, k) of Ind. as follow: $\varphi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$



· create 3 vertice for each clause
· connect in a Δ
· connect vertice with its ¬

EXACT 3 SAT Yes ⇒ Ind. Yes: For each C, take any l that is T in assig
Ind Yes ⇒ 3SAT Yes: · Ind contain one vertex from each Δ
· No literal and its negation are both in S
· set literals in S to T, ¬ to F, rest to arbitrary

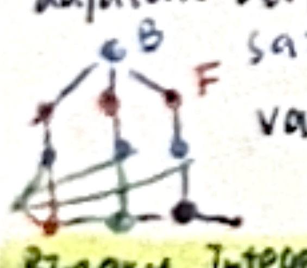
subsetSum
Input: Set of integers $S = \{w_1, \dots, w_n\}$, Integer W
Q: Is there $S' \subseteq S$ that adds up to exactly W.
① NP: accept every Yes instance with the right poly size advice; will not accept a no instance with any advice & verify in poly
② EXACT 3SET $\le_p$ Subset sum: Given a formula φ of EXACT 3SAT we want to construct (S, W) of subset sum with the same answer.

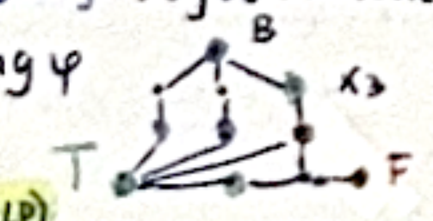
$C_1 = \bar{x} \vee y \vee z$
$C_2 = x \vee \bar{y} \vee z$
$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$

	x	y	z	C_1	C_2	C_3
x	1	0	0	0	1	0
¬x	1	0	0	1	0	1
y	0	1	0	1	0	0
¬y	0	1	0	0	1	1
z	0	0	1	1	1	0
¬z	0	0	1	0	0	1
	0	0	0	1	0	0
	0	0	0	2	0	0
	0	0	0	0	1	0
	0	0	0	0	2	0
	0	0	0	0	0	1
	0	0	0	0	0	2
W	1	1	1	4	4	4

每行 ∈ S; num selected = literal to T
dummy make sure 5=4 iff at least one literal is set to True.
set to one to make sure variable assign to T or F → W

3 Coloring input: undirected graph $G=(V,E)$ Q: Can we color each vertex of G using at most three colors such that no two adjacent vertices have the same color. eg $C = x_1 \vee \bar{x}_2 \vee x_3$
satisfying φ ⇒ valid 3 colory (by contradiction)
valid color ⇒ satisfying φ





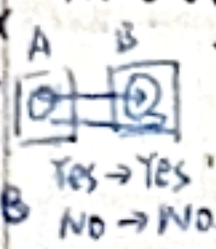
Binary Integer linear Programming (BILP)
Input: $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{R}$ (maximize)
Q: Does there exist $x \in \{0,1\}^n$ such that $c^T x \ge k$, $Ax \le b$
BILP feasibility $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, Does exist $x \in \{0,1\}^n$ $Ax \le b$
Take any formula φ of EXACT 3SAT, create binary x_i for each variable x_i in φ, represent its negation $\bar{x}_i = 1 - x_i$
For each clause c, we want at least 1 literal to be True ⇒ sum ≥ 1
eg $C = x_1 \vee \bar{x}_2 \vee \bar{x}_3$ ⇒ $x_1 + (1-x_2) + (1-x_3) \ge 1$
easy to check: · polynomial reduction · feasible sol iff φ is satisfiable

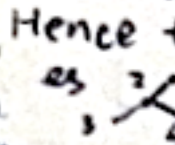
ILP feasibility input: ... Q: Does exist $x \in \mathbb{Z}^n$ s.t. $Ax \le b$.

vertex cover Input: undirected graph $G=(V,E)$, integer k
Q: Does there exist a vertex cover of size k
i.e. exist $S \subseteq V$ with $|S|=k$ s.t. every edge is incident to at least one vertex in S

例子: size 4 ✓ size 3 ✗
Claim: S is a vertex cover iff V\S is an independent set.
S is vertex cover ⇔ for every $(u,v) \in E$, at least one of $\{u,v\}$ is in S.
⇔ for every $(u,v) \in E$, at most one of (u,v) is in V\S ⇔ No two vertices of V\S are connected by an edge ⇔ V\S is ind. set.
Ind. set $\le_p$ vertex cover: Take an arbitrary instance (G, k) of ind. set → covert to (G, n−k) vertex cover.

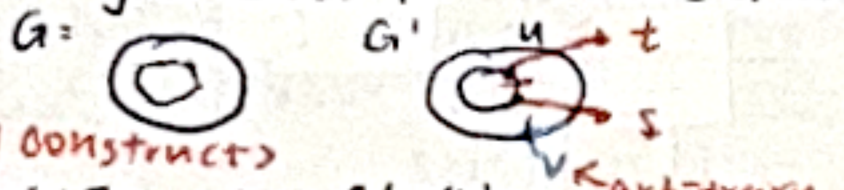
Set cover: Input: A universe of elements U, a family of subsets S and an integer k Q: Do there exist k sets from S whose union is U
vertex cover $\le_p$ Set cover: Given an instance of vertex cover with graph $G=(V,E)$ and k, create set cover instance: · Set U=E
· For each v∈V, S contains a set S_v of all the edges incident on v
· Selecting k set whose union is U = selecting k vertices such that union of their incident edges covers all edges.
Hence the problem has the same answer



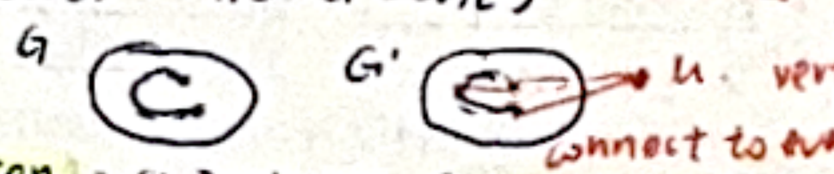
eg  $U = \{\{1,2\},\{2,3\},\{2,4\},\{4,5\}\}$
$S = \{\{(1,2)\}, \{(1,2)(2,3)(2,4)\}, \{(2,3)\}, \{(2,4)\} \dots$ (2,5)

NP vs CoNP: short prove for Yes ✓ for No ✗
Does there exist x satisfying φ? → Does there exist no x satisfy φ
$P \subseteq NP \cap CoNP$ ✓ $P = NP \cap CoNP$?

HC ≤ HP: Given arbitrary $G=(V,E)$ for HC, create $G'=(V',E)$
for HP:
Yes HC ⇒ Yes HP
Yes HP ⇒ Yes HC (by construct)



HP ≤ HC: HP: $G=(V,E)$ HC: $G'=(V',E')$
Yes HP ⇒ Yes HC
Yes HC ⇒ Yes HP



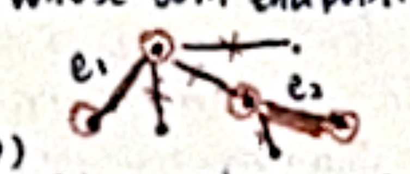
Subset sum ≤ Partition: SS instance S, W make T
Yes SS ⇒ Yes partition $T = S \cup \{|\Sigma - 2w|\}$ $\Sigma = 2T - 2w$
Yes partition ⇒ Yes SS $\Sigma_{a \in S} a$ sum of all element
$S' \subseteq S$ with sum = w
$S \setminus S'$ sum = Σ−w
相差 Σ−2w, 再加上 |Σ−2w| 2个 set 相等.
(Partition with equal sum $S_1 \cap S_2 = \emptyset$, $S_1 \cup S_2 = S$)
$P_1 \cup P_2 = T$, Each Σ = Σ−w $|t-2w| \in P_1$
$P_1 \setminus |t-2w| \subseteq S$, Σ = w

Approximation Algo (别忘 sort)
Approximation ratio of ALG on instance I: $\frac{Profit(OPT(I))}{Profit(ALG(I))}$ or $\frac{cost(ALG(I))}{cost(OPT(I))}$ (convention ≥ 1)
PTAS and FPTAS: arbitrary close to one approximation
PTAS: For every ε>0, there is a (1+ε)-approximation algo that runs in poly(n) on instance of size n. Note: could have exponential dependence on $\frac{1}{\varepsilon}$
FPTAS: poly(n, $\frac{1}{\varepsilon}$) on instance of size n.

Greedy Approximation: make greedy decision without future decision
Makespan minimization: Input: m identical machines, n jobs, job j requires processing time t_j. Output: Assign jobs to machines to minimize makespan (maximum load on machine)
load on machine i: $L_i = \sum_{j \in S[i]} t_j$ $\max L = \max L_i$
Greedy list-scheduling algo: (Assign each job j to a machine with smallest load so far. (2 approximation)) sort job in some order
let optimal makespan = L^*
Fact 1: $L^* \ge \max_j t_j$ Fact 2: $L^* \ge \frac{1}{m} \sum_j t_j$
Prove: · Suppose machine i is the bottleneck under greedy (so $L = L_i$)
· let j^* be the last job scheduled on machine i by greedy · right before j^* was assigned to i, i has the smallest load · load of the other machines could have only increased from then · $L_i - t_{j^*} \le L_k, \forall k$ ($L_i - t_j^* \le \frac{1}{m}\sum_{k=1}^{M} L_k$)
· Average overall k: $L_i - t_j^* \le \frac{1}{m}\sum_j t_j \Rightarrow L_i \le t_j^* + \frac{1}{m}\sum_j t_j \le L^* + L^* = 2L^*$
$2 - \frac{1}{m}$ tight (as bad as): eg: m(m−1) job of length 1, one job length m
G evenly distribute unit length jobs on all m machines, and assign the last heavy job makes makespan $m-1+m = 2m-1$. optimal: is m by evenly distributing jobs among m−1 machine and put the single heavy job on the remaining.
Greedy LPT (longest processing time First): $t_1 \ge t_2 \dots \ge t_n$.
Fact 3: if the bottleneck machine i has only one job j, then the solution is optimal. current solution has $L = L_i = t_i$ (Fact 1)
Fact 4: If there are more than m jobs, then $L^* \ge 2t_{m+1}$
The first m+1 jobs each have processing time at least t_{m+1}. By the pigeonhole principle, the opt sol must put at least two of them on the same machine.
LPT achieves 3/2 approximation: consider a bottleneck machine i and job j^* that was last scheduled on this machine by the G
case 1: Machine i has only one job j^* (Fact 3) 1-approximation
case 2: Machine i has at least two jobs · j^* must $t_j^* \le t_{m+1}$
As before $L = L_i = (L_i - t_j^*) + t_j^* \le L^* + \frac{1}{2}L^* = \frac{3}{2}L^*$ (F4)

$L_i - t_j^* \le \frac{1}{m}\sum_{k=1}^{m} L_k \le \frac{1}{m}\sum_{k=1}^{n} t_i \le L^*$ 2m−1
Tight example for Greed LPT: 2 job each of length m, m+1 ... one more job of length m. Greedy LPT has make span 4m−1
OPT has make span 3m. Thus, approximation ratio is at least as bad as $\frac{4m-1}{3m} = \frac{4}{3} - \frac{1}{3m}$.

Unweighted Vertex Cover: Undirected graph $G=(V,E)$ Output: vertex cover S of minimum cardinality (smallest vertex cover)
Greedy edge-selection algo: start with S=∅, while there exists an edge whose both endpoints are not in S, add both its endpoints to S




```
S ← ∅
E' ← E
while (E' ≠ ∅)
    let (u,v) ∈ E' be an arbitrary edge
    M ← M ∪ {(u,v)}  ← M is a match (no vertex appears twice in it)
    S ← S ∪ {u} ∪ {v}
    Delete from E' all edges incident to either u, v
return S
```

achieves 2-approximations
Theorem: Greedy edge selection algo for unweighted vertex cover
ob1: For any vertex cover S^* and any matching M, $|S^*| \ge |M|$, where |M| = number of edges in M. (S^* must contain one end point of each edge in M) ob2: G finds a vertex cover of size $|S| = 2|M|$
Hence, $|S| \le 2|S^*|$, where S^* = min vertex cover.
Collary: If M^* is a maximum matching, and M is a maximal matching then $|M| \ge \frac{1}{2}|M^*|$ proof: by design $|M| = \frac{1}{2}|S|$, $|S| \ge |M^*|$
Hence $|M| \ge \frac{1}{2}|M^*|$ (find maximum cardinality matching)

Greedy vertex selection algo
Start with S=∅
while S is not a vertex cover: (highest degree)
· Choose a vertex v which maximizes the number of uncovered edges incident on it
· Add v to S
Give $O(\log d_{max})$ approximation, d_{max} is the maximum degree of any vertex

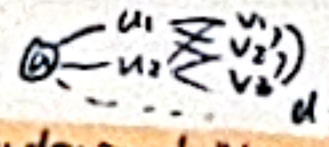
LP relaxation
Weighted Vertex Cover: minimum total weight.
LP with real variables: $\min \sum_v W_v \cdot X_v$
s.t. $x_u + x_v \ge 1, \forall (u,v) \in E$
$x_v \ge 0$, $\forall v \in V$
let x^*_{LP} be an optimal LP solution and x^*_{ILP} be an opt ILP solution $c^T x^*_{LP} \le c^T x^*_{ILP}$
· effectively round x^*_{LP} to an ILP feasible solution $\hat{x}$ without increasing the objective too much. prove $c^T\hat{x} \le \rho c^T x^*_{LP} \Rightarrow c^T\hat{x} \le \rho c^T x^*_{ILP}$
ρ-approximation
claim 1: $\hat{x}$ is a feasible solution of ILP (i.e. a vertex cover)
For every edge $(u,v) \in E$, at least one of $\{x^*_u, x^*_v\}$ is at least 0.5, so at least one of $\{\hat{x}_u, \hat{x}_v\}$ is 1
claim 2: $\sum_v w_v \hat{x}_v \le 2\sum_v W_v \cdot x^*_v$ ↳ LP opt solution x^*
Hence, $\hat{x}$ is a vertex cover with weight at most 2×LP opt ≤ 2 ILP opt

General Strategy Max $c^T x$ s.t. $Ax \le b$, $x \in \mathbb{N}$ (need not be binary)
Max $c^T x$ s.t. $Ax \le b$, $x \in \mathbb{R}^+$ (LP relaxation)
· LP optimal value ≥ ILP opt value (for maximization)
· x^* be LP opt sol · round x^* to $\hat{x}$ such that $c^T\hat{x} \ge \frac{c^T x^*}{\rho} \ge \frac{ILP\ opt}{\rho}$
· Gives ρ approx, Best ρ is the integrality gap.

Local search Paradigm
Max Cut Input: An undirected graph $G=(V,E)$ Output: A partition (A, B) of V that maximizes the number of edges going across the cut. i.e. maximizes |E'| where $E' = \{(u,v) \in E \mid u \in A, v \in B\}$
Algo: Initialize (A, B) arbitrary
while there is a vertex u such that moving u to the other side improves the object value:
move u to the other side
When increase objective value? when u has more incident edges going within the cut than across the cut. i.e. when $|\{(u,v) \in E \mid v \in A\}| > |\{(u,v) \in E \mid v \in B\}|$ When algo stop? Every iteration increase the # of edges across the cut by at least 1, so the algo must stop in at most |E| iteration. Approx ratio? At the end, every vertex has at least as many edges going across the cut as within the cut. Hence, at least half of all edges must be going across the cut.
Weighted Max Cut give integral edge weights $w: E \to \mathbb{N}$
Goal: maximize the total weight of edges going across the cut.
($\sum_{e \in E} W_e$ iteration ⇒ exponential in the input length.) ⇒ only move vertices when it sufficiently improves the objective value ⇒ we can find a 2+ε approximation in time polynomial in the input length and $\frac{1}{\varepsilon}$

Exact Max k SAT input: An Exact k-SAT formula $\varphi = C_1 \wedge C_2 \dots \wedge C_m$ where each clause C_i has exactly k literals, and a weight w_i of each clause C_i
output: A truth assignment τ maximizing the total weight of clauses satisfied under τ.
let W(τ) be the total weight of clauses satisfied under τ.
local neighborhood: $N_d(\tau)$ = set of all truth assignment τ' which differ from τ in the values of at most d variables.
Theorem: The local search with d=1 gives a 2/3 approximation to EXACT MAX 2SAT
Proof: Let τ be a local optimum. S_0 = set of clauses not satisfied under τ
S_1 = set of clauses from which exactly one literal is true under τ
S_2 = set of clauses from which both literals are true under τ.
Goal: $W(S_1) + W(S_2) \ge \frac{2}{3}(W(S_0) + W(S_1) + W(S_2))$
or $W(S_0) \le \frac{1}{3}(W(S_0) + W(S_1) + W(S_2))$
let A_j = set of clauses in S_0 involving variable j ⇒ $W(A_j)$
let B_j = set of clauses in S_1 involving j s.t. literal of j is true under τ ⇒ $W(B_j)$: C involves j if it contains x_j or $\bar{x}_j$
$2W(S_0) = \sum_j W(A_j)$ (every clause in S_0 is counted twice on RHS)
$W(S_1) = \sum_j W(B_j)$ (every clause in S_1 is counted one for RHS for variables whose literal was true under τ)
For each j, $W(A_j) \le W(B_j)$ (for local optimality of τ, since otherwise flipping the truth value of variable j would have increased the total weight)
⇒ $2W(S_0) \le W(S_1)$ ⇒ $W(S_0) \le \frac{1}{3}(W(S_0) + W(S_1) + W(S_2))$
$\frac{3}{4}$ approximation: include one more assignment in the neighbour $N(\tau) = N_1(\tau) \cup \{\tau^c\}$, where τ^c = complement of τ.
or redefine Better: improve the potential $1.5W(S_1) + 2W(S_2)$
"non-oblivious local search"
General: $(2^k - 1)/2^k$ approximation for EXACT-k-SAT

Greedy Independent set
```
I ← ∅
while V ≠ ∅ do
    select v∈V with the smallest numer of neighbors in V
    I ← I ∪ {v}
    V ← V \ ({v} ∪ {u: (u,v) ∈ E})
end.
```
① upper bound d+1. d
let M be a maximum cardinality Ind. set. ($|M| \le n$). each iteration, remove at most d+1
$|I| \ge \frac{n}{d+1} \ge \frac{|M|}{d+1}$ ∴ $\frac{|M|}{|I|} \le d+1$.
② lower bound. consider a graph $V = \{a, v_1 \dots v_k, u_1 \dots u_k\}$



$D(a) = k$ $D(u_i) = k+1$ $d = D(v_i) = 2k-1$
$G = \{a, v_1\}$ $|G| = 2$ ∴ $\frac{|OPT|}{|G|} = \frac{d+1}{4}$
$OPT = \{u_1 \dots u_k\}$ $|OPT| = k$

Randomized Algorithm
linearity of expectation $E[X+Y] = E[X] + E[Y]$
upper bound, For any two events $Pr[A \cup B] \le Pr(A) + Pr(B)$
Exact Max-k-SET Random Algo: set each variable to True with prob $\frac{1}{2}$ and to False with prob $\frac{1}{2}$
let τ be a random assignment.
For each clause C_i: $Pr[C_i \text{ is not satisfied}] = \frac{1}{2^k}$
Hence $Pr[C_i \text{ is satisfied}] = (2^k - 1)/2^k$
$E[W(\tau)] = \sum_{i=1}^{m} W_i \cdot Pr[C_i \text{ is satisfied}] = \frac{2^k - 1}{2^k}\sum_{i=1}^{m} W_i \ge \frac{2^k - 1}{2^k} OPT$

Derandomization: (Think about them one by one)
Goal: Gradully convert the random assignment τ to a deterministic assignment $\hat{\tau}$ s.t. $E[W(\hat{\tau})] \ge E[W(\tau)]$
$E[W(\tau)] = Pr[X_1 = T]\, E[W(\tau) \mid X_1 = T] + Pr[X_1 = F]\, E[\dots]$ ($\ge \frac{2^k - 1}{2^k} OPT$)
$= \frac{1}{2}E[W(\tau) \mid X_1 = T] + \frac{1}{2}E[W(\tau) \mid X_1 = F]$
Hence $\max(E[W(\tau) \mid x_1 = T], E[W(\tau) \mid x_2 = F]) \ge E[W(\tau)]$
(just x_1 be deterministic, $x_2 \dots x_n$ random) (pick better one)

Derandomized Algo
For i = 1, ..., n
· let $z_i = T$ if $E[W(\tau) \mid x_1 = z_1, x_2 = z_2 \dots x_i = T] \ge E[W(\tau) \mid x_1 = z_1, x_2 = z_2 \dots x_i = F]$
and $z_i = F$ otherwise
· set $x_i = z_i$
"method of conditional expectation"
$E[W(\tau) \mid x_1 = z_1, \dots, x_{i-1} = z_{i-1}, x_i = T]$
$= \sum_r W_r \cdot Pr[C_r \text{ is satisfied} \mid x_1 = z_1, x_2 = z_2 \dots x_i = T]$
· Set the values of $x_1, x_2 \dots x_{i-1}, x_i$
· If C_r resolves to True already, the corresponding prob. is 1
· otherwise, if there are l literals left in C_r, after setting $x_1 \dots x_i$ the corresponding prob is $(2^l - 1)/2^l$

MAX SET input $\varphi = C_1 \wedge C_2 \dots \wedge C_m$, where each C has weight $w_i \ge 0$ (and can have any number of literals)
output: Truth assignment that approximately maximize the weight of clauses satisfied.

LP formulation of MAX-SAT

In MAX_SAT

variables: $y_1, \dots, y_n \in \{0,1\}$ ($y_i = 1$ iff variables x_i = True)

$z_1, \dots, z_m \in \{0,1\}$, ($z_j = 1$ iff clause C_j is satisfied)

$\max \sum_j w_j z_j$ LHS=0 => $z_j \le 0$ LHS≥1 => $z_j = 1$

s.t. $\sum_{x_i \in C_j} y_i + \sum_{\bar{x}_i \in C_j} (1-y_i) \ge z_j, \forall j \in \{1 \dots m\}$

$y_i, z_j \in \{0,1\}, \forall i \in \{1, \dots, n\}, j \in \{1 \dots m\}$

LP relaxation: $y_1 \dots y_n \in [0,1]$ ($y_i = 1$ iff x_i = True)

$z_1 \dots z_m \in [0,1]$ ($z_j = 1$ iff C_j is satisfied)

$\max_j w_j z_j$

s.t. $\sum_{x_i \in C_j} y_i + \sum_{\bar{x}_i} (1-y_i) \ge z_j \quad \forall j \in \{1 \dots m\}$

$y_i, z_j \in [0,1] \quad \forall i \in \{1..n\}, j \in \{1..m\}$

Randomized Rounding: Find the optimal solu (y^*, z^*) of LP

compute a random ~~LP~~ solution $\hat{y}$ such that

- each $\hat{y}_i = 1$ with prob y_i^* and 0 with probability $1-y_i^*$
- Independently of other $\hat{y}_i$'s
- The output of the algo is the corresponding True assignment

What is the Pr[C_j is satisfied] if C_j has k literal

$1 - \prod_{x_i \in C_j} (1-y_i^*) \prod_{\bar{x}_i \in C_j} y_i^* \;(\frac{}{k})^k$

AM-GM: $\frac{a_1 + \dots + a_k}{k} \ge \sqrt[k]{a_1 \dots a_k}$

$\ge 1 - (\sum_{x_i \in C_j} (1-y_i^*) + \sum_{\bar{x}_i \in C_j} y_i^*)^k$

$= 1 - (k - n_{neg} - \sum_{x_i \in C_j} y_i^* + \sum_{\bar{x} \in C_j} y_i^*)^k / k$

$\ge 1 - (\frac{k - z_j^*}{k})^k = 1 - (1 - \frac{z_j^*}{k})^k$ and $k \in \mathbb{N}$

Claim: $1 - (1-\frac{z}{k})^k \ge (1 - (1-\frac{1}{k})^k) z$ for all $z \in [0,1]$

$\therefore \Pr[C_j \text{ is satisfied}] \ge (1-(1-\frac{1}{k})^k) z_j^* \ge (1-\frac{1}{e}) z_j^*$

Hence. E[# weight of clauses satisfied] $\ge (1-\frac{1}{e}) \sum_j w_j z_j^* \ge (1-\frac{1}{e})$ OPT

Standard inequality: $1-x \le e^{-x}$, $(1-p)^k \le e^{-kp}$

optimal LP obj ≥ opt ILP obj

Improve: run "LP relaxation + randomized rounding" and "naive randomized ~~go~~ algo". return the best of two => $\frac{3}{4}$ approximation! (can be derandomized)

weighted MAX CUT Input: undirected graph $G=(V,E)$, and weight w_e for each edge $e \in E$. Output: A cut (A,B) ($V = A \cup B$, $\emptyset = A \cap B$) that approximately maximizes the total weight of edges across the cut. **local search**: repeatedly find a vertex u such that moving it to the other side would increase the objective value and do so. (2 approximation, but not poly)

randomized: puts each vertex $i \in V$ to A and B with prob. $\frac{1}{2}$ each, independently of the other vertices.

2-approx: let (A,B) be the random cut formed by this algo.

$E[\sum_{e \in E(A,B)} w_e] = E[\sum_{e \in E} w_e I[e \in E(A,B)]] = \sum_{e \in E} E[w_e I(e \in E(A,B))]$ P(A)

$= \sum_{e \in E} (w_e \cdot \Pr[e \in E(A,B)]) = \frac{1}{2} \sum_{e \in E} w_e \ge \frac{1}{2} \sum_e w_e \ge w(OPT)$ some subset

derandomized We consider vertices of v being assigned to either A or B one by one. We start off by all these choices being random and make them deterministic one by 1. let $w(A,B,C)$ denote the expected total weight of edges going across the cut in this ~~derando~~ partially derandomized sol.

$w(A,B,C) = \sum_{e \in E(A,B)} w_e + \frac{1}{2} \sum_{e \in E(A,C)} w_e + \frac{1}{2} \sum_{e \in E(B,C)} w_e + \frac{1}{2} \sum_{e \in E(C,C)} w_e$

evaluate in $O(|V|+|E|)$. at each v, $w(A \cup \{v\}, B, C \setminus \{v\}) - w(\dots) > 0$?

$= \frac{1}{2} \sum_{e \in E(B,v)} w_e - \frac{1}{2} \sum_{e \in E(A,v)} w_e$ (only check incident edges)

sum of degree of all ~~edge~~ vertices $= 2|E| \Rightarrow O(|V|+|E|)$

Algo: $A, B \leftarrow \emptyset$
for $v \in V$, do
  if $\sum_{e \in E(B,v)} w_e - \sum_{e \in E(A,v)} w_e > 0$ then
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow B \cup \{v\}$
return (A,B)

BST (AVL): Insert(logn) delet(logn) Search(logn) traverse O(logn)

binary search O(logn)

heap: max O(1) sort(nlogn) build O(n) insert O(logn) extract max O(logn)

Divide & conquer

$T(n) \le 2T(\frac{n}{2}) + O(n) = O(n\log n)$

counting inversions: ~~Pro~~ Correctness: induction on n. Assume the sub parts correctly, argu base & combine step

Master theorem

$T(n) \le aT(\frac{n}{b}) + f(n)$, $d = \log_b a$ if $f(n) = O(1) \Rightarrow O(n)$

if $f(n) = O(n^{d-\varepsilon})$ for some constant $\varepsilon > 0$. Then $T(n) = O(n^d)$

If $f(n) = O(n^d \log^k n)$ for some $k \ge 0$, Then $T(n) = O(n^d \log^{k+1} n)$

If $f(n) = O(n^{d+\varepsilon})$ for some constant $\varepsilon > 0$, Then $T(n) = O(f(n))$

compare $f(n)$ with $n^{\log_b a}$

Greedy Algorithm "natural order" "include if compatible"

Goal: find a solution x maximizing or min objective function f

Approach: instead of taking all decision together, take one at a time. Take the next decision greedily to maximize the immediate benefit without knowing how you will take future decision (most in poly)

LP Interval scheduling: max size of subset of mutually compatible jobs.

(EFT): sort jobs by finish time, say $f_1 \le f_2 \le \dots \le f_n$ O(nlogn)

For each job j, we need to check if it is compatible with all previously choose job => check if $s_j \ge f_{i^*}$, where i^* is the last added job • For any job added before i^*, $f_i \le f_{i^*}$

• By keeping track f_{i^*}, we can check job j in O(1) time. T=O(nlogn)

Proof by Induction: let S_j be the subset of jobs picked by G after considering the first j jobs by increasing finish time.

let $S_0 = \emptyset$. Promising: if there is a way to extend it to an opt sol by picking some subset of jobs $j+1 \dots n$ => $\exists T \subseteq \{j+1 \dots n\}$ s.t. $O_j = S_j \cup T$ is optimal. Inductive claim: For all $t \in \{1 \dots n\}$, S_t is promising.

Base case: For t=0, $S_0 = \emptyset$ is clearly promising.

Ind. hyp: Suppose the claim hold for $t = j-1$ and optimal sol O_{j-1} extends it.

Ind. Step: at $t = j$, two possibility:

(1) Greedy did not select job j, so $S_j = S_{j-1}$

• j must have conflict with some job in S_{j-1}. Since $S_{j-1} \subseteq O_{j-1}$, O_{j-1} also cannot possibly include j. Hence $O_j = O_{j-1}$ also extends S_j

(2) Greedy did select j. So $S_j = S_{j-1} \cup \{j\}$

• Consider the earliest job r in $O_{j-1} \setminus S_{j-1}$

- consider O_j obtained by replacing r with j in O_{j-1}

• O_j is still feasible due to $f_j \le f_r \le s_l$ for any $l \in O_{j-1} \setminus \{S_{j-1} \cup r\}$

• O_j is still optimal and extends S_j

Prove by contradiction

• Suppose for contradiction that greedy sol is not optimal.

- Say Greedy select jobs $i_1 \dots i_k$ sorted by finish time

• consider an optimal sol $j_1 \dots j_m$ by finish time, which matches G for as many indices as possible. That is $i_1 = j_1 \dots i_r = j_r$ for the greatest possible r.

claim: $r < k \le m$. Consider a new solu $i_1 \dots i_r, i_{r+1}, j_{r+2} \dots j_m$

we have replaced j_{r+1} by i_{r+1} in this optimal sol.

This is still feasible because $f_{i_{r+1}} \le f_{j_{r+1}} \le s_{j_t}$, for $t \ge r+2$

This is still optimal bc m jobs are selected.

But it matches the Greedy solution in r+1 indices → contradiction!

Interval partition: (EST) min # of classrooms O(n)

(store classrooms in a priority Q (key = latest finish time)

Add job j to classroom O(nlogn) & update queue.

Prove: # ≥ depth (lower bound) depth ≥ d = # of G used (UPP)

Minimizing lateness (EDT) minimize maximum lateness

Prove: ob1: ∃ opt no idel time ob2: EDF no idel time (3): EDT has no inversion (4) ∃ inversion => ∃ inversion in adjacent pairs (5) swap adjacently scheduled inverted jobs doesn't increase lateness, but ↓ inversion by 1. Proof of opt by reverse induction: claim for each $r \in \{0, 1, \dots, \binom{n}{2}\}$ there is an opt permutation with at most r inversions. Base case: $r = \binom{n}{2}$ use any opt permutation

I.H: Suppose the claim holds for $r = t+1$

I.S: let opt* be an optimal permutation with at most t+1 inversions. If it has at most t inversion done! If it has exactly t+1 inversion find and swap an adjacent pair to get a new permutation with t inversions => when t=0 ...

Huffman coding Build a priority Queue by adding (x, w_x) for each symbol x. while |queue| > 2: Take the two symbol with the lowest weight (x, w_x) and (y, w_y) · merge them into one symbol with weight $w_x + w_y$ O(nlogn)

Proof: take any opt Tree. let x be the label with the lowest frequency. If f doesn't have the largest encoding, swap it with.. · due to optimality, x must have a sibling. If it's not y, swap.. let H be the Huffman Tree. T be opt in which x, y sibling. $\text{leng}(H') \le \text{len}(T')$ $\text{len}(H) = \text{leng}(H') + w_x + w_y \le \text{len}(T) = \text{len}(T') + w_x + w_y$

Dynamic programming

1A global array

Weighted Interval Scheduling

$OPT[j] = \begin{cases} 0, & j = 0 \\ \max\{OPT(j-1), w_j + OPT(P[j])\} & \text{if } j > 0 \end{cases}$

$S(j) = \begin{cases} \bot, & j = 0 \\ N, & \text{if } j > 0 \wedge OPT(j-1) \ge w_j + OPT(P[j]) \\ Y, & \text{if } \quad < \end{cases}$

Knapsack problem item 1 ... i in pack with capacity W

(1) when capacity small: OPT(i,w) = max value using only

$OPT(i,w) = \begin{cases} 0, & \text{if } i = 0 \\ OPT(i-1, w), & \text{if } w_i > w \\ \max\{OPT(i-1, w), v_i + OPT(i-1, w - w_i)\} & \text{o/w} \end{cases}$ O(nW)

(2) when value small: OPT(i,v) = min capacity need to pack value ≥ v, using 1, ..., i O(nV)

$OPT(i,v) = \begin{cases} 0, & \text{if } v \le 0 \\ \infty, & \text{if } v > 0, i = 0 \\ \min\{w_i + OPT(i-1, v - v_i), OPT(i-1, v)\}, & \text{if } v > 0, i > 0 \end{cases}$

Single source shortest path opt(t,i) = shortest path from s to t using at most i edges.

$OPT(t,i) = \begin{cases} 0, & i = 0, t = s \\ \infty, & i = 0 \wedge t \ne s \\ \min\{OPT(t, i-1), \min_u OPT(u, i-1) + l_{ut}\} & \text{o/w} \end{cases}$

network flow

• capacity constraint: $\forall e \in E$, $0 \le f(e) \le c(e)$

• Flow conservation: $\forall v \in V \setminus \{s,t\}$ $\sum_{e \text{ entering } v} f(e) = \sum_{e \text{ leaving } v} f(e)$ (f^{in}, f^{out})

• $v_f = f^{out}(s) = f^{in}(s)$

• residual Graph G_f: Forward edge capacity = $c(e) - f(e)$; reverse edge: $e^{rev} = (u,v)$ capa = f(e)

• Augment: sending bottleneck(P,f) units of flow along P.

Ford-Fulkerson Algo: O((m+n)C)

```
Max Flow(G):
  set f(e)=0 for all e in G          -> use BFS, shortest path O(nm^2)
  while P = find path(s,t, residual(G,f)) != None:   O(C)
      A = augment(f,P)               -> O(m+n)
      update residual(G,f)
  return F'
```

Cuts and cuts capacity cap(A,B) = Σ of capacity of edges leaving A

Th: for any flow f and and cut (A,B). $v(f) = f^{out}(A) - f^{in}(A)$

For any flow f and and any s-t cut (A,B) $v(f) \le cap(A,B)$

$\therefore \max_f v(f) \le \min_{(A,B)} cap(A,B)$

FF finds the maximum flow: f be the flow returned by FF, A^* = nodes reachable from s in G_f

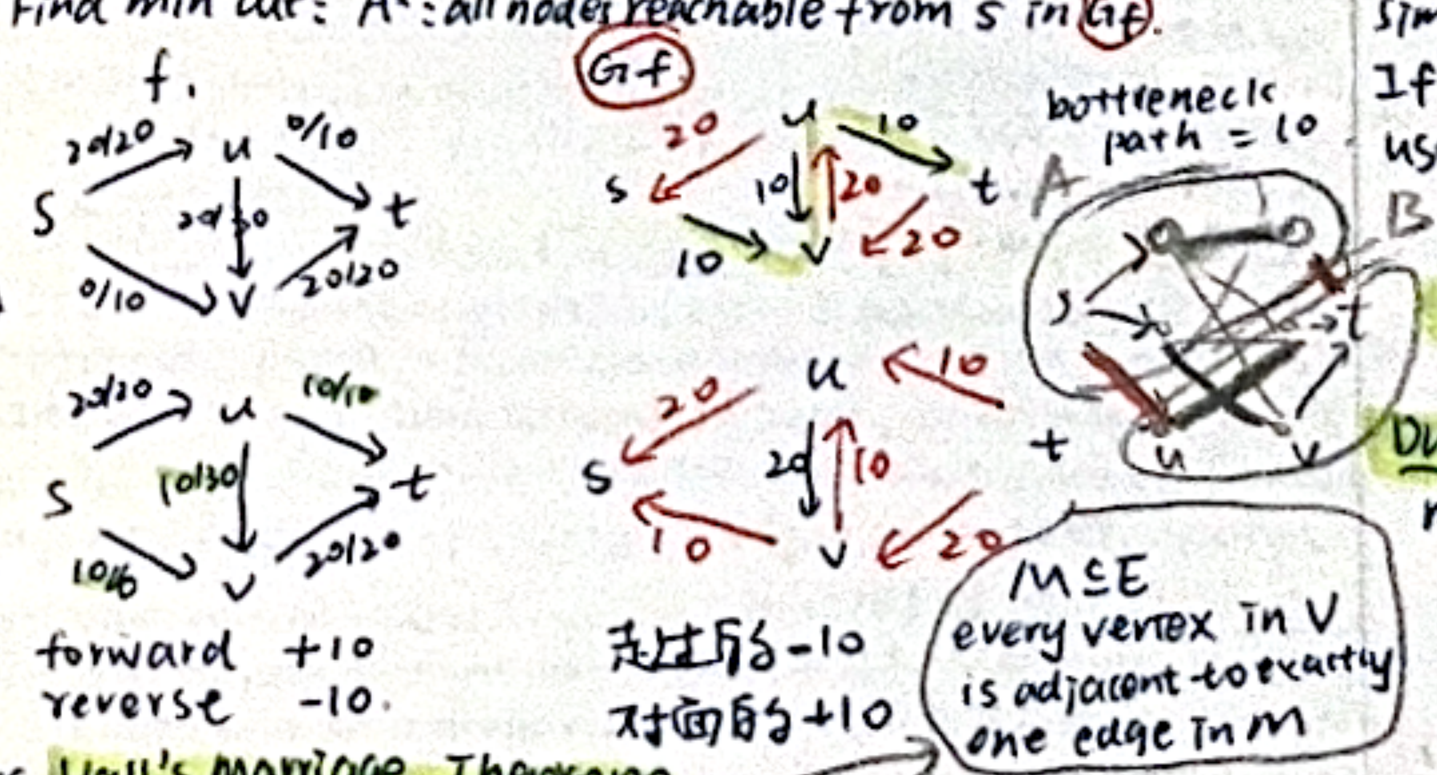
(1) (A^*, B^*) is valid cut ($t \notin A^*$)

(2) blue edge is saturated; red edge has zero flow

$\therefore v(f) = f^{out}(A^*) - f^{in}(A^*) = cap(A^*, B^*)$

Max Flow Min Cut Theorem In any graph, the value of the maximum flow is equal to the capacity of the min cut

Find min cut: A^*: all nodes reachable from s in G_f.



Hall's Marriage Theorem

G has a perfect matching iff $|N(S)| \ge |S|$ for each $S \subseteq V$

proof: suppose G does not have a perfect matching

Hence maxflow = min cut < n

let (A,B) be the min cut.

• can have any u→v (∞ capacity)

has unit capacity edge $s \to u \cap B$ and $v \cap A \to t$

$cap(A,B) = |u \cap B| + |v \cap A| < n = |u|$

so $|v \cap A| < |u \cap A|$

But $N(u \cap A) \subseteq v \cap A$

So $|N(u \cap A)| \le |v \cap A| < |u \cap A|$

Menger's Theorem

In any directed / undirected graph, the maximum # of edges disjoint s→t path = min # of edges whose removal disconnects s and t.

Linear Programming

standard form

$\max c^T x$ s.t. $Ax \le 0$, $x \ge 0$

$\max c^T x$ s.t. $a_1^T x \le b_1, \dots, a_m^T x \le b_m$ } m const; $x \ge 0$ n constraint

≥ => −( ) => ≤

= => ≥ & ≤

min => −( ) => max

unconstrained $x = x' - x''$, $x' \ge 0$, $x'' \ge 0$

Fail when: infeasible or unbounded.

Simplex Algo

let v be any vertex of the feasible region

while there is a neighbor v' of v with better objective value

  Set v = v'

WC: exponential

slack form $z = c^T x$, $s = b - Ax$, $s, x \ge 0$.

$\max 2x_1 - 3x_2 + 3x_3$

s.t. $x_1 + x_2 - x_3 \le 7$ => $x_4 = 7 - x_1 - x_2 + x_3$

$-x_1 - x_2 + x_3 \le -7$ => $x_5 = -7 + x_1 + x_2 - x_3$

$x_1 - 2x_2 + 2x_3 \le 4$ basic $x_6 = 4 - x_1 + 2x_2 - 2x_3$

$x_1, x_2, x_3 \ge 0$ $x_1 \dots x_6 \ge 0$

(1) Start with feasible vertex (if $b \ge 0$ => $x = 0$ is feasible)

(2) entering variable: non basic variable with >0 efficient

$z = 3x_1 + x_2 + 2x_3$

$x_4 = 30 - x_1 - x_2 - 3x_3 \Rightarrow x_1 \le 30$ ← $x_4 \ge 0$, $x_2 = x_3 = 0$

$x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \Rightarrow x_1 \le 12$

$x_6 = 36 - 4x_1 - x_2 - 2x_3 \Rightarrow x_1 \le 9$ tightest

$z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4}$ 把 x_1 用 x_6 替换.

$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}$

$x_4 = 21 - \frac{3}{4}x_2 - \frac{5}{2}x_3 + \frac{x_6}{4}$

$x_5 = \dots$

final $z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2}{3}x_6$

$x_1 = 8 + \dots$, $x_2 = 4 + \dots$, $x_3 = 18 + \dots$

LP1: $\max c^T x$ s.t. $a_1^T x \le b_1 \dots$, $x \ge 0$

LP2: $a_1^T + s_1 = b_1$ ⋮ $x, s \ge 0$

LP3: $a_1^T x + s_1 = b_1$ ⋮ $-a_2^T - s_2 = -b_2$, $x, s \ge 0$

LP4: $\min \sum_i z_i$ s.t. $a_1^T x + s_1 + z_1 = b_1$, $-a_2^T x - s_2 + z_2 = -b_2$, $x, s, z \ge 0$

solve ~~using~~ LP4 using simplex with initial basic sol being $x = s = 0$, $z = |b|$. If opt value is 0, extract a basic feasible x^* from it use it to solve LP, using simplex. If opt value for LP4 is >0, then LP1 is infeasible.

RT: # vertices of a polytope can be exponential in # constraint.

Dual: Primal — upper bound of Primal — Dual

$\max -4x - 3y' + 3y'' + 6z$ => $\min -2a + 10b - 10c$

$2x - y' + y'' + 3z \le -2$ — $2a + 3b - 3c \ge -4$

$3x + 2y' - 2y'' + 5z \le 10$ — $-a + 2b - 2c \ge -3$

$-3x - 2y' + 2y'' - 5z \le -10$ — $a - 2b + 2c \ge 3$

$3a + 5b - 5c \ge 6$

$a, b, c \ge 0$

If Dual = primal => both opt

weak $c^T x \le y^T b$ strong $c^T x^* = (y^*)^T b$

$|x| \le 3 \Rightarrow x \le 3, -x \le 3$

$\min 3|x| + y \Rightarrow t \ge x$, $t \ge -x$, $\min 3t + y$

$\min \max(3x+y, x+2y) \Rightarrow t \ge 3x + y$, $t \ge x + 2y$, $\min t$.

21.6
3)65
6
5
3
20