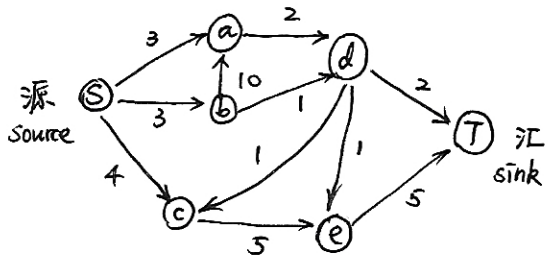
- Max Flow



$G=(V,E)$ 边视作水管. 边权 $C_{uv}>0$ 为 capacity (水管容量)

Goal: Find f_{uv} (flow). $\forall (u,v)\in E$, s.t.

1) $\forall (u,v)\in E.\ 0\le f_{uv}\le C_{uv}$ (水管不能爆. 也不能倒流)

2) $\forall v\in V\setminus\{s,t\}.$

$$\sum_{u:(u,v)\in E} f_{uv} = \sum_{w:(v,w)\in E} f_{vw}$$

(非源.汇的节点. 进入的流量等于流出的流量)

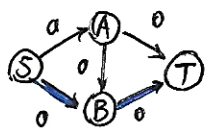
即中间节点不储存水. 流量平衡

We want to $\max_f \sum_{v:(s,v)\in E} f_{sv}$.

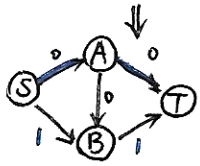
即整个网络中总流量的最大化.

显然这是个 LP 问题. 我们知道存在 O(Polynomial) 的算法来解该问题.

▷ Simplex Algorithms (单纯形类的算法) ("单纯形算法"是一类算法!)



(假设: $C_{uv}=1$)



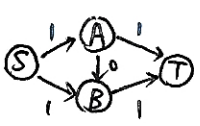
$f^{(1)}=\{f_{uv}^{(1)}\}_{(u,v)\in E}$. $f_{uv}^{(1)}=0$ 初始化

Pick a path from S to T. $S\to B\to T$. Maintaing restrictions.

However, $\Sigma f\uparrow$. Merge it.

$f^{(2)}$ 如左图.

Pick $S\to A\to T$. ……



$\Rightarrow f^{(3)}=2.$

无法更优. [END]

Consider graph $G^{t+1} = G^t$ with $w(u,v) = c_{uv} - f_{uv}^{(t)}$.

↑ capacity

if $w(u,v) = 0$ i.e. $c_{uv} = f_{uv}^{(t)}$. Remove the edge from G^{t+1}

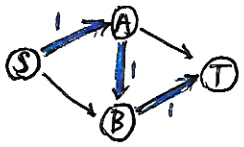
Then we consider G^{t+1}. Find a path from S to T on G^{t+1}

把 S→T 上的水管灌满.

问题从 G^t 上转移到 G^{t+1} 上.

.....

问题:



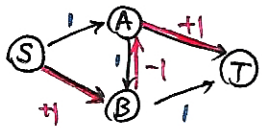
What if we pick S→A→B→T?

似乎无法达到最优解 2.

原始的单纯形算法无法产生所有的可行解. 不能保证最优.

原本的"只增加流量"限制了接近最优解.

每条边上flow

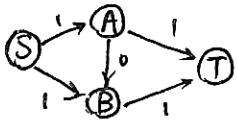


We allow the path to go in inverse direction of an edge.

Pick "S→B→A→T".

(允许减少每条边中flow流量)

仍满足"流量平衡".



该操作即:

可增可减流量

$$\begin{cases} f_{uv}^{(t+1)} \geqslant f_{uv}^{(t)} & \text{if. } f_{uv}^{(t)} < c_{uv} \\ f_{uv}^{(t+1)} \leqslant f_{uv}^{(t)} & \text{if. } f_{uv}^{(t)} = c_{uv} \end{cases}$$

→ 可以加一条反向的管道.

可减流量

为更好表述. 引入残量网络.

Def. The residual network

$f = \{f_{uv}\}$ (current flow) $G^f = (V^f, E^f)$

$V^f = V$. an edge $(u,v) \in E^f$ with capacity c_{uv}^f.

$$c_{uv}^f = \begin{cases} c_{uv} - f_{uv} & \text{if } (u,v) \in E,\ f_{uv} < c_{uv} \\ f_{vu} & \text{if } (v,u) \in E,\ f_{vu} > 0 \end{cases}$$

有 $v \to u$ 的流量时, ← 增加一条反向管道.

(减少流量使用该管道)

Algorithm. Modified Simplex Algorithm (Ford-Fulkson Algorithm)

IDEA: 1) Start from $f^{(1)} = \{f_{uv}^{(1)}\}_{(u,v)\in E}$. $f_{uv}^{(1)} = 0$.

2) In the t-th iteration, pick path from S to T on $G^{f^{(t)}}$.

Update: $f^{(t+1)} = \cdots\cdots$.

(Merge the path into $f^{(t)}$.)

Input: $G=(V,E,c)$ $c:\{c_{uv}\}_{(u,v)\in E}$

循环：至多 $|E|$ 条路径. — while: $O(|E|)$

(每次至少让一条边反向)

E-K: use BFS to find a path, $O(|E|)$ (minimum number of edges)

We call the path "augmenting path" 增广 路

❷ Initialize f with $f_e=0$ for any $e\in E$.

While ($\exists$ a path from S to T on G^f. let the path be P)

对 P 不加限制. 运行时间取决于 $c_{u,v}$

Augment (f, P);

end while

Return f;

最多只能把 P 中管道灌满. 即增加flow流量为 c^f 的最小值

Augment (f, P):

$b \leftarrow \min_{e\in P} c_e^f$;

for $e=(u,v)\in P$: $O(|n|)$

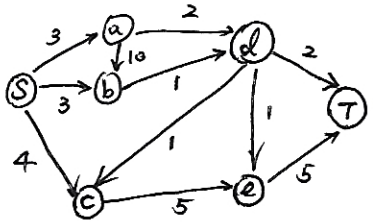
if $e=(u,v)\in E$ then $f_e = f_e + b$;

else $f_e = f_e - b$;

$O(m^2n)$ = $|E|\,|V|$.

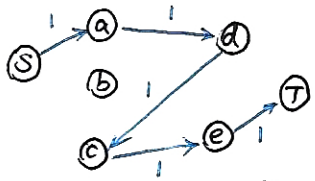
优化: Edmonds-Karp Algo.

e.g. Max Flow on



$f^{(0)}$: 全0 (不连通)

$f^{(1)}$:

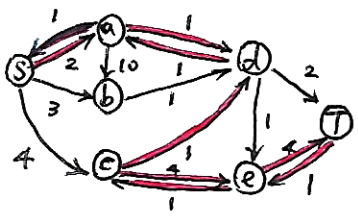


$G^{f^{(0)}}$: 原图.

Pick: $S\to a\to d\to c\to e\to T$

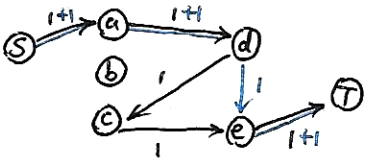
能增加的最大流量: 1 ($d\to c$ 满)

$G^{f^{(1)}}$:

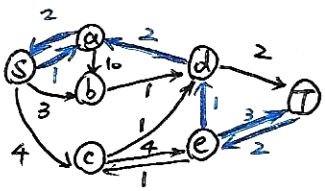


$f^{(2)}$: Pick $S\to a\to d\to e\to T$.

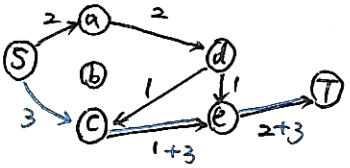
能增加的最大流量: 1 ($a\to d$ 满)



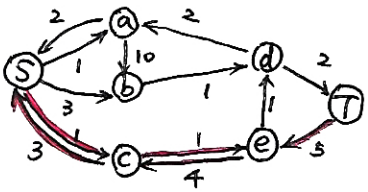
$G^{f^{(2)}}$:



$f^{(3)}$: Pick $S\to c\to e\to T$. 能增加的最大流: 3 ($e\to T$ 满)

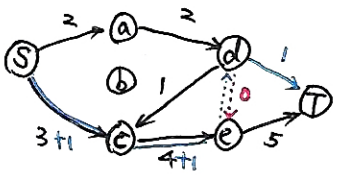


$G^{f^{(3)}}$:

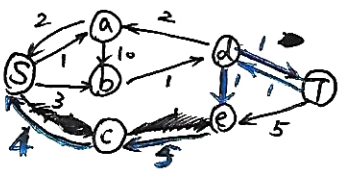


$f^{(4)}$: Pick $S\to c\to e\to d\to T$. 能增加的最大流: 1

$e\to d$: 使 $d\to e$ 流量减1

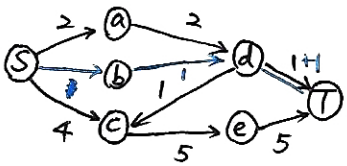


$G^{f^{(4)}}$

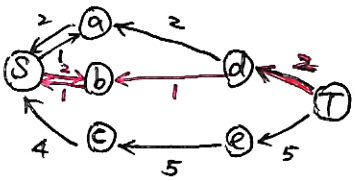


$f^{(5)}$: Pick $S\to b\to d\to T$. 能增加的最大流: 1

($b\to d$. $d\to T$ 满)



$G^{f^{(5)}}$



此时 $S\to T$ 无路可走.

[END]

Def. for (S,T): cut of G is a partition (L,R) of V

s.t. $\begin{cases} ① & L \cap R = \emptyset. \quad L \cup R = V \\ ② & S \in L. \quad T \in R \end{cases}$

capacity of a cut $(L,R) := \sum_{\substack{(u,v)\in E \\ u\in L,\, v\in R}} C_{uv}$

$\quad \big\Vert$

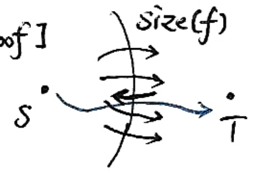
$\text{cap}(L,R)$

In fact, cut is the dual problem of flow. (See +2 Page)

Prop. $\forall$ (S,T)-cut (L,R). $\forall$ flow f, we have:

$$\text{cap}(L,R) \geq \text{size}(f)$$

[Proof]



Obvious. $\xRightarrow{\text{Proof}}$

$f(L,R) = \sum_{\substack{(u,v)\in E \\ u\in L,\, v\in R}} f_{uv}$.

$f(R,L) = \sum_{\substack{(u,v)\in E \\ u\in R,\, v\in L}} f_{uv}$.

We have [redacted]

$v(f) = \sum_{v:(S,v)\in E} f_{Sv} = \sum_{v:(v,T)\in E} f_{vT}$

$= f(L,R) - f(R,L)$

$\therefore v(f) \leq f(L,R)$

$= \sum_{\substack{(u,v)\in E \\ u\in L,\, v\in R}} f_{uv}$

$\leq \sum_{\substack{(u,v)\in E \\ u\in L,\, v\in R}} C_{uv}$

$= \text{Capacity}(L,R)$. $\square$

Prop. If find (L^*, R^*) s.t. $\text{capacity}(L^*, R^*) = \text{size}(f^{Algo})$

then $\text{capacity}(L^*, R^*)$ is optimal, f^{Algo} is optimal.

[Proof] Trivial. (Given the proposition above.)

We can give a visible expression of L^* and R^*.

$\begin{cases} L^* := \text{vertices reachable from } S \text{ in } G^{f^{Algo}}. \\ R^* := V \setminus L^*. \end{cases}$

[Proof] $\forall\, u \in L^*,\ v \in R^*$. we have $f_{uv} = C_{uv}$.

Otherwise: $v(f) \leq \sum_{\substack{u\in L^* \\ v\in R^*}} f_{uv} < \sum_{\substack{u\in L^* \\ v\in R^*}} C_{uv} = \text{Capacity}(L^*, R^*)$. Contradiction.

($f_{uv} < C_{uv}$ for some (u,v))

$\forall\, v \in L^*,\ u \in R^*$. we have $f_{uv} = 0$.

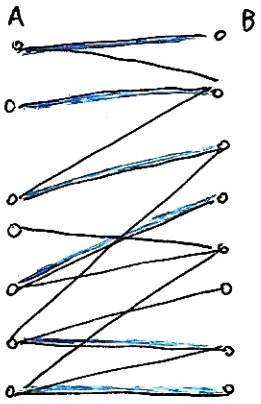
If $f_{uv} > 0$. $\Rightarrow$ on $G^{f^{Algo}}$. exists $v \to u$ ($L^* \to R^*$).

Strong duality:

$$\text{flow} \leq \text{Max flow} = \text{Min cut} \leq \text{cut}$$

$\qquad (v(f)) \qquad (\text{Capacity}(\text{cut}))$

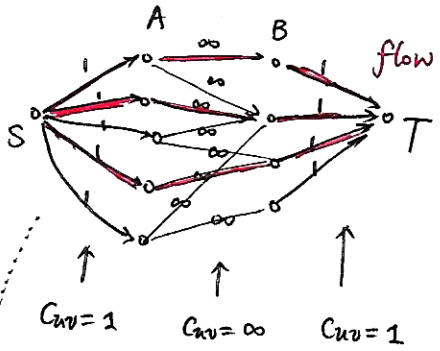
▷ Bipartite Graph



matching. → discrete mathematics

Max matching? ~ We convert the problem into a minimum cut/max flow

Define C_{uv}.



$C_{uv}=1$ $C_{uv}=\infty$ $C_{uv}=1$

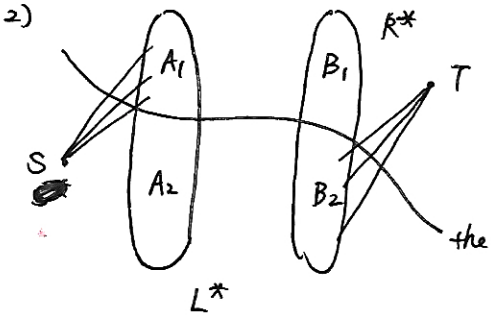
max matching

↓

max flow from $S \to T$.

1) the minimum cut on the graph is a minimum vertex cover. (任意边至少有一个端点在其中)

2)



no edge between A_2, B_1.

(Otherwise, capacity(cut) = ∞, is not a minimum cut)

$\Rightarrow A_2 \cup B_1$ is an independent set.

(任意两点间无边)

$A_1 \cup B_2 = V \setminus (A_2 \cup B_1)$ is another independent size.

capacity$(L^*, R^*) = |A_1| + |B_2|$ ~ $A_1 \cup B_2$ is a minimum independent set.

minimum minimum

=

max flow $(S \to T)$ = max matching

□

▷ Championship

MLB

		Win	Lose	Remain (未比)	A	B	C	D	E
					Remain games				
≤1	A	75	59	~~28~~		3	8	7	3
≤4	B	72	62	~~28~~	3			2	7	4
≤7	C	69	66	~~27~~	8	2			0	0
≤16	D	60	75	~~27~~	7	7	0		0
	E	49	86	~~27~~	3	4	0	0	

假设E全赢

+27 76 假设A.B.C.D与其它非A~E队伍全输.

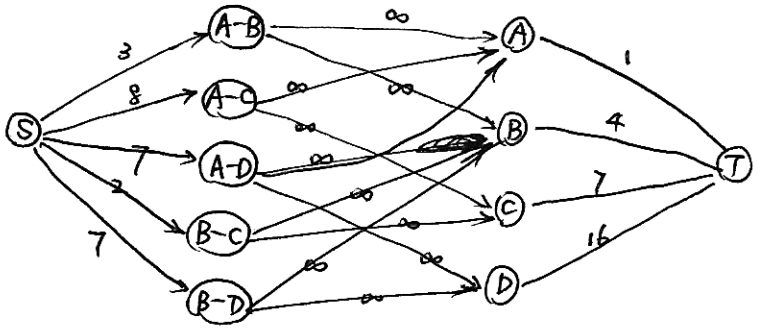
(略去与其它队伍no比赛)

Does E have the possibility to win the championship?

→ 分配A.B.C.D的比赛结果

使 $A_{min} \le 1$. $B_{min} \le 4$. $C_{min} \le 7$. $D_{min} \le 16$?

将该问题建模为网络流问题.



比赛　　结果(哪方赢)

if exists a max flow

s.t. $v(f) = 27$.

then E might win the champion.

(若存在最大流，则能分配出一个符合要求的结果 ⇒ E 可能夺冠.)

▷ Duality of Min-cut and Max-flow *

Max Flow (std. form)

$$\max \sum_{u:(s,u)\in E} f_{su}$$

s.t. $\forall u \in V\setminus\{s,t\}$, $\sum_{u:(v,u)\in E} f_{vu} - \sum_{w:(u,w)\in E} f_{wu} \le 0$

$\forall u \in V\setminus\{s,t\}$, $-\sum_{u:(v,u)\in E} f_{vu} + \sum_{w:(u,w)\in E} f_{wu} \le 0$

$\forall (u,v)\in E$, $f_{uv} \le c_{uv}$

$\forall (u,v)\in E$. $f_{uv} \ge 0$

▷ Analysis of Edmonds-Karp Algorithm.

$f_0 \to f_1 \to f_2 \to \cdots \to f_T = f^*$. E-K: the augmentation path is the shortest one in BFS.

$\forall u \in V$. $\delta_i(u) :=$ distance from s to u is G^{f_i}. (the residual graph)

↑ the depth of the BFS tree.

① $\delta_i(u)$ is non-decreasing in i. i.e. $\delta_{i+1}(u) \geq \delta_i(u)$.

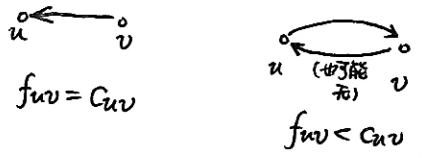
[Proof] Trivial. During the process. we might remove the edge on the path from s to u.

$G^{f_i} \to G^{f_{i+1}}$.

But we might add some other edges (the Inverse of some original edges).

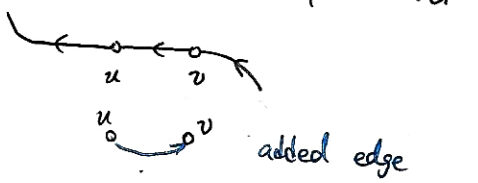
an edge exists $\begin{cases} (u,v) & c_{uv} > f_{uv} \leftarrow$ 如果前一次 $f_{uv} = c_{uv}$. 此时会加这条边. \\ (v,u) & f_{uv} > 0. \leftarrow \text{newly-added edges} \end{cases}$

CASE 01. G^{f_i} $G^{f_{i+1}}$

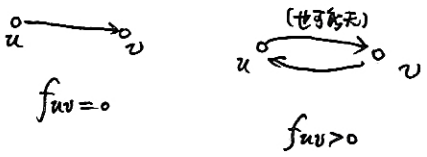


$f_{uv} = c_{uv}$ $f_{uv} < c_{uv}$

thus, we find shortest augment path on G^{f_i}

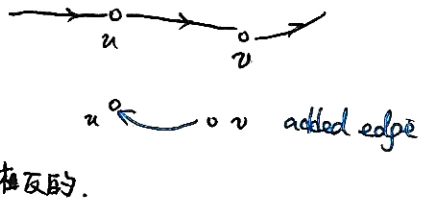


CASE 02. G^{f_i} $G^{f_{i+1}}$



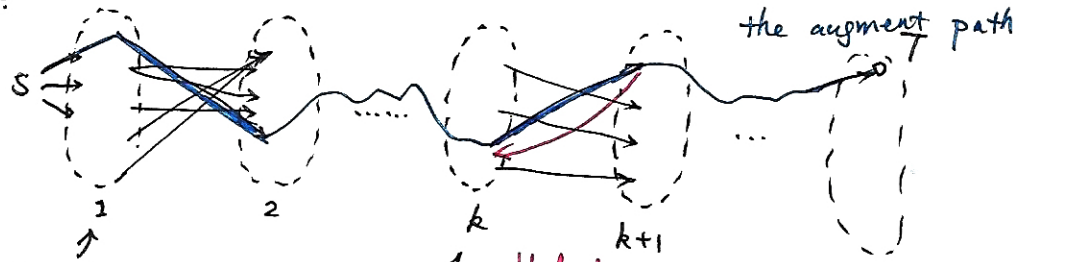
$f_{uv} = 0$ $f_{uv} > 0$

the shortest augment path on G^{f_i}



新引入的边始终与上一次残量图上增广路径的方向是相反的.

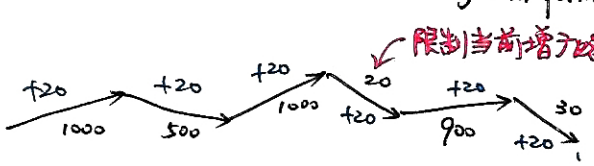
In BFS:



因此 $\delta_{i+1}(u) \geq \delta_i(u)$.

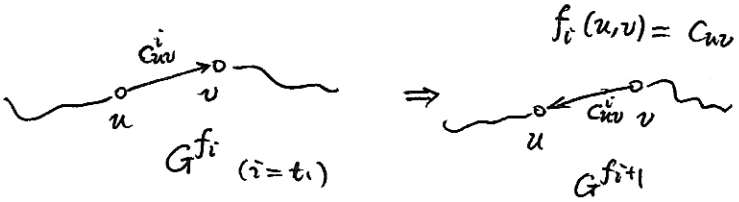
□

② critical edge: exists the shortest augmentation path. ($f^{t+1} \leftarrow f^t + \min_{e \in p} c_e^f$).



(限制了这一轮能增加的最大流量)

当一条边成为 critical edge 时



$f_i(u,v) = c_{uv}$!

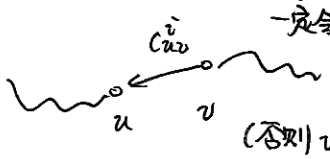
t_1 时刻：第1次成为 critical edge

$\delta_{t_1}(v) = \delta_{t_1}(u)+1$.

(在增广路上)

(u,v) 下一次成为 critical edge 之前某个时刻 $t_2^{[*]}$

一定会有：



(否则 $v \to u$ 不会反向)

$\delta_{t_2}(u) = \delta_{t_2}(v)+1$

$\geqslant \delta_{t_1}(v)+1 = \delta_{t_1}(u)+2$

(u,v) 第2次 critical 时刻 (t_3). 有：

$\delta_{t_3}(v) = \delta_{t_3}(u)+1 \geqslant \delta_{t_2}(u)+1 \geqslant \delta_{t_1}(u)+3$.

critical 两次后，δ 就要加3.

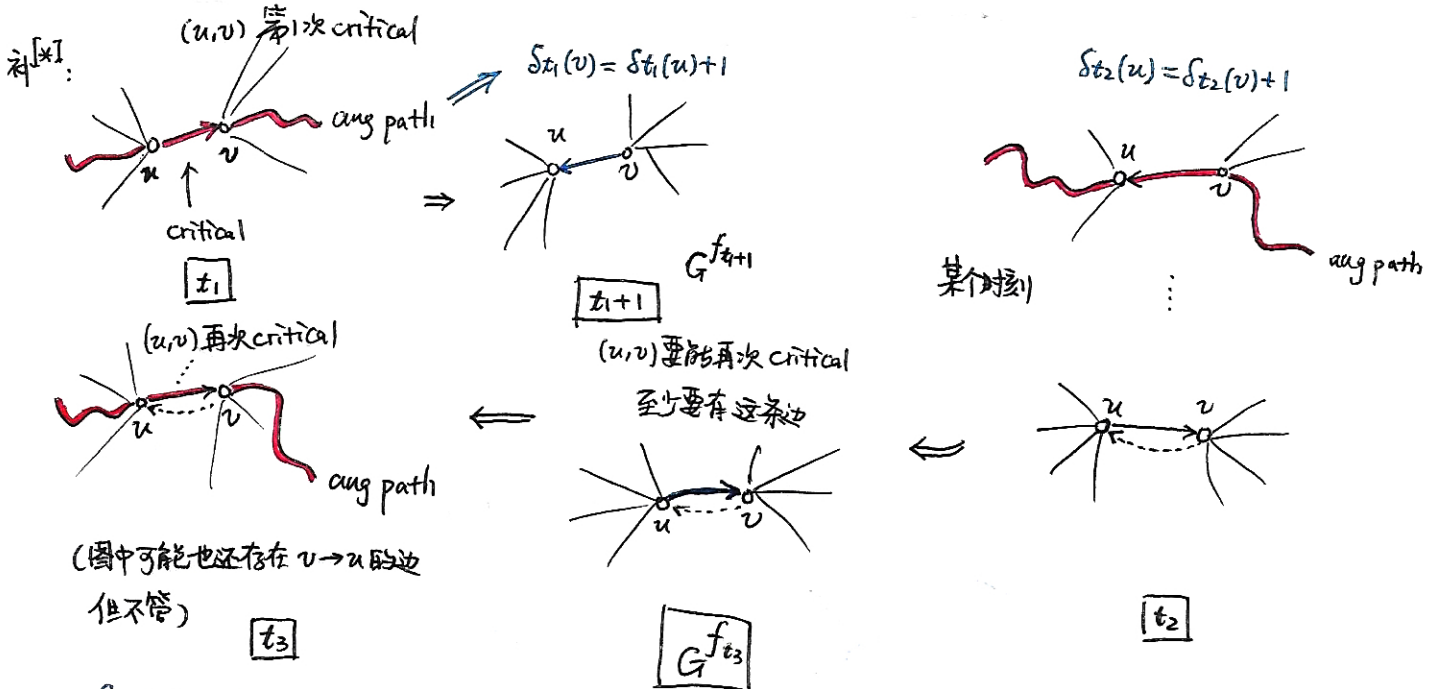
显然 $\delta_t(w) \leqslant n \quad (\forall w \in V) \Rightarrow$ #({u,v} becomes a critical edge) $\leqslant n$. (其实大概只是 $\frac{1}{3}n$ 左右)

③ 共 m 条边. 每条边最多 critical n 次. 每次找增广路要 $O(m)$ (BFS)

(在增广路中)

最多 $O(mn)$ 次找增广路 $\Rightarrow$ $O(m^2n)$

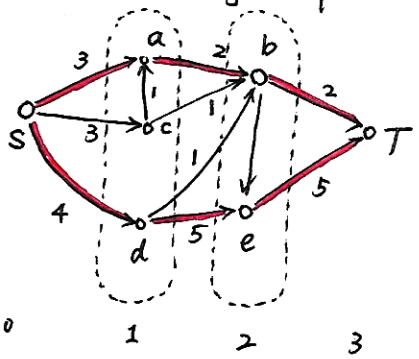
对[*]：



$\delta_{t_3}(u) = \delta_{t_3}(u)+1 \geqslant \delta_{t_2}(u)+1 = \delta_{t_2}(v)+2 \geqslant \delta_{t_1}(v)+2 = \delta_{t_1}(u)+3$

▷ 优化：每次可以多找几条 augment path 同时处理

e.g.



每次把所有 BFS 中 $S \to T$ 最短路 i.e. 路径作为增广路. 同时处理 Dinic's Algorithm

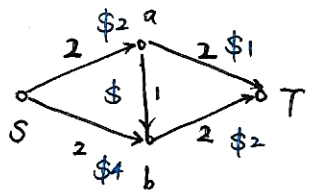
$O(mn^2)$

*但BFS直接找到的路径可能会重合

有其它具体 i.e. 操作

▷ Almost-Linear Max-flow Algorithm (2022)*

▷ Min-cost Max-flow



$\forall (u,v) \in E$. capacity: c_{uv}. expense: w_{uv} (per unit of water)

find a max flow with min cost $\sum_{(u,v)\in E} f_{uv} \cdot w_{uv}$.

(在所有最大流中找最小的成本) ⇝ 最小成本：最短路问题.

G_f: add (u,v) with weight $w(u,v)$ ~ 原本正向边

add (v,u) with weight $-w(u,v)$ if $f_{uv}>0$ (新增反向边)

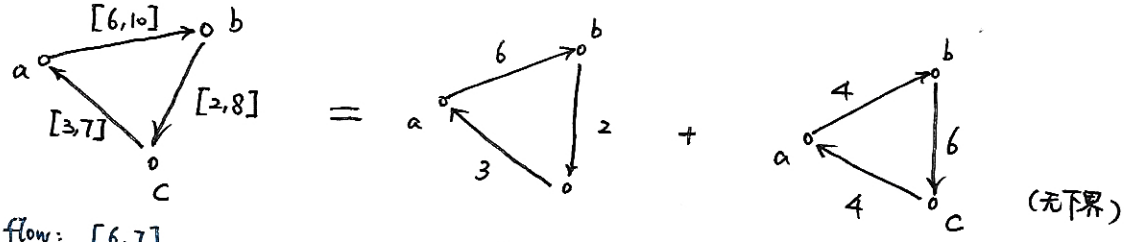
In E-K. we find the augmentation path using Bellman-Ford. w.r.t. $w(\cdot)$. (因为可能有负边存在)

(加权的最短增广路) — 每次找代价最小的增广路

[Proof of Correctness] 略. 比较复杂

▷ Max-flow with Bounds (带上下限的网络流)

1) 无源无汇



(无下界)

flow: [6,7]

f_{low} + f_1 =

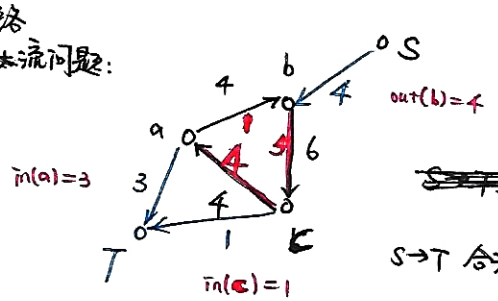
恒定的流

S.t. $In(a) = 3$

$Out(b) = 4$

$In(c) = 1$

网络转化为最大流问题:



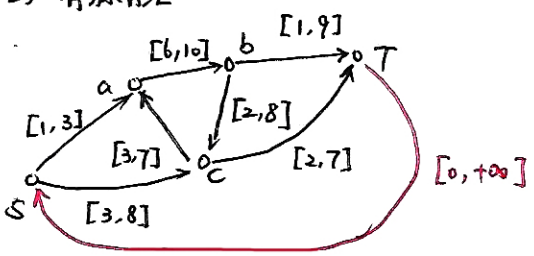
~~S→T 最大流即为答案~~ ~~原图上的最大流~~

S→T 合法的网络流 ~ 原图上的合法网络流

0, 4, 3 6, 6, 6

1, 5, 4 7, 7, 7.

2) 有源有汇



⇒ 转化为无源无汇问题

▷ Given $A \in \mathbb{R}^{m \times n}$

Find $B \in \mathbb{R}^{m \times n}$. $B_{ij} \in [L, R]$ ⇝ $\min\left(\max_i \left|\sum_{j=1}^{n} A_{ij} - B_{ij}\right|, \max_j \left|\sum_{i=1}^{n} A_{ij} - B_{ij}\right|\right)$

转化为网络流.

1. • 二分枚举答案 x. (猜一个答案). 问: 是否存在 B s.t. $\max_i \left|\sum_{j=1}^{n} A_{ij} - B_{ij}\right| \le x$

$\max_j \left|\sum_{i=1}^{n} A_{ij} - B_{ij}\right| \le x$.

存在 → $x^{(+1)} \in [x^{(\text{ho})}+1, \text{upper}]$

不存在 → $x^{(+1)} \in [\text{lower}, x^{(+0)}-1]$.

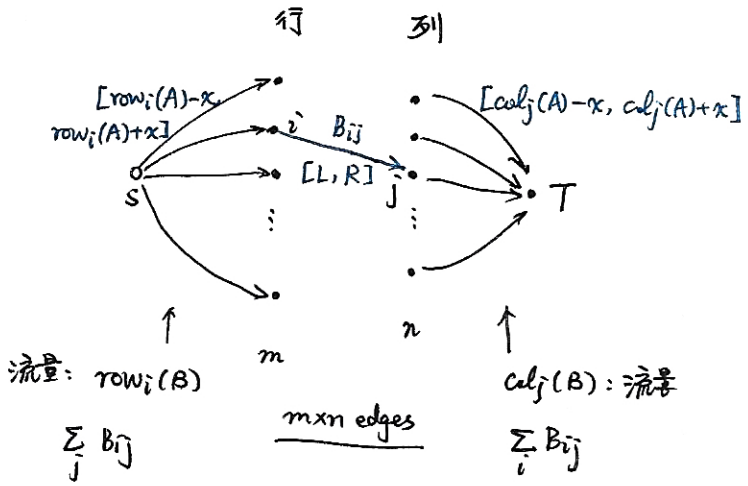
2. 判断 x 可不可行.

Def. $\text{row}_i(M)$: $\sum_{j=1}^{n} M_{ij}$ (第 i 行和). $\text{col}_j(M)$: $\sum_{i=1}^{n} M_{ij}$ (第 j 列和)

则 x 可行 ⇔ $\text{row}_i(A) - x \le \text{row}_i(B) \le \text{row}_i(A) + x \quad \forall i$

$\text{col}_j(A) - x \le \text{col}_j(B) \le \text{col}_j(A) - x \quad \forall j$

3. 转化为网络流:



⇒ 有源.汇的带上.下界网络流问题

• Zero-Sum Game 零和博弈

▷ rock, paper, scissors

	r	p	s
r	0	-1	1
p	1	0	-1
s	-1	1	0

row player 得分按行. —— maximize

column player 按列 —— minimize

column player: $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ (r p s) row 各种策略都一样

e.g. r: $0 \times \frac{1}{3} + 1 \times \frac{1}{3} - 1 \times \frac{1}{3} = 0$

column player: $(\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$ → 一直为 r 最优

r p s

row player: (x_1, x_2, x_3) column 知晓对方策略. → 计算各种出拳胜率.

AIM ⇒ choose (x_1, x_2, x_3) s.t. $\max \min \begin{cases} M_{11}x_1 + M_{21}x_2 + M_{31}x_3 \\ M_{12}x_1 + M_{22}x_2 + M_{32}x_3 \\ M_{13}x_1 + M_{23}x_2 + M_{33}x_3 \end{cases}$

最佳: $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

c希望r得分尽可能少. 选择对方得分最少的策略.

是否先选无妨啊. (平和 ⇒ r加1分, c减1分)

▷

	A	B
①	3	-1
②	-2	1

① Row 先选策略. column 后选. 且知道 row 的策略.

$\max \min \{3x_1 - 2x_2, -x_1 + x_2\} =: t$ (x_1, x_2) ① ②

$\max$: 我知道 column 会这么做. 所以我尽可能优化 x_1, x_2 选择使我得分更高.

$\min$: column 想让 row 得分尽可能少

$\Leftrightarrow$ $\max t$ s.t. $3x_1 - 2x_2 \geq t$, $-x_1 + x_2 \geq t$, $x_1, x_2 \geq 0$, $x_1 + x_2 = 1$ (LP1)

② column 先选. row 后选且知道 column 的策略

$(y_1 \quad y_2)$ A B

$\min \max \{3y_1 - y_2, -2y_1 + y_2\} =: w$

$\min$: column 想让我得分尽可能小.

$\max$: 我会让自己得分尽可能大

$\Leftrightarrow$ $\min w$ s.t. $3y_1 - y_2 \leq w$, $-2y_1 + y_2 \leq w$, $y_1, y_2 \geq 0$ $y_1 + y_2 = 1$ (LP2)

⇒ LP2 is the dual of LP2. Strong duality: $\boxed{OPT(LP1) = OPT(LP2)}$

Thm. Minimax Thm. (Von Neumann)

$$\min_y \max_x A_{ij} x_i y_j = \max_x \min_y A_{ij} x_i y_j$$ (对冲和博弈)

Computations : NP and P

NP: 暂时还没有找到多项式时间算法

e.g. CNF-SAT Problem (Satisfiability)

CNF: $\phi = (x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x})$

n variables, m clauses $\rightsquigarrow$ Algorithm: $O(1.33^n)$ (NP)

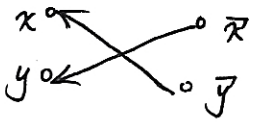
非多项式时间

Restricted Version: 2-SAT

$\phi = \bigwedge_{i=1}^{m} C_i \quad |C_i| = 2$ —— O(Polynomial)

$C_i = x \vee y$. ← x 和 y 至少有1个 True

$\Leftrightarrow \bar{x} \to y \Leftrightarrow \bar{y} \to x$ 建图:

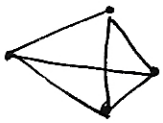


~~Variable~~.

如果∃ variable i s.t. i 和 $\bar{i}$ 在同一个强连通块中. → UNSAT. 否则 SAT.

e.g. TSP (Travelling Salesman Problem)

$G = (V, E)$ $d: \binom{V}{2} \to \mathbb{R}$. cost of edge 找一个 cost 之和最小的环.



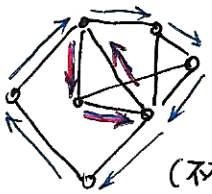
In DP: $O(n \cdot 2^n)$ —— NP (Non-Polynomial)

v.s. MST (Minimum Spanning Tree) 找一个 weight 之和最小的树.

Kruskal / Prim — O(Polynomial) $(O(|V|+|E|))$ —— P

e.g. Hamiltonian Circuit & Eulerian Circuit

H $G = (V, E)$ whether exists a circuit visiting each vertex for only once. NP

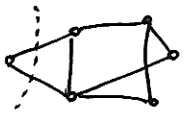


(不存在)

E whether exists a circuit visiting each edge for only once

O(Polynomial) $O(|E|)$

e.g. min-cut : #(edges across the cut) to be minimum.



不给定源和汇？ → 枚举 (s,t)，每个算一次 min-cut. O(Polynomial)

Find $C_1, C_2 \subseteq V$. s.t. $C_1 \cap C_2 = \emptyset$. $C_1 \cup C_2 = V$

$\min |E(C_1, C_2)|$

balanced min-cut : find $C_1, C_2 \subseteq V$ s.t. $C_1 \cap C_2 = \emptyset$. $C_1 \cup C_2 = V$

$|C_1| > n/3$. $|C_2| > n/3$

$\min |E(C_1, C_2)|$ O(NP)

e.g. Linear Programming $\min c^T x$ s.t. $Ax \le b$. $x \ge 0$. O(Polynomial)

Interior Point Method / Ellipsoid Method

Integer Programming $\min c^T x$ s.t. $Ax \le b$. $x \in \{0,1\}$ O(NP)

* → ZOE. (Zero-One Equation)

for a given matrix A. ($\forall ij$ $a_{ij} \in \{0,1\}$.) whether $\exists x$ s.t. $Ax = 1$. $x_i \in \{0,1\}$

e.g. max independent set / min vertex cover / max clique on graph: O(NP)

↑ 任意两点无边的最大点集

↑ 任意边至少有1个端点在集合中

↑ 任意两点均有边的最大点集

on tree/bipartite graph : O(P)

e.g. Shortest Path from s to t. O(Polynomial)

Longest Path from s to t O(NP)

Simple (每个点至多被访问一次)

e.g. Knapsack. 背包问题 (NP)

n个item. 占空间 a_i 价值 w_i 背包容量 W → $O(W \cdot \text{Polynomial}(n))$

W的输入用二进制. b位 → $O(2^b \cdot \text{Poly})$

不知道W时. Non-Poly...

$w_i = 0$ 某个特别版本

子集和问题 n个数 $w_1 \sim w_n$. Pick $I \subseteq [n]$. s.t. $\sum_{k \in I} w_k = W$ O(NP)

e.g. Matching

网络流 O(Poly)

3D-Matching

变难 变简单

一个房间：boy, girl, pet

最多n个景？ O(NP)

对问题回要求稍作修改. 松弛或特例. → NP/P.

上述问题均无法证明不存在 Polynomial 算法. 但可以比较问题 in 难度

严格定义需要给出图灵机的定义. 这里并不完全展开.

• Theory of Computational Complexity

To simplify the problem, we only consider decision problems. (判定性问题)

Def. Problem $f: \{0,1\}^* \to \{0,1\}$

Decision

e.g. SAT. $\phi \to \mathbb{I}[\phi \text{ is satisfiable}]$

Def. An algorithm $\mathcal{A}$ computes f

iff. $\forall x \in \{0,1\}^*$. $\mathcal{A}(x) = f(x)$. where $\mathcal{A}$ is a Turing Machine.

Def. $P :=$ collection of decision problems computable by a polynomial-time algorithm.

$= \{ f: \{0,1\}^* \to \{0,1\} \mid \exists \mathcal{A}.\ \mathcal{A}$ computes f; $\forall x.\ \mathcal{A}(x)$ terminates in $|x|^{O(1)}$ time. $\}$

Polynomial-time Computable Problems

Def. $NP := \{ f: \{0,1\}^* \to \{0,1\} \mid \exists$ non-deterministic Turing Machine $\mathcal{A}$ s.t. (非确定性 图灵机)

Non-deterministic Polynomial-time

$\mathcal{A}$ computes f; $\forall x.\ \mathcal{A}(x)$ terminates in $|x|^{O(1)}$ time. $\}$

$:=$ Polynomial-time Verifiable Problems (等价定义)

* Poly-Verifiable 一定能在指数时间内解.

$f \in NP$ iff. $\exists$ algorithm $V: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ s.t. ($\exists$ polynomial p)

1) $\forall x \in \{0,1\}^*$ if $f(x)=1$. $\exists y$: $|y| = |x|^{O(1)}$ ($y \le p(|x|)$) and $V(x,y)=1$;

2) $\forall x$ if $f(x)=0$. $\forall y$. $V(x,y)=0$ ($|y| = |x|^{O(1)}$. ($y \le p(|x|)$))

3) $V(x,y)$ terminates in $|x|^{O(1)}$.

V: the verifier. 类似于看相答案看看对不对. 定理证明器 (给我1个y. 我看看y对不对. 符不符合x)

e.g. $x \leftrightarrow$ math statement. $y \leftrightarrow$ a proof of x.

V: verifier of the proof.

$f(x)=1$ iff. x is correct.

一个比对至少要(长度)时间.

非Polynomial则一定不满足3).

e.g. 条件1): 看看存不存在1个证明y. 存在1个证明. (有限. 且实际上是Polynomial长度)

对正确的x

条件2): 本身错的x. 不存在证明y.

$V(x,y)$: $\mathbb{I}[y \text{ is a proof of } x]$.

e.g. For SAT Problem. $x: \phi$ $y: \sigma$ (assignment) $\sigma: V \to \{True, False\}$ (variables)

$f(x) = \mathbb{1}[x \text{ is SAT}]$. $V(x, y) = \begin{cases} 1 & \text{if } \sigma \vDash \phi \\ 0 & \text{if } \sigma \nvDash \phi \end{cases}$

ϕ, σ

$\therefore$ SAT $\in$ NP.

e.g. TSP Problem

$f(\underbrace{G, k}_{x}) = \begin{cases} 1 & \exists \text{ a tour with cost} \le k \\ 0 & \text{otherwise} \end{cases}$ y: tour 的证的编码

$\therefore$ TSP $\in$ NP

$V(x, y) = \mathbb{1}[$ the cost of tour y on $G \le k]$.

Thm. $P \subseteq NP$. (A 本身就是个 verifier).

Problem: $NP \subseteq P$? ← 无法证明. 也无法证伪.

$NP = P$?

~~[redacted]~~

Assumption. $NP \neq P$.

NP-Complete: 最"难"的 NP 问题

↑ to be defined

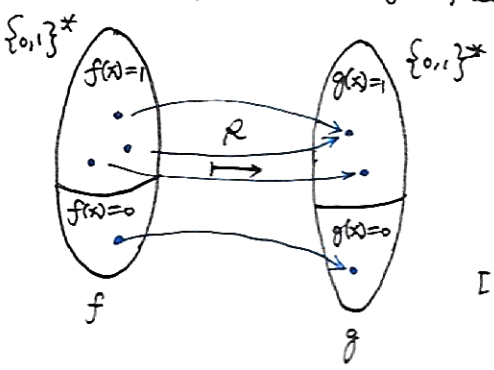
▷ Reduction 归约

Def. (Karp-) Reduction. problem $f, g: \{0,1\}^* \to \{0,1\}$.

$f \le_K g$ iff. $\exists$ Polynomial-time Algorithm $\mathcal{R}: \{0,1\}^* \to \{0,1\}^*$ (many-to-one 允许不是单射)

s.t. $\forall x \in \{0,1\}^*$. $f(x) = 1 \Leftrightarrow g(\mathcal{R}(x)) = 1$.

直观理解: g 比 f 难. f 的难度 $\le g$ 的难度. (若有算法能解决 g. 那它一定能解决 f.)



Thm. If g can be solved in polynomial time by $\mathcal{B}$.

$\Rightarrow f$ can be solved in polynomial time.

[Proof] The algorithm $\mathcal{A}$ to solve f:

$\mathcal{A}(x)$: return $\mathcal{B}(\mathcal{R}(x))$

Qed. □

Def. NP-Complete.

$f: \{0,1\}^* \to \{0,1\}$ is NP-Complete iff.

1) f is NP.

2) $\forall g \in NP$. $g \le_K f$.

← ① 存在性?

② 一定在 $f \le_K g$ 不成立时有 $g \le_K f$ 吗.

$f, g \in$ NP-complete $\Rightarrow f \le_K g$ and $g \le_K f$.

Thm. (Cook-Levin Theorem)

SAT is NP-Complete. → The existence of NP-Complete.

[redacted]

[Remark] So far we discussed on only decision problems. ← Karp-Reduction is defined on reduction.

* If f, g are not necessarily decision problem,

$$f \leq_T g \qquad \text{(Turing / Cook-) Reduction}$$

:= " If one can solve g in Polynomial Time, then he or she can solve f in Polynomial Time."

Obvious $f \leq_K g \Rightarrow f \leq_T g$.

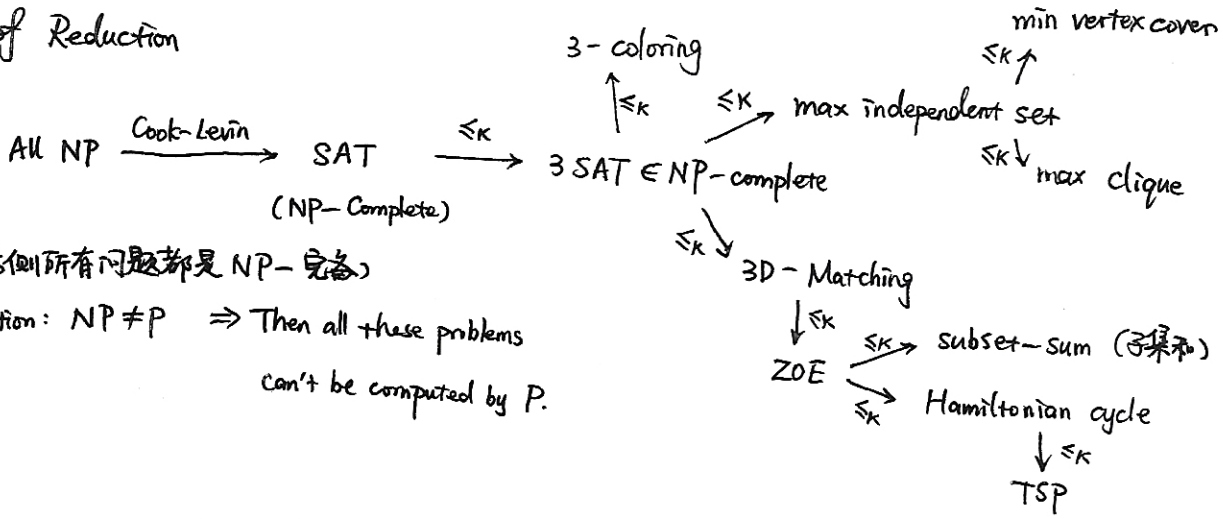
Def. A problem $g: \{0,1\}^* \to \{0,1\}^*$ is *NP-hard* if

$\exists f$. 1) f is NP-complete, 2) $f \leq_T g$.

NP-complete ⇒ NP-hard ; NP-hard $\not\Rightarrow$ NP-complete

(e.g. 停机问题)

▷ Web of Reduction



(图中右侧所有问题都是 NP-完备)

Assumption: NP ≠ P ⇒ Then all these problems can't be computed by P.

Informal

[Proof of C-L Thm.] 需要严格定义图灵机.

只需构造 $R: x \mapsto \phi_x$ (SAT) $\quad f(x)=1$ iff. ϕ_x is SAT.

$x \in NP \Rightarrow \exists$ Verifier $V(x,y)$. Polynomial-time verifiable

Turing Machine

图灵机当前状态 ~ snapshot (指令纸带、草稿纸带的指针位置及对应值)

Turing Machine 每走一步记录1个 snapshot ← snapshot 的可能性只有常数多种

" $S_t \to S_{t+1}$ 是否合法" 可以用 逻辑公式表述 ↘

Verifier 能走到终止 ⇔ " $S_t \to S_{t+1}$" 每一步都合法 ⇔ 那个很长的公式是 SAT.

□ SAT $\leq_K$ 3-SAT. ($\phi = \bigwedge_i c_i$. $|c_i| = 3$) $\Rightarrow$ 3-SAT is NP-complete

[Proof] $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m \mapsto \psi = c_1' \wedge c_2' \wedge \cdots \wedge c_n'$

s.t. ϕ is SAT $\Leftrightarrow$ ψ is SAT. $|c_i'| \leq 3 \quad \forall i \in [n]$.

易证3-SAT是NP的. (y: σ.)

① $|c_k| \leq 3$ → 保留. $c_k' = c_k$.

② $|c_k| > 3$. $c_k = a_1 \vee a_2 \vee \cdots \vee a_q$

introduce new variable $y_1, \cdots y_{q-3}$.

replace c_k by "$(a_1 \vee a_2 \vee y_1) \wedge (\overline{y_1} \vee a_3 \vee y_2) \wedge (\overline{y_2} \vee a_4 \vee y_3) \wedge \cdots$
$(\overline{y_{q-4}} \vee a_{q-2} \vee y_{q-3}) \wedge (\overline{y_{q-3}} \vee a_{q-1} \vee a_q)$" $\Rightarrow$ 3SAT.

* 这个在离散数学讲过 即3SAT算法

Prove that ϕ is SAT $\Leftrightarrow$ ψ is SAT.

$\exists \sigma$ s.t. ϕ is SAT. for c_k. assume a_i = True in σ. (至少肯定有1个是True)

σ': y_j = True ($j \leq i-2$). y_j = False ($j \geqslant i-2$)

e.g. a_4 = True. $(a_1 \vee a_2 \vee y_1) \wedge (\overline{y_1} \vee a_3 \vee y_2) \wedge (\overline{y_2} \vee a_4 \vee y_3) \wedge \cdots (\overline{y_{q-4}} \vee a_{q-2} \vee y_{q-3}) \wedge (\overline{y_{q-3}} \vee \cdots)$

True True True False $\overline{y_3}$ True True True

$\exists \sigma'$ s.t. ψ is SAT exist i s.t. a_i = True in σ'. $\Rightarrow \sigma$. ✓

(otherwise: y_1 = True $\Rightarrow y_2$ = True $\Rightarrow \cdots \Rightarrow y_{q-3}$ = True $\Rightarrow$ 最后1个clause:
(a_i = False, $\forall i$)
False ∨ False ∨ False = F. ×)

SAT → 3-SAT (某一类特殊的SAT问题) 难度并没有下降.

Qed. □

□ k-Independent Set Problem is NP-complete.

Given Graph G and number k. Does G contain an independent set with size $\geq k$?

① k-Independent Set Prob. is NP

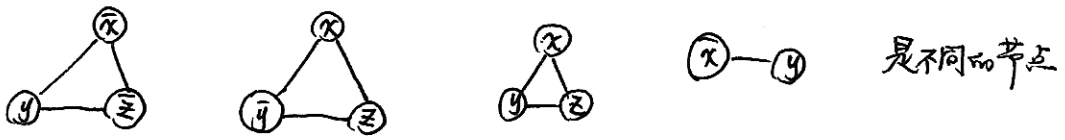
[Proof] y: an independent set with size $\geq k$ on G. ~ $V(x, y) = 1$. (G, k)

可以在Polynomial time里确定是否是独立集. 及size $\geq^{?} k$.

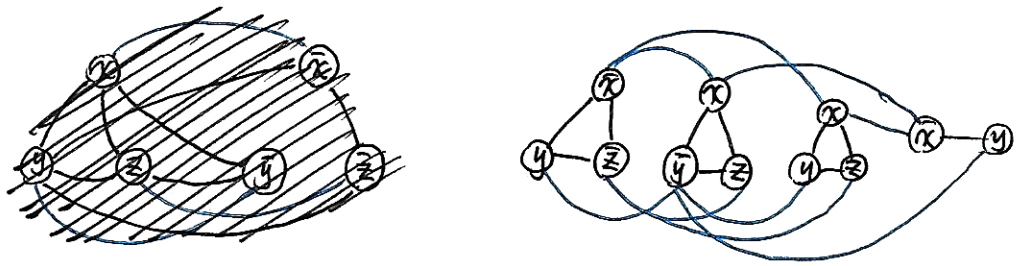
② 3-SAT $\leq_K$ k-Independent Set Problem

[Proof] 3-SAT: 实际上要求每个clause里选一个 literal 出来让它为True. ~ Independent Set.
至少
同时每个变量不能又True又False.

例子. $\phi = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee y)$



同时. x 与 $\bar{x}$ 不可能同时为真 ~ x 与 $\bar{x}$. y 与 $\bar{y}$. z 与 $\bar{z}$ 有连线 (连线…… 两端不能同时取)



$\psi_1 = c_1 \wedge c_2 \wedge \cdots c_m$. 对每个 $c_k = \hat{a}_i \vee \hat{a}_j \vee \hat{a}_k$ ($\hat{a}_i, \hat{a}_j, \hat{a}_k \in \{a_1, \cdots a_n, \bar{a}_1, \cdots \bar{a}_n\}$)

创建3个节点. 在 $\hat{a}_i$. $\hat{a}_j$. $\hat{a}_k$ 三个节点之间连边.

在每个变量 a 与 $\bar{a}$ 之间连边 ⇒ 得到 G_ψ

$\psi \in 3SAT$ ⇔ G_ψ contains an independent set with size $\geq m$.

⇒: 能SAT的 assignment ↑ ⇝ independent set (size = m) 的选点 . (每个三角形里必定有至少一个是 True. 就选它) (有个的话就选1个)

~~是不会有某个时两个literal为true? ⇒ 就里选一个点~~

⇐: independent set 的选点 (size ≥ m) ⇝ assigment ↑ (被选的点为True).

一定不会有 $a = \bar{a} =$ True (中间有连边. 不可能) 其它变量随意

Qed. □

* k-Ind. Set ⇒ SAT? ① Cook-Levin Thm.

② 连边 ⇒ 逻辑运算公式. SAT. (反正这个证明不用双向证明) 上面

□ k-Clique is NP-Complete.

(G, k)
$\exists S \subseteq V.\ |S| \geq k$ s.t. $\forall u, v \in S.\ u \neq v,\ \{u, v\} \in E.$

① k-Clique is NP. [Proof] y: a clique with size $\geq k$ on G. $x = (G, k)$

$V(x, y) = 1$ only if ... ✓

~~k-Clique~~

② k-Independent Set $\leq_K$ k-Clique

$(G, k) \to (\bar{G}, k)$ $\bar{G} = (V, \bar{E})$. $\bar{E} = V \times V \setminus \{(u, u) \mid \forall u \in V\} \setminus E$.

k-independent set on $\bar{G}$

⇔ k-clique on G.

Qed. □

□ k- Vertex Cover is NP- Complete. | $\exists$ vertex cover S. $|S| \le k$

① NP. ② k- Independent Set $\le_K$ k-Vertex Cover: [redacted]

S is an independent set $\Leftrightarrow$ $V \setminus S$ is a vertex cover.

$\exists$ k- Independent set on G $\Leftrightarrow$ $\exists (|V|-k)$- vertex cover on G. Qed. □

□ 3SAT* Problem, 3-SAT → 3-SAT*：某一类特殊化为 3-SAT，但难度并未下降（下证）

$\psi \in$ 3-CNF*

$\forall$ variable $x \in Variable(\psi)$.

限制每个变量出现的次数

$$\begin{cases} \#(x) + \#(\bar{x}) \le 3 \\ \#(x) \le 2 \\ \#(\bar{x}) \le 2 \end{cases}$$

NP Problem.

(y: assignment σ)

3-SAT $\le_K$ 3-SAT*.

[Proof] if a variable x appear k times. $k > 3$

Introduce $x_1, \cdots x_k$. Replace the x's first appearance with x_1, second with x_2, ...

e.g. $((x) \vee \cdots) \wedge (\overline{(x)} \vee \cdots) \wedge \cdots \wedge (\overline{(x)} \vee \cdots) \wedge ((x) \vee \cdots)$

make x_1, x_2, x_3, x_4

Then conjunction with some clause restricting $x_1, \cdots x_k$ with $x_1 = x_2 = \cdots = x_k$.

(3CNF* clause)

"$x_1 = x_2 = \cdots = x_k$" $\Leftrightarrow$ $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \cdots \wedge (x_{k-1} \to x_k)$

$\Leftrightarrow$ $(\bar{x_1} \vee x_2) \wedge (\bar{x_2} \vee x_3) \wedge \cdots \wedge (\bar{x_{k-1}} \vee x_k)$ ← 补上这些 clause

Then we convert a 3-CNF into a 3-CNF*. $\Rightarrow$ 3-SAT $\le_K$ 3-SAT*.

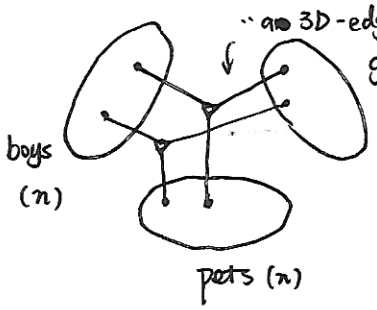
$\sigma \rightleftharpoons \sigma^*$

(3-CNF $\Leftrightarrow$ 3-CNF*)

Thus, 3-SAT* is also a NP-complete problem.

Qed. □

□ 3D- Matching is NP- Complete.



Whether we can choose n 3D-edges (a matching) s.t.

for any $v_1 \in$ Boys ∪ Girls ∪ Pets. exists exactly one 3D-edge in the matching s.t. v_1 is adjacent to the edge.

(每个点都刚好有1个和另两类中两个点 in matching)

We use  to represent 3D-edge. / hyper-edge.

① 3D-Matching is NP. y: 某一种选法. $V(x,y)$: 看看 y 是不是 x 上的 3D-Matching.
(在Polynomial 时间内可验证)

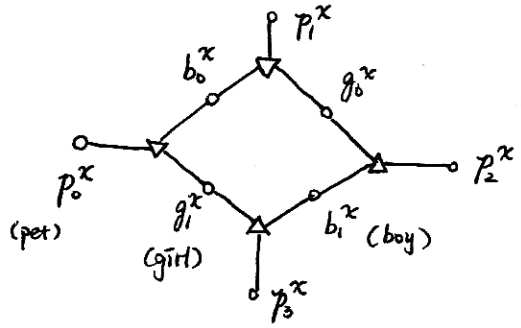
② $3\text{-SAT}^* \le_K$ 3D-Matching

$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, $V = \{x_1, x_2, \cdots x_n\}$ $\mapsto$ G_ϕ 3D-Matching.

一般: 找方法模拟 3-SAT^* 变量. 然后用 3D-Matching 表示 clauses.

(1) Variable Gadgets

a variable x $\mapsto$



Possible Matching: (p_0^x, b_0^x, g_1^x), (p_2^x, b_1^x, g_0^x) | (p_1^x, b_0^x, g_0^x), (p_3^x, b_1^x, g_1^x) ~~(p_3^x, b_0^x, g_1^x)~~.

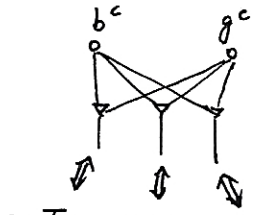
2 possible matching.

"x= True" (p_1^x, p_3^x: available) "x= False" (p_0^x, p_2^x: available)

(2) Clause Representation

a clause $c = x \vee \bar{y} \vee \bar{z}$.

"at least one of "x", "$\bar{y}$", "$\bar{z}$" is true" $\Leftrightarrow$



x= True $\bar{y}$= True $\bar{z}$= True
(y= False) (z= False)

b^c and g^c will at least choose a hyperedge. (in the matching)

x= True $\rightarrow$ p_0^x and p_2^x is already matched in x-gadget.
We connect b^c, g^c to p_1^x or p_3^x.

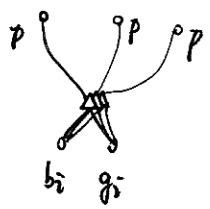
$3\text{-SAT}^* \Rightarrow$ "x" appears ≤ 2 times, "$\bar{x}$" appears ≤ 2 times. $\rightarrow$ 所以两个空余的 p 够用.

What if "x" appears only once? $\sim$ might leave some pets unmatched.

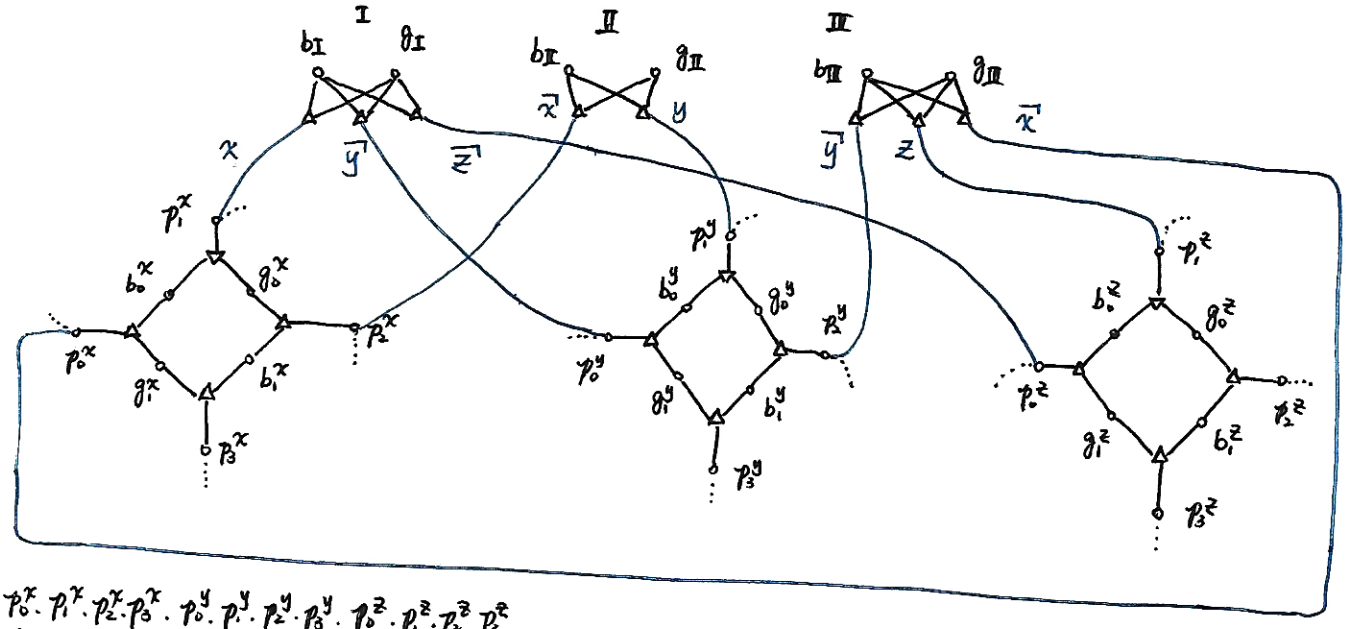
(3) Unmatched Dealing

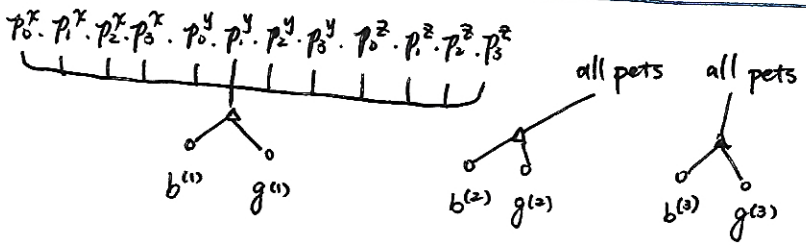
#(unmatched pets) is a known number. $\rightarrow$ Introduce #(unmatched pets) boys and girls.
connect with all pets.

$2n-m$ (n: clause. m: variables)



e.g. $\phi = \underbrace{(x \vee \bar{y} \vee \bar{z})}_{\text{I}} \wedge \underbrace{(\bar{x} \vee y)}_{\text{II}} \wedge \underbrace{(\bar{y} \vee z \vee \bar{x})}_{\text{III}}$ 3-CNF*.



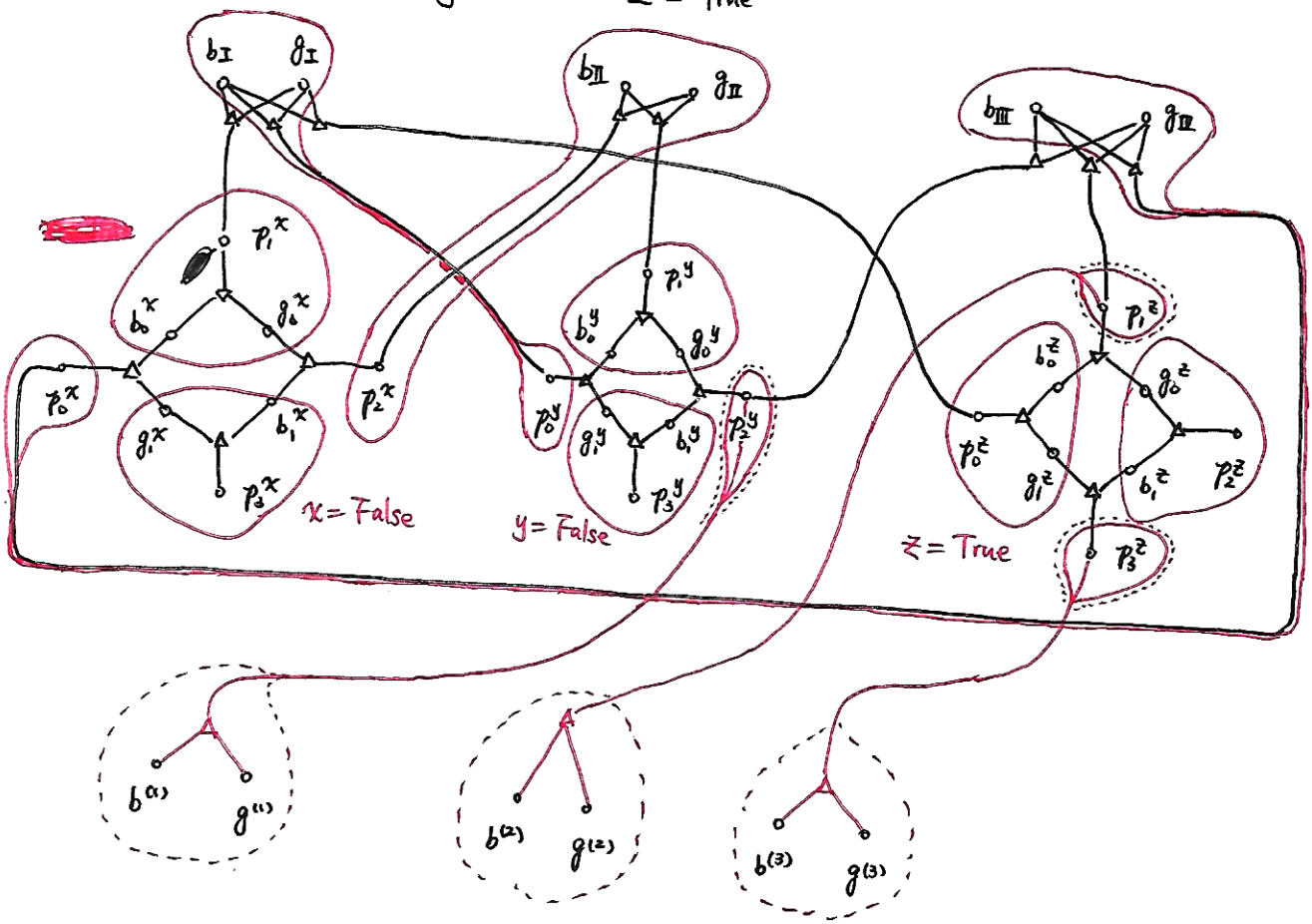


$\#(\text{clause}) = 3 =: n$

$\#(\text{variables}) = 3 =: m$

$2n - m = 3$

Assignment: x = False. y = False. z = True.



assignment $\rightleftarrows$ 3D-matching exists on the problem we construct

Thus, 3D-Matching is NP-complete.

Qed. □

□ ZOE is NP-complete.

$A \in \{0,1\}^{m \times n}$. whether exists $x \in \{0,1\}^n$. s.t. $Ax = \mathbb{1}$

① ZOE is NP. [Proof]: y : given such x. "x": A. $V(x,y)$: $Ax \overset{?}{=} \mathbb{1}$ (A, x)

② ZOE is NP-complete.

3D-Matching $\leq_K$ ZOE.

Hyperedge: Tuples $(b_{i_1}, g_{j_1}, p_{k_1})$, $(b_{i_2}, g_{j_2}, p_{k_2})$, ⋮, $(b_{i_m}, g_{j_m}, p_{k_m})$

$\forall \ell = 1, 2, \dots, m$. $x_\ell = \mathbb{1}$ [the ℓ-th tuple is chosen in the 3D-matching]

$x = (x_1, x_2, \dots x_m)^T$

$\forall$ boy i: $\sum_{\ell:\ \text{boy } i \text{ is in the } \ell\text{-th hyperedge}} x_\ell = 1$.

$\forall$ girl j: $\sum_{\ell:\ \text{girl } j \text{ is in the } \ell\text{-th hyperedge}} x_\ell = 1$.

$\forall$ pet k: $\sum_{\ell:\ \text{pet } k \text{ is in the } \ell\text{-th hyperedge}} x_\ell = 1$.

written in matrix form. ⇒ $Ax = \mathbb{1}$.

Qed. □

□ Subset Sum is NP-complete.

given $a_1, \dots a_n$ and W. whether exists $I \subseteq [n]$. $\sum_{i \in I} a_i = W$.

① Subset Sum is NP. [Proof] y: $(\mathbb{1}[a_i \text{ is chosen in } I])_i$ x: $a_1, \dots a_n, W$

$V(x,y)$: check whether "$\sum_{i \in I} a_i = x[1:n] \odot y = W$."

② ZOE $\leq_K$ Subset Sum.

e.g. ZOE: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

$x \in \{0,1\}^n$. ⇝ ZOE 可以完全转化为没有进位的 Subset Sum 问题.

$a = \{8, 3, 2, 5\}$ $W = 15$

Qed. □

□ Hamiltonian Cycle

a cycle visiting all vertices for exactly once

① NP: y: a possible order of all vertices $x = G$. $V(x,y)$: if y is a H. cycle on x.

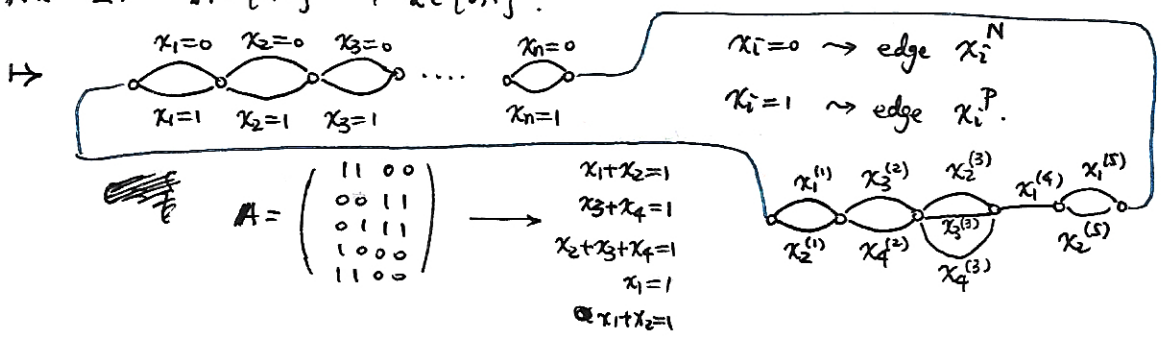
② ZOE $\leq_K$ Hamiltonian Cycle

[Proof]. We prove ZOE $\leq_K$ H.C. with forbidden pairs $\leq_K$ H.C.

H.C. with forbidden pairs:

Given $G=(V,E)$. $C \subseteq E \times E$. whether exists a Hamiltonian cycle H s.t.
(edge pairs) for any $(e_1,e_2) \in C$. either $e_1 \in H$ or $e_2 \in H$.

(1) $Ax = \mathbb{1}$. $A \in \{0,1\}^{m \times n}$, $x \in \{0,1\}^n$.



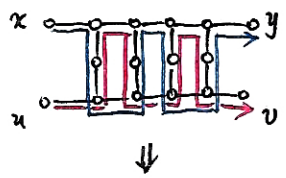
Problem: We want x_1^P and $x_1^{(1)}$ are chosen simultaneously. (因为代表含义都是 $x_1=1$. 所有 $x_1=1$ 的边 应该是同时被选入H.C.的)

e.g. Choose $x_1^N \rightarrow$ can't choose $x_1^{(1)}$.

$$\Rightarrow C := \left\{ (x_1^{(1)}, x_1^N), (x_2^{(1)}, x_2^N), (x_3^{(2)}, x_3^N), (x_4^{(2)}, x_4^N) \ \ldots\ (x_1^{(5)}, x_1^N), (x_2^{(5)}, x_1^N) \right\}$$

∴ ZOE $\leq_K$ Hamiltonian Cycle with forbidden pairs

(2)



Left → Right (no need to visit all the four vertices on the left(right))

2 paths. (Blue & Red) can visit all vertices in the internal part

$\Downarrow$

$((x,y),(u,v))$ pair!

Blue → Use $\{x,y\}$ in H.C. with forbidden pairs

Red → Use $\{u,v\}$ in H.C. with forbidden pairs

Replace all pairs in C with the gadget → New Graph $\tilde{G}$

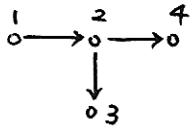
Hamiltonian Cycle on G with forbidden pairs $\Rightarrow$ Hamiltonian Cycle on $\tilde{G}$

∴ Hamiltonian Cycle with forbidden pairs $\leq_K$ Hamiltonian Cycle

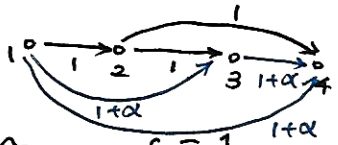
ZOE $\leq_K$ Hamiltonian Cycle. Qed. □

□ TSP

① NP. omitted ② Hamiltonian Cycle $\leq_K$ TSP.



$\Rightarrow$



$\widetilde{G}$: $w(\cdot)$ $\begin{cases} = 1. & \cdot \in E \\ = 1+\alpha & \cdot \notin E \end{cases}$

$\widetilde{E} = V \times V$

Hamiltonian Cycle on G $\Rightarrow$ TSP on $\widetilde{G}$. $TSP = n$

no Hamiltonian Cycles on G $\Rightarrow$ TSP on $\widetilde{G}$. $TSP \geq n+\alpha$ Qed. □

* There are no c-suboptimal ~~solution~~ Polynomial algorithm of TSP if c is finite.

e.g. 2^n-suboptimal. 用2^n-近似算法. $\alpha = 2^n$ 若有 Polynomial 算法 $\Rightarrow$ H.C. 是 P.

→ 任意近似比求解 TSP 依然是 NP-complete 问题.

□ 3-Coloring

Graph $G = (V, E)$. Whether we can color vertices of G with only 3 kind of colors s.t. adjacent vertices are of different color.

① 3-Coloring is NP. [Proof] y: Vertices → colors. A coloring

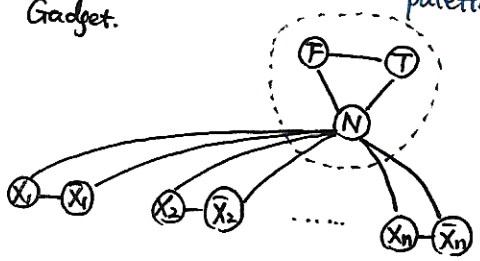
$x = G$.

$V(x,y)$: whether y is a 3-coloring.

verifiable in polynomial time.

② 3-SAT $\leq_K$ 3-Coloring.

Variable Gadget.



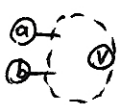
color of T: "True"

color of F: "False"

color of N: "Neutral"

x_i and $\bar{x}_i$ are either of color "True, False" or of color "False, True".

Clause Gadget. —— Goal:



① one of a, b is of color "True"

∃ coloring s.t. v is of color "True"

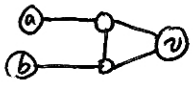
② neither of a, b is of color "True"

∀ coloring. v can not be of color "True".

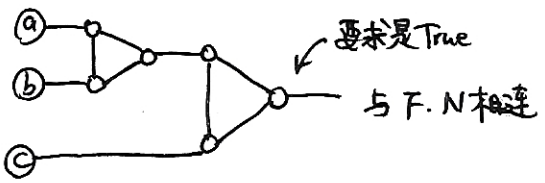
$C = a \vee b \vee c$

$(a, b, c \in \{a_1, \bar{a}_1, a_2, \bar{a}_2, \cdots a_n, \bar{a}_n\})$

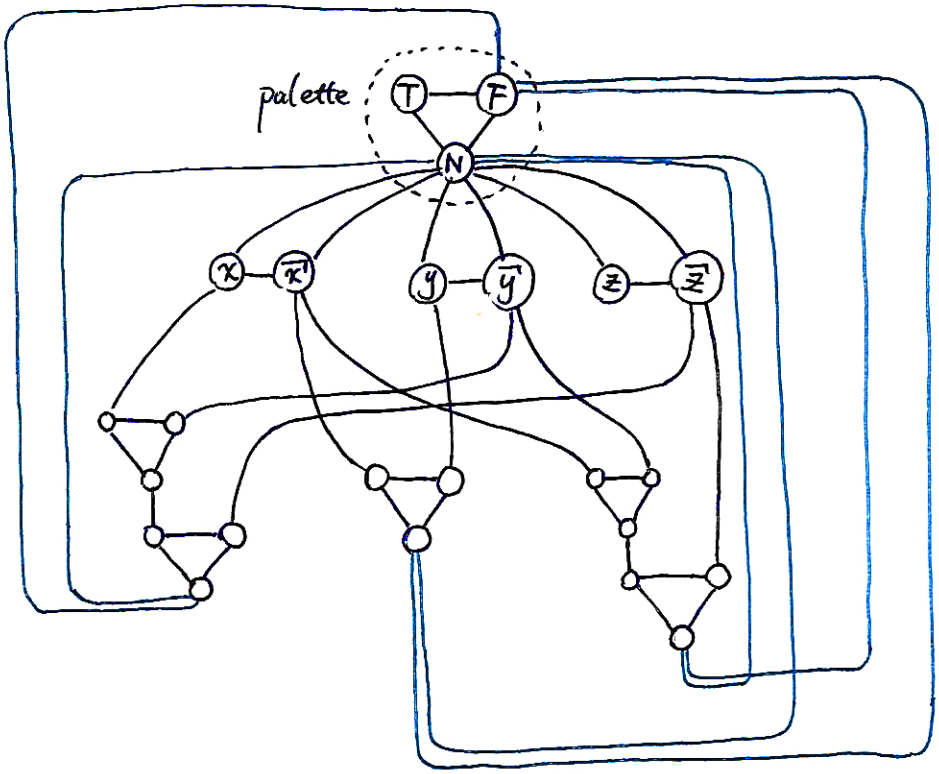
OR Gadget:



$C = a \vee b \vee c$. →



e.g. $(x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \wedge \bar{y} \wedge z)$



Qed. □