$f(n)=o(g(n))$ $\lim_{n\to\infty}\frac{f}{g}=0$ $f<g$ $2^{\sqrt{n}}>n^{\log n}$

$f(n)=O(g(n))$ $\lim_{n\to\infty}\frac{f}{g}<\infty$ $f\le g$

$f(n)=\Theta(g(n))$ $0<\lim_{n\to\infty}\frac{f}{g}<\infty$ $f\sim g$

$f(n)=\Omega(g(n))$ $\lim_{n\to\infty}\frac{f}{g}>0$ $f\ge g$

$f(n)=\omega(g(n))$ $\lim_{n\to\infty}\frac{f}{g}=\infty$ $f>g$

$\ln n< n< n\ln n< n^2< e^n< n!$

Greedy
证明 ① 对步骤进行归纳，证明每步后取得到的结果至少和其他任何algo一样好。② 交换论证：把任何解转化为greedy.

Interval scheduling
Earlier finish time first.
Assume greedy is not optimal. Let $i_1, i_2, \dots i_k$ denote set of jobs selected by Greedy; Let $j_1, j_2, \dots j_m$ denote set of jobs in the optimal solution with $i_1=j_1, i_2=j_2, \dots i_r=j_r$ for the largest possible value of r.
Job i_{r+1} finishes before j_{r+1}, we can replace j_{r+1} with i_{r+1}
⇒ solution still feasible and optimal ⇒ r is not the largest.

Minimizing lateness
Earliest deadline first
S^* be an optimal schedule that has the fewest number of inversions and no idle time. If S^* has no inversion, $S=S^*$; Else, swapping inversions does not increase the maximum lateness and decrease the number of inversions.
⇒ S与S^*是最少inversion conflict.

Divide and Conquer
Merge sort
$T(n)=2T(\frac{n}{2})+\Theta(n)=\Theta(n\log n)$

Closest Pair
① sort in x; sort in y $O(n\log n)$.
② $T(n)\le T(\frac{n}{2})+O(n) \Rightarrow T(n)=O(n\log n)$
(对于中间：scan points in y-order and compare distance between each point and next 11 neighbors. If any of these distances is less than δ, update δ.)

Integer multiplication
$T(n)=4T(\frac{n}{2})+\Theta(n)$

$xy=2^n x_1y_1+2^{\frac{n}{2}}(x_1y_0+x_0y_1)+x_0y_0$

Karatsuba: $T(n)\le T(\lfloor\frac{n}{2}\rfloor)+T(\lceil\frac{n}{2}\rceil)+T(1+\lceil\frac{n}{2}\rceil)+\Theta(n)$

$T(n)=3T(\frac{n}{2})+\theta(n) \Rightarrow T(n)=\sum_{k=0}^{\log_2 n} n(\frac{3}{2})^k=3n^{\log_2 3}-2n$

Matrix multiplication
$T(n)=8T(\frac{n}{2})+\Theta(n^2)\Rightarrow T(n)=\Theta(n^3)$

Dynamic Programming
Weighted Interval scheduling
Label by finish time
$p(j)$ largest index $i<j$ such that job is 相容 with j.

$OPT(j)=\begin{cases}0, & \text{if } j=0\\ \max\{v_j+OPT(p(j)), OPT(j-1)\}, & \text{otherwise}\end{cases}$

时间：① sort by finish time → $O(n\log n)$ ② $p(\cdot)\to O(n)$.
③ Each $OPT(j)$ is computed once. In each computation, $OPT(\cdot)$ is invoked twice. → $O(n)$.

Knapsack Problem
w_i, v_i. 重量限制W.

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

$\Theta(nW)$. 指数.

RNA secondary structure
$OPT(i,j)=\begin{cases}0, & j-i\le 4. \text{ (no sharp turn)}\\ \max\{\underbrace{OPT(i,j-1)}_{b_j\text{不配对}}, 1+\max_{i\le t<j-4}\{OPT(i,t-1)+OPT(t+1,j-1)\}\}, & \text{otherwise}\end{cases}$

先计算最短区间 $(j-i=5)$, 依次增大.
and b_t-b_j 可配对.
$O(n^3)$.

Sequence Alignment
$cost(M)=\sum_{(x_i,y_i)\in M}\alpha_{x_iy_i}+\sum_{i:x_i \text{ unmatch}}\delta+\sum_{j:y_j \text{ unmatch}}\delta$

$OPT(i,j)=\begin{cases}j\delta, & i=0\\ \min\{\alpha_{x_iy_j}+OPT(i-1,j-1), \delta+OPT(i-1,j), \delta+OPT(i,j-1)\}\\ i\delta, & j=0.\end{cases}$

Time & space: $\Theta(mn)$
x长度 y长度

Shortest Path
从s→t. 走无环. (正环非最短，负环无最短路)

$OPT(i,v)=\begin{cases}0, & v=t.\\ \infty, & i=0, v\ne t.\\ \min\{\min_{(v,w)\in E}\{OPT(i-1,w)+c_{vw}\}, OPT(i-1,v)\}, & \text{otherwise}\end{cases}$

最长i条边 从v出发
结果: $OPT(n-1,v)$. $n:|V|$.

Sequence mergable check
$A.B\to C$.
$R(i,j)$ $A[0:i]+B[0:j]\to C[0:i+j]$.

$R(0,0)=True$

$R(0,j)=(R(0,j-1))\wedge B[j-1]==C[j-1]$ $j\ge 1$

$R(i,0)=(R(0,j-1))\wedge A[i-1]==C[i-1]$ $i\ge 1$

$R(i,j)=(R(i,j-1)\wedge A[j-1]==C[i+j-1])\vee(R(i-1,j)\wedge B[j-1]==C[i+j-1])$

Result $R(m,n)$ Time, space $O(mn)$.

Sparse table 区间最值query
preprocessing: $\theta(n\log n)$
create sparse[n][$\lfloor\log n\rfloor+1$]; initialize sparse[i][0]
for j in {1, ⋯ $\lfloor\log_2 n\rfloor$}:
  for i in {0, …, $n-2^j$}:
    sparse[i][j] ← max{A[sparse[i][j-1]], A[sparse[$i+2^{j-1}$][j-1]]}
Lookup: $\theta(1)$
len ← r−l−1, k ← $\lfloor\log_2(len)\rfloor$
return max{A[sparse[l][k]], A[sparse[$r+len-2^k$][k]]}

Master Theorem
$T(n)=aT(\frac{n}{b})+f(n)$

① $f(n)=O(n^{\log_b a-\varepsilon})$ for some constant $\varepsilon>0$, Then $T(n)=\Theta(n^{\log_b a})$

② $f(n)=\Theta(n^{\log_b a})$, $T(n)=\Theta(n^{\log_b a}\lg n)$

③ $f(n)=\Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon>0$, 且 $af(\frac{n}{b})\le cf(n)$ for some constant $c<1$ when n足够大, $T(n)=\Theta(f(n))$

CS101
Guests
n people, enough table & chair. r_i, l_i 希望空出 neighbors
Sort l, r in increasing order respectively.
$\sum_{k=1}^{n}\max(l_k,r_k)+n$. 用 Left[i] to record neighbors
用inversion 证.

Three partition
$OPT(i,j,k)=\begin{cases}(i==0)\wedge(j==0), & \text{when } k=0\\ OPT(i,j,k-1)\vee OPT(i,j-a_k,k-1)\vee OPT(i-a_k,j,k-1), & \text{otherwise}\end{cases}$
result: $OPT(\frac{w}{3},\frac{w}{3},n)$
$O(w^2n)$. pseudo

Steel Beams
$C=(c_1,\cdots c_k)$ 升序, 求 $(a_1,\cdots a_k)$ 使 $\sum_{i=1}^{k}a_ic_i=T$ 的最小 $\sum a_i$. 当sum是n, $\sum a_i$ 最小值
$f(n)=\min_{i\in\{1,\cdots k\}} f(n-c_i)+1$, $f(0)=0$. $O(Tk)$.

Network Flow
Max flow min cut
$v(f)=\sum_{e \text{ out of } A}f(e)-\sum_{e \text{ out of } A}f(e)$

$cap(A,B)=\sum_{e \text{ out of } A}c(e)$. $v(f)\le cap(A,B)$

FordFulkson
$O(m^2\log C)$, $O(mnC)$.
Start with $f(e)=0$ for all edges $e\in E$. 算 G_f.
Find an augmenting path P in the residual graph (can be chosen using capacity scaling)
Augment flow along path P. Repeat until stuck.

Bipartite Matching
原图 $G=(L\cup R,E)$
Create diagraph $G'=(L\cup R\cup\{s,t\},E')$ (method)
Direct all edges from L to R and assign $c=\infty$.
Add source s, and unit capacity edges from s to L.
Add sink t, and unit capacity edges from each node in R to t.
(Proof) Max cardinality matching in G = $v(f)$ max in G'.
≤ Given max matching M of cardinality k. Consider a flow that sends 1 unit along each of k paths.
⇒ f is a flow. $v(f)=k$.
≥ f be a max flow in G' of value k. k is integral.
Consider M = set of edges from L to R with $f(e)=1$.
Then each node in L and R participants in at most one edge in M; $|M|=k$.

Perfect matching
⇔ $|N(S)|\ge|S|$ for all subsets $S\subseteq L$
(Proof) ⇐ Suppose G does not have a perfect matching.
Formulate as a max flow problem and let (A,B) be min cut in G'. Define $L_A=L\cap A$, $L_B=L\cap B$, R_A, R_B.
$cap(A,B)=v(f^*)=|M|<|L|$ (因为无perfect matching)
Since min cut doesn't use ∞ edges in between L&R, no edge between L_A and R_B. $cap(A,B)=|L_B|+|R_A|$.
$\therefore N(L_A)\le R_A$. $\therefore |N(L_A)|\le|R_A|=cap(A,B)-|L_B|<|L|-|L_B|=|L_A|$.
contradict ⇐

Disjoint path
⇔ no edge in common. (V,E), 已知 s, t.
(Theorem) Assign unit capacity to every edge. Max # of edge disjoint s-t paths equals max-flow value.
(Proof) ≤ Given k edge-disjoint paths $P_1, \cdots P_k$.
Set $f(e)=1$ if e participates in some path P_i; else set $f(e)=0$. Since paths are edge-disjoint, f is a flow of value k.
≥ Let f be a max flow whose value is k.
Integrality theorem ⇒ k is integral and can assume f is 0-1. Consider edge (s,u) with $f(s,u)=1$.
$\exists e=(u,v), f(e)=1$. continue until reach t, always choose new edge. ⇒ get a s-t path. Reduce flow to 0 along the path.
⇒ get a flow of k−1. keep doing for k times ⇒ k dis paths.

Extensions to max flow
Circulation with demands
$d(v)\begin{cases}>0 & \text{demand}\\ <0 & \text{supply}\end{cases}=\sum_{e \text{ in } v}f(e)-\sum_{e \text{ out } v}f(e)$

Necessary condition: $-\sum$ supply $=\sum$ demand $=D$
Method: Add new source s and sink t. For each v with $d(v)<0$, add (s,v) with capacity $-d(v)$; For each v with $d(v)>0$, add edge (v,t) with capacity $d(v)$.
Claim: G has circulation ⇔ G' has max flow $v(f)=D$.
不存在circulation ⇔ ∃ node partition (A,B) such that $\sum_{v\in B}d_v>cap(A,B)$

Circulation with d & lower bound
$l(e)\le f(e)\le c(e)$, $d(v)$.
(v) —[2,9]→ (w)   (v) —7→ (w)
d(v) d(w)   d(v)+2 d(w)−2

Other max flow example
Image segmentation
Find (A,B) max $\sum_{i\in A}a_i+\sum_{j\in B}b_j-\sum_{(i,j)\in E, |A\cap\{i,j\}|=1}p_{ij}$
⇔ min $\sum_{j\in B}a_j+\sum_{i\in A}b_i+\sum_{(i,j)\in E, |A\cap\{i,j\}|=1}p_{ij}$

Method
$G'=(V',E')$; Add s to foreground with $c=a_i$, add sink to background with b_i; Use 2 anti-parallel edges instead of undirected edges. (p_{ij} each way)
Proof $cap(A,B)=\sum_{j\in B}a_j+\sum_{i\in A}b_i+\sum_{(i,j)\in E, i\in A, j\in B}p_{ij}$

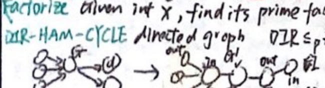
Project Selection
Method
Include v→w if v is prerequisite of w. set $c(v,w)=\infty$.
Add (s,v) with capacity p_v if $p_v>0$.
Add (v,t) with capacity $-p_v$ if $p_v<0$.
Proof
Infinite capacity edges ensure A−{s} feasible.
Max revenue because $cap(A,B)=\sum_{v\in B, p_v>0}p_v+\sum_{v\in A, p_v<0}(-p_v)$

$=\sum_{v\in B, p_v>0}p_v+\sum_{v\in A, p_v>0}p_v-\sum_{v\in A, p_v>0}p_v+\sum_{v\in A, p_v<0}(-p_v)=\underbrace{\sum_{v, p_v>0}p_v}_{\text{constant}}-\sum_{v\in A}p_v$.

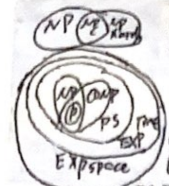
最大独立集
Method (m, n).
A: m+n is odd B: m+n is even.
v_A & v_B adjacent: add$(v_A,v_B)=\infty$.
Add (s,v) with capacity c(v) for v∈A.
Add (v,t) with capacity c(v) for v∈B.
Proof
Infinite capacity ensure we only cut (s,v) or (v,t). Those that are not in the cut but connect to s/t are maximum selected elements.

Secrete Santa
Method For guest i, make 2 vertices u_i and v_i.
$U=\{u_i: i=1,\cdots n\}$, $V=\{v_i: i=1,\cdots n\}$. $G=(U\cup V,E)$.
There is an edge between (u,v) when i can give a gift to j.
$c_e=1$. Run Fold Furlkson. size(f) = n ⇔ 可以分配出去礼物.
Proof size(f) = n ⇔ perfect matching

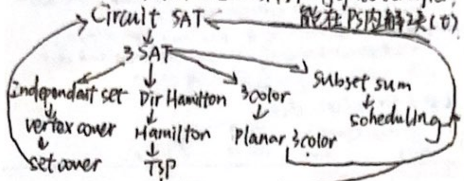
NP
(NPC 还需证 mapping take poly time)
Independent Set Is there $|S|\ge k$, 对于每条边至多一点在S?
Vertex Cover Is there $|S|\le k$, 对于每条边至少一点在S?
Set Cover U有子集 S_i. Is there ≤k个子集的union是U?
3SAT ≤ IS. △一个clause一个△, $\bar{x}$ 与 x 连. k=個句中可满足.
NP decision problem + 多项式验证 certificate 值, certifier 算法.
poly size poly time
3SAT $(x_1\vee\bar{x_2}\vee\bar{x_3})\wedge(\cdots)\wedge\cdots$
HAM-CYCLE (无向) does there exist a simple cycle that visits every node?
SUBSET-SUM does there exist a sub set sum of W? (一个整数)
CO-NP 补问题为NP $P\subseteq NP\cap co\text{-}NP$ 不等于. e.g. factor ∈ P.
If $NP\ne co\text{-}NP\Rightarrow P\ne NP$. certifier + disqualifier
Factor (NP) Given 2 int x and y. does x have a nontrivial factor less than y.
这两不是NPC
Factorize Given int x, find its prime factorization.
DIR-HAM-CYCLE directed graph DIR $\le_p$ 无向.



TSP Given n cities, $d(u,v)$, is there a tour ≤ D. (length)
HAMCYCLE $\le_p$ TSP $d(u,v)=\begin{cases}1, & \text{if } (u,v)\in E\\ 2, & \notin E\end{cases}$ D=节点数.
3D matching X, Y, Z disjoint. Each of size n. $T\subseteq X\times Y\times Z$.
Does there exist a set of n triples in T such that each element of $X\cup Y\cup Z$ is in exactly one of these triples
$P\subseteq NP$, $co\text{-}NP\subseteq EXP$, $P\subseteq NP\le PSPACE\le EXPTIME$
PSPACE decision problem + poly space.



P space: QSAT, Planning, Facility Location.
Pspace complete: QSAT, competitive
3D matching $\in$ NPC
FACTOR $\in$ NP FACTOR $\in$ P.
QSAT $\in$ PSPACE-COMPLETE
QSAT $\in$ NP-hard, QSAT $\in$ EXP.
NPC 不是 PSC. 若 P=PSPACE, NP=PSPACE;
若 a PSPACE-Complete $\in$ P, NP $\neq$ EXP.
X $\in$ PSPACE $\Rightarrow$ X 有 poly space certifier, 能在 PS 内解决 (?)



Las Vegas 总是对. Monte Carlo 有时对.
ZPP $PC \subseteq ZPP \subseteq RP$. RP.

Coupon collector's problem

每次去 random 给一个. 共 n 个 coupon. 求得到所有 n 个需要 take 的次数. X. $E(X) = n\ln n + O(n)$ with 高概率
证: $E(X) = O(n\log n)$ with high probability
Epochs. time period between getting a new coupon.
X_i = the length of the ith epoch (已有 i 张)
$E(X_i) = \frac{n}{n-i}$. (在第 i 次得到 coupon 的概率是 $\frac{n-i}{n}$)
$X = X_1 + X_2 + \cdots + X_{n-1} \Rightarrow E(X) = \frac{n}{n-1} + \frac{n}{n-2} + \cdots \frac{n}{1} \simeq n\ln n$
F_i^r = the event that coupon type i is not collected in the first r trials. $Pr(F_i^r) = (1-\frac{1}{n})^r \le e^{-r/n}$
Let $r = \beta n\ln n$ $(\beta > 0)$ $e^{-r/n} = e^{-\beta\ln n} = n^{-\beta}$
$Pr(X > r) = Pr(\bigcup_{i=1}^n F_i^r) \le \sum_{i=1}^n Pr(F_i^r) \le n^{-\beta+1}$
$\beta \to \infty$. $Pr(X > r) \to 0$.

$\forall n \ge 2$, 有 $\frac{1}{e} \le (1-\frac{1}{n})^{n-1} \le \frac{1}{2}$, 且 $\frac{1}{4} \le (1-\frac{1}{n})^n \le \frac{1}{e}$.
MIS maximal 局部 →
MaxIS maximum 全局.

shortest Path

$DP(i,v)$ 用 i 个边从 v 到 $t \Rightarrow O(n^2)$. $DP(i,v) = \min\begin{cases} DP(i-1,v) \\ DP(i-1,w) + w_{uv}\end{cases}$
只用更新上次变过的点. 查负环: 对所有 V, $OPT(n-1,v) = OPT(n,v)$ 无负环.

Lower Bound

Merging 2 lists (sorted list a, b) lower = upper = $2n-1$ 比较
① $\forall i$. 需比较 $a_i, b_i \Rightarrow n$. ② $\forall i < n$. a_{i+1}, b_i 比 $\Rightarrow n-1$
Finding the max lower = upper = $n-1$.
3-color array: white, Blue, Red
未比较过 最大 最小
$\forall k$. $w_k + b_k \ge n-k$. (induction WW, WB, WR, RR, BB, RB).
Find the max and min lower = $\frac{3n}{2} - 2$.
4-color array: 白, 蓝, 红, 紫
$\wedge/\wedge$ $>>$ $<<$ $<b<$ r = ww + bw − rr.
claim w = n − 2ww − rw − bw − pw b = ww + rw + pw − bb.
Sorting upper = lower = $n\log n$.
output a permutation of the order of input.

Global min cut

Given 连通无向图 $G=(V,E)$, find a cut (A,B) of min cardinality $O(n^2\log n)$ (权重为1) 目前最快 $O(m\log^3 n)$
Contraction algorithm pick an edge at random and contract it. (保留平行边, 去除自环) 运行直到只剩两点.
至少概率. $\ge \frac{2}{n^2}$. global min cut (A^*, B^*). 令 F^* be edges with one point in A^* and other point in B^*. $k = |F^*|$
Step 1: $k/|E|$ contract F^*. 每个 node 都有 $\deg \ge k$.
$\Rightarrow |E| \ge \frac{1}{2}kn \Leftrightarrow \frac{k}{|E|} < \frac{2}{n}$.
Step j. $n' = n-j$. $Pr \le 2/n'$. E_j: 某边在 iter j 中未被 contract $\in F^*$
$Pr(E_1 \cap E_2 \cap \cdots E_{n-2}) = Pr(E_1) \times Pr(E_2|E_1) \times \cdots Pr(E_{n-2}|E_1 \cap \cdots E_{n-3})$
$\ge (1-\frac{2}{n})(1-\frac{2}{n-1})\cdots(1-\frac{2}{4})(1-\frac{2}{3})$ $(1-\frac{1}{x})^x \le \frac{1}{e}$.
$= \frac{n-2}{n}\frac{n-3}{n-1}\cdots\frac{2}{4}\frac{1}{3} = \frac{2}{n(n-1)} \ge \frac{2}{n^2}$.
$n^2\ln n$ times 失败率 $(1-\frac{2}{n^2})^{n^2\ln n} = ((1-\frac{2}{n^2})^{\frac{1}{2}n^2})^{2\ln n} \le (e^{-1})^{2\ln n} = \frac{1}{n^2}$

Knuth-Morris-Pratt

目前字符串匹配 T 长 S=m 短

```
i←1, j←1                      O(n)
while (j≤n) and i≤m
  if T[i] = S[j]  match
    i←i+1, j←j+1
  elif j=1   first char mis / 不match T[j]
    i←i+1
  else
    j←f(j-1)+1
If j=m-1
  return i-m
return "S not in T".
```

Failure Function O(m)

```
i←2 j←1
f(1)←0
while (i≤m)
  if S[i] = S[j]
    f(i)←j, i←i+1, j←j+1
  else if j>1
    j←f(j-1)+1
  else
    f(i)←0
    i←i+1
```

$\Rightarrow O(m+n)$ 字符串匹配 $= O(n)$

König-Egervary In a bipartite graph, the max cardinality of a matching = min cardinality of a vertex cover.

Dijkstra (G,w,s)

```
Initialize single source (G,s)
S=∅ Q=G.V
while Q≠∅
  u=extract min(Q)
  S=S∪{u}
  for each vertex v∈G.adj[u]
    RELAX(u,v,w)
RELAX(u,v,w)
  if v.d>u.d+w(u,v)
    v.d=u.d+w(u,v)
    v.π=u
```

Binary heap $O((V+E)\log V)$
Fibonacci heap $O(E+V\log V)$

MST-PRIM (G,w,r)

```
for each u∈G.V
  u.key=∞  u.π=NIL
r.key=0  Q=G.V
while Q≠∅
  u=EXTRACT-MIN(Q)
  for each v∈G.adj[u]
    if v∈Q and w(u,v)<v.key
      v.π=u
      v.key=w(u,v)
```

Fibonacci
make, insert, min, union, decrease-key $O(1)$.
extract min, delete $O(\log n)$

Universal Hashing $Pr(h(x)=h(y)) = \frac{1}{m}$. for any keys $x \neq y$, $h \in H$

Nash 纯纳什均衡总是存在 for k-agent multicast routing with fair sharing. Best Nash is never more than a factor of $H(k)$ worse than the social optimum. 目前无 find nash in polytime AI.

Markov 不等式 $Pr(X \ge a) \le E(X)/a$ for $\forall a > 0$.
Chebyshev $Pr(|X-E(X)| \ge a) \le Var(X)/a^2$.
Chernoff ① $x_1 \cdots x_n$ i.i.v. $E(x_i) = p_i$. $X = \sum_i x_i$
$u = E(X) = \sum_i p_i \Rightarrow 0 < \delta \le 1$, $Pr(X \ge (1+\delta)u) \le e^{-u\delta^2/3}$
$\delta > 1$. $Pr(X \ge (1+\delta)u) \le e^{-u\delta\ln\delta/3}$
$0 \le \delta \le 1$. $Pr(X \le (1-\delta)u) \le e^{-u\delta^2/2}$
② $Pr(X \ge (1+\delta)u) \le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^u$
③ $x_1, \cdots x_n$ i.i.v. 取值 $\{-1,1\}$: $Pr(x_i=1) = Pr(x_i=-1) = \frac{1}{2}$.
$X = \sum_i x_i \Rightarrow \forall \delta \ge 0$. $Pr(X \ge \delta) = Pr(X \le -\delta) \le e^{-\frac{\delta^2}{2n}}$
E with 高概率 $\Leftrightarrow Pr(E) \ge 1 - n^{-c}$ $\exists c \ge 1$.

Contention Resolution n 个 node. 一轮只能一个
Algorithm each node try to send with $p = \frac{1}{n}$ per round.
$S_{i,t}$: node i 成功 send at t th round.
$Pr(S_{i,t}) = \frac{1}{n}(1-\frac{1}{n})^{n-1}$ 左边上届A.
$\Rightarrow Pr(S_{i,t}) \ge \frac{1}{ne}$. $Pr(i$ fails at round $t) \le 1 - \frac{1}{ne}$
Thm After $2en\ln n$ rounds, all nodes 成功 with $p \ge 1 - \frac{1}{n}$.
F_i: node i fails after $2en\ln n$ 轮. F: any node
$Pr(F_i) \le (1-\frac{1}{en})^{2en\ln n} \le (\frac{1}{e})^{2\ln n} \le \frac{1}{n^2}$
$Pr(F) \le \sum_i Pr(F_i) \le \frac{1}{n}$

Distributed MIS Each node v chooses a random # $r(v) \in [0,1]$ and sends to its neighbors. If $r(v) < r(w)$ for all neighbors w of v, then v adds itself to the MIS and informs its neighbors. If v or one of its neighbors entered the MIS, v 退出, 加入以后. otherwise 重启 直到 graph 空.

d维LP If $|H| = d$, output their intersection. Pick random constraint $h \in H$. Recursively find $OPT = B(H - \{h\})$. If opt 不 violate h, output OPT. Else project $H - \{h\}$ onto h's boundary to obtain $d-1$ 维 LP. 递归求解 $d-1$ 维 LP. → OPT violate h.
$T(n,d) \le T(n-1,d) + O(d) + \frac{d}{n}(O(dn) + T(n-1,d-1)) = O(d!n)$

近似 A is α-approx algorithm for X if A 总是返回 至少 $\frac{1}{\alpha}$ times the optimal 的结果 $\frac{1}{\alpha}$ 值 < A < α 值
Las Vegas 答案确对, 时间 vary.
Monte Carlo 时间相同 答案可能错. $d=2$
vertex cover 随意挑边, 删 u, v 以及所有 w 相关边.

Finger Printing

p 为随机素数
$a = \sum_{i=1}^n a_i 2^{i-1}$, $b = \sum_{i=1}^n b_i 2^{i-1}$. $F(a) = a \bmod p$.
String Matching X. Y. n, m. $n \ge m$ $O(n+m)$
$X = x_1x_2\cdots x_n$. $X(j) = x_j x_{j+1} \cdots x_{j+m-1}$, 与 Y 一样长.
Y match X $\Leftrightarrow \exists 1 \le j \le n-m+1$, $X(j) = Y$
Monte Carlo $\forall 1 \le j \le n-m+1$ 比 $X(j)$. Y 看是一个逐都对.
计算 $F(X(j)) = X(j) \bmod p$. $F(Y) = Y \bmod p$.
若 $F(X(j)) \neq F(Y)$, go to next j; 若相同, output "match yet".
若一整个 for 无结果, 说 "Y 不 match X".
False match $\le \frac{1}{n}$ $O(n+m)$
Las Vegas 若 $F(X(j))$ 相等 $= Y$. Brute force 检查:
Expected: $O(n+m)$

R QuickSort

$R(n) = \frac{1}{n}(R(1) + R(n-1) + R(2) + R(n-2) + \cdots + R(n-1) + R(1)) + n/n(n-1) = 2/n \sum_k R(k) + n - 1$.
For a good split, it can split the set into 2 parts S_1 and S_2 such that $\min(|S_1|, |S_2|) \ge \frac{1}{4}|S|$ prob $\ge \frac{1}{2}$
对每个 good 分, S 至少 reduce 到 $\frac{3}{4}S$. $x = \log n / \log(\frac{4}{3}) = c\log n$.
Suppose we run the program and stop when the depth of the recursion tree is $c\log n$ for some constant c.
For a single path from root to leaf, average # of good 分 is $\frac{1}{2}c\log n$. $\therefore Pr(\#good 分 \le (1-\delta)\frac{1}{2}c\log n) \le e^{-\frac{1}{2}c\log n\frac{\delta^2}{2}} = \frac{1}{n^{\frac{c\delta^2}{4}}}$
Let $c \ge 8\delta^2$. 右 $= \frac{1}{n^2}$.
For one level, $O(n)$ nodes. union bound
$\therefore Pr(\exists$ some path 从 0 到 $c\log n) < \frac{1}{n^2} \cdot n = \frac{1}{n}$
$\therefore$ Do in $O(n\log n)$ $P \ge 1 - \frac{1}{n}$.

Bellman Ford $O(VE)$. 可检测负环.
Dijkstra $O((V+E)\log V)$

Each node v maintains estimate of it d from s in $v.d$ and its parent in shortest path tree in $v.\pi$. RUN $|V|-1$ rounds. In each round, relax all nodes in arbitrary order. Finally, check if $v.d > u.d + w(u,v)$ for any edge (u,v).
If so, has an NWC, return False.
otherwise, for all nodes v. $v.d = \delta(s,v)$
$v.\pi$ = parent in shortest path tree forms