$\log_b(M \cdot N) = \log_b M + \log_b N$ $\quad \log_b(1) = 0$

$\log_b\left(\frac{M}{N}\right) = \log_b M - \log_b N$ $\quad \log_b b = 1$

$\log_b(M^k) = k \cdot \log_b M$ $\quad \log_b b^K = k$

$b > 1,\ M, N > 0$ $\quad b^{\log_b(k)} = k$

Master Theorem

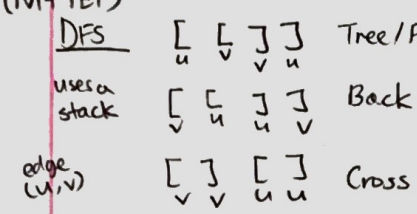
$T(n) = aT(\lceil n/b \rceil) + O(n^d)$

$a > 0,\ b > 1,\ d \geq 0$

$$T(n) = \begin{cases} O(n^d) & d > \log_b a,\ a < b^d \\ O(n^d \log n) & d = \log_b a,\ a = b^d \\ O(n^{\log_b a}) & d < \log_b a,\ a > b^d \end{cases}$$

$O(|V|+|E|)$

DFS



reverse postorder == topo sort

Summations

$1+2+3+\cdots+(N-1)+N = \sum_{i=1}^{N} i \in \Theta(N^2)$

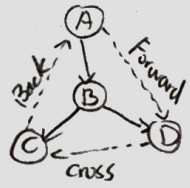
$1+2+4+\cdots+(N/2)+N = \sum_{i=0}^{N} 2^i \in \Theta(N)$

$1+2+4+\cdots+2^{N-1}+2^N = \sum_{i=0}^{N} 2^i \in \Theta(2^N)$

$\log(n!) \in \Theta(n \log n)$

$S_n = \frac{a_1(1-r^n)}{1-r}$ geo

$H_k = \sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k)$



DAG

- has a sink and a source
- a directed graph has a cycle iff its DFS has a back edge
- Every edge leads to vertex w/ lower post #
- Every directed graph is a DAG of its SCCs
- explore(u) will stop when all nodes reachable from u have been visited
- node w/ highest post # in a DFS must be in a source SSC
- If C and C' are SSCs and edge from C to C', the highest post # in C > highest post # in C'

SSCs Two nodes u and v of a directed graph are strongly connected if path u to v and v to u

SSC is a set of vertices such that any pair u, v are strongly connected

$a = e^{2\pi i \theta_1}$, $b = e^{2\pi i \theta_2}$, $ab = e^{2\pi i (\theta_1 + \theta_2)}$

$M_n(w)^{-1} = \frac{1}{n} \cdot M_n(w^{-1})$

FFT

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ 1 & w^{(n-1)} & w^{2(n-1)} & \cdots & w^{(n-1)(n-1)} \end{bmatrix} \quad w = e^{\frac{2\pi i}{n}} \quad w^{ij} \quad \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

$O(n \log n)$

Group even & odd powers to reduce calc

SSC Finding (linear time, $O(|V|+|E|)$)

1. run DFS on G^R to get reverse post-order #'s if directed Graph
2. Init ccnum = 0
3. DFS on G processing vertices in decreasing post numbers. In pre-visit, set cc(v) = ccnum. Increment ccnum when stack empties.

BFS (G, s) $\quad O(|V|+|E|)$

```
for all u ∈ V: dist(u) = ∞
dist(s) = 0
Q = [s] (queue with only s)
while Q is not empty:
    u = eject(Q)
    for all edges (u,v) ∈ E:
        if dist(v) = ∞:
            inject(Q, v)
            dist(v) = dist(u) + 1
```

Dijkstra (G, ℓ, s) priority queue

```
for all u ∈ V: dist(u) = ∞  prev(u) = nil
dist(s) = 0
H = makequeue(V) (using dist-values as key)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + ℓ(u,v):
            dist(v) = dist(u) + ℓ(u,v):
            prev(v) = u
            decreasekey(H, v)
```

$O((|V|+|E|)\log|V|)$ binary heap

Bellman-Ford (G, ℓ, s): $O(|V| \cdot |E|)$

```
for all u ∈ V: dist(u) = ∞  prev(u) = nil
dist(s) = 0
repeat |V|-1 times:
    for all e ∈ E:
        update(e)
```

run 1 more time to detect neg cycle

Least updates if run in topo order for DAGs

$f(n) \in \Omega(g(n))$ f is lower bounded by g

$f(n) \in O(g(n))$ f is upper bounded by g

$$\begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} \quad x^3 + 2x^2 + 3x + 4$$

No edge weights: Try DFS/SCC decomposition

DAG: try topo sort

Has edge weight: Modify Dijkstra / Bellman-Ford (if neg weight)

$T(n) = 2T(\sqrt{n}) + 3$ and $T(2) = 3$ $\quad n^{1/2^h} = 2$

$h = \Theta(\log\log n)$ $\quad T(n) = \Theta(\log n)$ $\quad 2^{h+1} - 1$

More FFT

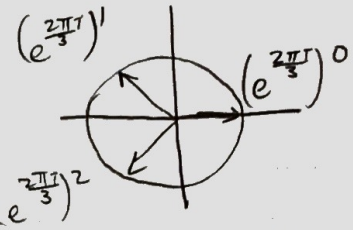
$P(x) = A(x^4) + xB(x^2)$

$P(x^2) = A(x^4) + x^2 B(x^4)$

$P(x^3) = A(x^6) + x^2 B(x^6)$

$P(x^4) = A(x^8) + x^2 B(x^8)$

Need d+1 pts for deg d polynomial



for n, we have n roots of unity

$e^{2\pi i 0}, e^{2\pi i \frac{1}{n}}, e^{2\pi i \frac{2}{n}}, \cdots, e^{2\pi i \frac{(n-1)}{n}}$

$P(x) = P_1(x^3) + xP_2(x^3) + x^2 P_3(x^3)$

$P_1(x^3) = a_0 + a_3 x^3 + a_6 x^6$

$P_2(x^3) = a_1 + a_4 x^3 + a_7 x^6$

$P_3(x^3) = a_2 + a_5 x^3 + a_8 x^6$

```
for j=0 to n/2:                    n=2   -1
    r_j     = s_j + w^j (s'_j)     n=4   -i
    r_{j+n/2} = s_j - w^j (s'_j)   n=8   √2/2 + √2/2 i
```

LP row $\max z$, $\sum x_i = 1$ $\quad z \leq$ expected val for each pure strat

$(z \leq x_2 - x_3)$

Complex # inverse 1. write reciprocal

2. multiply top and bottom by conjugate

3. simplify

```
for j=0 to n/2:
    r_j = s_j + w^j (s'_j)
    r_{j+n/2} = s_j - w^j (s'_j)
```

```
if w == 1: return A(1)
express A(x) in form Ae(x^2) + x Ao(x^2)
call FFT(Ae, w^2) for Ae at even powers
call FFT(Ao, w^2) for Ao at even powers
for j=0 to n-1:
    compute A(w^j) = Ae(w^{2j}) + w^j Ao(w^{2j})
return A(w^0), ..., A(w^{n-1})
```

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \quad \text{inv}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & 1 & i \end{pmatrix} \quad \text{normal}$$

	Dijkstra Heap Runtime		
	deletemin	insert/ decrease key	$\lvert V\rvert$ · deletemin + $(\lvert V\rvert + \lvert E\rvert)$ · insert
Array	$O(\lvert V\rvert)$	$O(1)$	$O(\lvert V\rvert^2)$
bi-heap	$O(\log\lvert V\rvert)$	$O(\log\lvert V\rvert)$	$O((\lvert V\rvert + \lvert E\rvert)\log\lvert V\rvert)$
d-ary heap	$O(\frac{d\log\lvert V\rvert}{\log d})$	$O(\frac{\log\lvert V\rvert}{\log d})$	$O((\lvert V\rvert \cdot d + \lvert E\rvert)\frac{\log\lvert V\rvert}{\log d})$
Fib-heap	$O(\log\lvert V\rvert)$	$O(1)$ amortized	$O(\lvert V\rvert\log\lvert V\rvert + \lvert E\rvert)$

Misc. If post(u) < pre(v) in DFS, edge (u, v) cannot exist

Kruskal $O(E \log V)$

Add lightest edge w/o cycles

Properties of tree

Tree w/ n nodes has n-1 edges

Connected graph $G(V, E)$ w/ $|E| = |V| - 1$ is a tree

Undirected graph is a tree iff there is a unique path between any pair of nodes.

Cut property

Edges X are part of a MST of $G = (V, E)$. Pick any subset of nodes S for which X does not cross between S and V-S, and let e be the lightest edge across the partition. Then $X \cup e$ is part of some MST.

Prims

Each iter, subtree x grows by one lightest edge between a vertex in s and one outside S.

Huffman

Opt encoding based on distribution

- keep combining two lowest frequencies nodes
- if more than 0.4, then code word of length 1 guaranteed

Horn Formulas

- Implications - LHS is an AND of any # of pos literals and RHS is a single pos literal.

$(z \wedge w) \Rightarrow u$

If left holds, right must be true

- pure negative clauses, consisting of an OR of any # of negative literals.

$(\bar{u} \vee \bar{v} \vee \bar{y})$

- To solve, set all to false while there is an implication not satisfied:

set RHS of implication to True

If all pure neg clauses are satisfied return assignment

return "impossible"

Set Cover Algorithm

while (B not covered):

Pick the set S_i w/ largest # of uncovered elems.

Dynamic Programming

Longest increasing subsequence

```
L = {}
for j = 1,2,...n
    L[j] = 1 + max{L[i] : (i,j) in E}
return max(L)
```

$O(n^2)$

Edit distance

- choose from min of adding a gap between letters or matching the two letters and move on.

```
for i = 0,1,...m
    E(i,0) = i
for j = 1,2,...n
    E(0,j) = j
for i = 1,2,...m
    for j = 1,2,...n
        E(i,j) = min{E(i-1,j)+1,
            E(i,j-1)+1, E(i-1,j-1)+diff(i,j)}
return E(m,n)
```

Knapsack $O(nW)$

w/ repetition:

$K(w) = \max_{\text{items}} \{K(w - w_i) + v_i\}$

w/o repetition:

$K(w, j) = \max_{\text{avail}} \{K(w - w_j, j-1) + v_j, K(w, j-1)\}$

Chain Matrix Multiplication

$C(i,j) = \min\{C(i,k) + C(k+1, j) + m_{i-1} \cdot m_k \cdot m_j\}$

Traveling Salesman Problem

$C(\{1\}, 1) = 0$ $\quad O(n^2 2^n)$

```
for s = 2 to n:
    for all subsets S in {1,2,...n} of
    size s and has 1:
        C(S,1) = inf
        for all j in S, j != 1:
            C(S,j) = min(C(S-{j}, i) + d_ij
                i in S, i != j
return min over j, C({1,...n}, j) + d_j1
```

LP

1. max $\Rightarrow$ min, just mul obj func by -1
2. Turn inequality constraint like $\sum_{i=1}^{n} a_i x_i \leq b$ into an equation, use a new var s and use $\sum_{i=1}^{n} a_i x_i + s = b$, $s \geq 0$
3. To change $ax = b$, write as $ax \leq b$ and $ax \geq b$
4. If LP unbounded, dual must be infeasible.

Max-Flow Min Cut

size of max flow in a network equals the capacity of the smallest (s,t) cut where an (s,t) partitions the vertices into 2 disjoint groups L and R such that s in L and t in R.

Ford-Fulkerson $O(|E| * f)$

1. While there exists an augmenting path from s-t
2. push as much flow along the path possible
3. when no path exists, max-flow found

Zero sum games

	C_1	C_2	C_3
R_1	5	0	-2
R_2	-3	-1	-10
R_3	3	1	-4

maximizer:

$\max z$

$z \leq 5x_1 - 3x_2 + 3x_3$

$z \leq -x_2 + x_3$

$z \leq -2x_1 - 10x_2 - 4x_3$

$x_1 + x_2 + x_3 = 1$

$x_1, x_2, x_3 \geq 0$

minimizer:

$\min w$

$w \geq 5y_1 - 2y_3$

$w \geq -3y_1 - y_2 - 10y_3$

$w \geq 3y_1 + y_2 - 4y_3$

$y_1 + y_2 + y_3 = 1$

$y_1, y_2, y_3 \geq 0$

$\log^*(65536) = 4$

set cover $O(\ln n)$

Primal

$\max c_1 x_1 + \cdots + c_n x_n$

$a_{i1} x_1 + \cdots + a_{in} x_n \leq b_i$ for $i \in I$

$a_{i1} x_1 + \cdots + a_{in} x_n = b_i$ for $i \in E$

$x_j \geq 0$ for $j \in N$

Dual

$\min b_1 y_1 + \cdots + b_m y_m$

$a_{1j} y_1 + \cdots + a_{mj} y_m \geq c_j$ for $j \in N$

$a_{1j} y_1 + \cdots + a_{mj} y_m = c_j$ for $j \notin N$

$y_i \geq 0$ for $i \in I$

$\max 2x_1 + x_2$

$x_1 + 2x_2 \leq 5$

$3x_1 + 4x_2 \leq 2$

$x_1, x_2 \geq 0$

$\min 5y_1 + 2y_2$

$y_1 + 3y_2 \geq 2$

$2y_1 + 4y_2 \geq 1$

We constrain y_1, y_2 so that they yield coeff x_1, x_2 that are >= those in primal obj. LP will give $(y_1 + 3y_2)x_1 + (2y_1 + 4y_2)x_2 \leq 5y_1 + 2y_2$. The constraints in dual force $y_1 + 3y_2 \geq 2$ and $2y_1 + 4y_2 \geq 1$. $2x_1 + x_2 \leq 5y_1 + 2y_2$ for all feasible

Simplex

typically poly time, exp worst case

let v be any vertex of feasible region

while there's a neighbor w/ better value

set $v = v'$

return v

Two player coin game

Array $[a_1 \dots a_n]$

Let $S(i,j)$ be the maximum score the curr player can obtain if player is left w/ coin array $A[i \dots j]$

Recurrence

$S(i,j) = \max\{L_i, R_i\}$

$L_i = A[i] + \min\{S(i+2,j), S(i+1,j-1)\}$

$R_i = A[j] + \min\{S(i+1,j-1), S(i,j-2)\}$

Base cases

$S(i,j) = \begin{cases} 0 & \text{if } i > j \\ A[i] & \text{if } i = j \end{cases}$

Breaking chocolate

$B[i_1, j_1, i_2, j_2]$ to be the minimum # of breaks needed to separate sub-matrix $A[i_1 \le i_2, j_1 \le j_2]$ into pure pieces

Recurrence

$B[i_1, j_1, i_2, j_2] =$

$\min \begin{cases} 0, \text{ if all } A[i_1 \dots i_2, j_1 \dots j_2] \text{ are equal} \\ 1 + B[i_1, j_1, i_1+k, j_2] + B[i_1+k+1, j_1, i_2, j_2] \\ 1 + B[i_1, j_1, i_2, j_1+k] + B[i_1, j_1+k+1, i_2, j_2] \end{cases}$

for any $k \in \{1 \dots i_2 - i_1,\ j_2 - j_1\}$

Balloon popping

$C(i,j)$ = the maximum amount of mine by popping all balloons in the sequence $i, i+1, \dots j$. Smallest case is 1 balloon size of sub problem is # of mul

Base case

one balloon → Spopped

recurrence

$C(i,j) = \max_{i \le k \le j} \{C(i,k-1) + C(k+1,j) + S_{i-1} \cdot S_k \cdot S_{j+1}\}$

Zero sum games

max w

$w \le 200x_1$

$w \le -100x_1 + 200x_2$

$x_1 + x_2 = 1$

$x_1, x_2 \ge 0$

min z

$z \ge 200y_1 - 100y_2$

$z \ge 200y_2$

Row max

$200x_1 = -100x_1 + 200x_2$

$300x_1 = 200x_2$

$3x_1 = 2x_2$

$200y_1 - 100y_2 = 200y_2$

$200y_1 = 300y_2$

$2y_1 = 3y_2$

$y_1 = 0.6 \quad y_2 = 0.4$

Since $x_1 + x_2 = 1$, $x_1 = 0.4 \quad x_2 = 0.6$

find max by setting equal

	200	-100
	0	200

expected payoff

$200x_1y_1 - 100x_1y_2 + 0x_2y_1 + 200x_2y_2 = 80$

moving on a grid

Let $T(i,j)$ be the min # of tokens needed to start at cell (i,j) and reach (n,n)

Base cases

$T(n,n) = 0$

Recurrence

Let $T(i,j) = \max\{\min(T(i+1,j), T(i,j+1)) - G(i,j), 0\}$

minimum gold stakes

Let $s(i)$ be the minimum # of stakes needed to reach checkpoint n from i

$s(n) = 0$

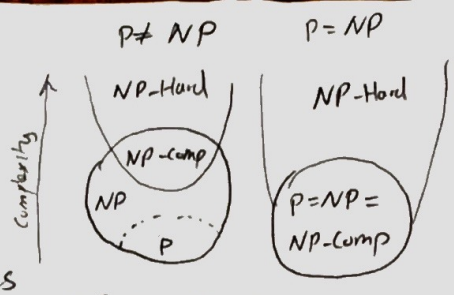
$s(i) = 1 + \min_{i+1 \le j \le i+d(i)} s(j)$ for $i < n$

n entries

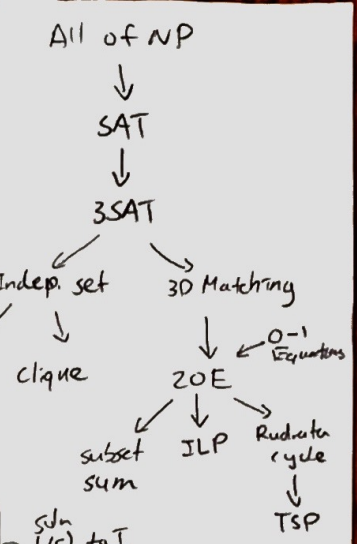
each takes $O(1)$

↳ $O(n)$

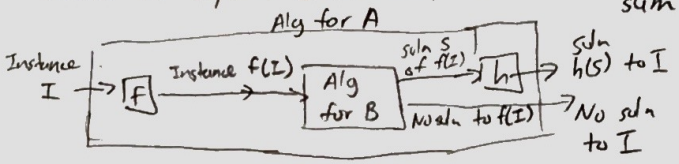
NP-Hard Problems (NP-Comp)	Easy Problems (in P)
3SAT	2SAT, HORN SAT
TSP	MST
Longest path	shortest path
3D Matching	bipartite matching
Knapsack	unary knapsack
Independent set	Independent sets on trees
Integer LP	LP
Rudrata Path	Euler Path
Balanced Cut	Minimum cut



If $A \to B$, B is at least as hard as A.



If any search problem can be reduced to B, then B is NP-Complete



To prove reduction;
1. Prove pre process and post process polynomial
2. Given B soln S_B, $h(S_B)$ is valid soln to A
3. No soln to B → no soln to A
   or soln to A, then there exists solution to B

NP-decision problems for which instances where ans is yes can be verified in poly time

NP-hard: If there's an NP-comp problem that reduces to it

Quantum

$|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$

$|\alpha\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$

$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$

Partial Measure: look at one bit

Result: 0 (w/ prob $|\alpha_{00}|^2 + |\alpha_{01}|^2$)

new internal state: $\dfrac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$

scale to make probs add to 1

$\alpha_0|0\rangle + \alpha_1|1\rangle \quad \beta_0|0\rangle + \beta_1|1\rangle$

Joint state: $\alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$

not all can be decomposed. eg. $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$

Hadamard Gate

$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad |1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

Two bits $H(\alpha_0|0\rangle + \alpha_1|1\rangle) = \dfrac{\alpha_0+\alpha_1}{\sqrt{2}}|0\rangle + \dfrac{\alpha_0-\alpha_1}{\sqrt{2}}|1\rangle$

CNOT

takes two qubits and flips the target bit if the control bit is set to 1

Modular If $x \equiv x' \pmod N$ and $y \equiv y' \pmod N$

$x+y \equiv x'+y' \pmod N$ and $xy \equiv x'y' \pmod N$

If x, y are pos int and $x \geq y$, $\gcd(x,y) = \gcd(x \bmod y, y)$

Fermat Little Theorem $a^{p-1} \equiv 1 \pmod p$ if p is prime $1 \leq a < p$

Carmichael #'s – passes Fermat but not prime

MWU probability of choosing i is $\frac{w(i)}{W}$ where W = total weight

# of mistakes M made by MWU w/ factor of $(1-\varepsilon)$ is

$$M \leq (1+\varepsilon)m + \frac{\ln n}{\varepsilon}$$ take deriv and set to 0 to find ε

m is the # of mistakes of the best expert

$$\text{Regret } R = \frac{1}{T}\left(\sum_{t=1}^{T} c(\vec{x}_t) - \min_k \sum_{t=1}^{T} c(x_t, k)\right)$$

larger ε = more penalty for mistakes

To prove A is NP-Comp/problem must be in NP

2. Find B in NP-Hard s.t. $B \to A$

$NP \to B \quad B \to A \quad NP \to A$

Coping w/ NP-Completeness

Backtracking: Try an assignment, test if it meets our constraint. If not, reject.

Branch and Bound: To reject a subproblem in a minimization prob, we must be certain that its cost exceeds some other soln

Approximation Alg: For a min. prob, find a soln w/ factor $\alpha_A = \max_I \frac{A(I)}{OPT(I)}$ away from the optimum.

local search: If we have a set of soln, we def a neighborhood struct to relate them, then we seek local optima

Matching: subset of edges that have no vertices in common, any matching is a lower bound on the optimal soln for vertex cover.

TSP – Is there a tour, which visits each node exactly once, within budget b.

Rudrata path = path passing thru each vertex exactly once

Balanced cut: with Budget b for cut, partition into 2 sets such that each has $\geq 1/3$ elems

3D matching: Given M valid tuples, match n boys, girls, pets. Find n disjoint triples that get along

Independent set: Graph, integer g: find g vertices that are Independent (no 2 share edge)

Vertex cover: b vertices that touch any edge

Dominating set: b vertices s.t. every edge is in the set, or has a neighbor in the set

Set cover: given subsets, select b subsets such that the union is the entire set

Clique: g vertices, each is connected w/ every other

Longest path: is there a simple path, length g or more, from s to t?

Hashing

$n = 257$

IP addresses – $h_a(x_1, \ldots x_4) = \sum_{i=1}^{4} a_i \cdot x_i \bmod n$

pick a_i at random

If two IP differs at $x_4 \neq y_4$ then:

$$\sum_{i=1}^{3} a_i (x_i - y_i) \equiv a_4 \cdot (y_4 - x_4) \bmod n$$

n is prime so $(y_4 - x_4)$ must have unique inverse

Set LHS to c, then a_4 needs to be $c(y_4 - x_4)^{-1} \bmod n$ which has probability $1/n$.

Universal hash := $\forall x, y \in U, x \neq y$: $\Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}$

$H = \{h : U \to [m]\}$

$f : [m] \to [n]$ then n^m possible functions

Set Packing

Find k mutually disjoint subsets

Proof: Reduce 3D Matching to set packing. Given an instance Φ of 3D matching, we can construct instance Ψ of set packing. Let the set $U = \{p_1, \ldots p_n, b_1, \ldots b_n, g_1, \ldots g_n\}$ be the set of all pets, boys, girls in Φ and let $k = n$. For each tuple (p_i, b_j, g_k) we create one set $S_e = \{p_i, b_j, g_k\}$. The reduction is in polynomial time and a collection of tuples is a valid matching for Φ iff their corresponding sets is a valid set packing solution.

Reduce Hamiltonian Path to degree-bounded Spanning Tree. we can make it into a DBST problem w/ graph G and $b_i = 2 \; \forall i$. If there is a Hamiltonian Path, then this is a spanning tree where each deg is at most 2, so it must be the case that DBST has a possible spanning tree where each degree is at most 2.

If there is a ST where deg is at most 2, then this implies there is a Hamiltonian path. Because it is spanning, it includes all vertices, and it's acyclic (tree). If every deg is at most 2, this forces the only possible ST to be a path that visits all the vertices exactly once. This is a Ham Path. So if there is such a ST, there is a Ham Path.

Streaming

F_0 - counts # of distinct items F_1 - length of stream

To estimate item freq f_j within additive error n/k using $O(k(\log n + \log m))$ mem,

1. Maintain set S of k counters, For each elem x_i
2. If $x_i \in S$, increment counter for x_i
3. If $x_i \notin S$, add x_i to S if space avail, else decrement all counters in S. (Remove when 0)

Exact counts require $O(n)$ space

Tell if # of distinct items N is more than $2k$

1. Choose uniformly random hash func $h : [n] \to [B]$, where # of Buckets $B = O(k)$
2. Output 'yes' if there is some $x_i \in S$ s.t. $h(x_i) = 0$ else 'no'

When $k = 2$, show this is a 3/4 approximation

Let OPT denote the total # of elems covered by the optimal soln and let x denote the # of elems covered by the first subset in the greedy soln. Since OPT elements can be covered by two subsets, the first subset picked by the greedy must cover at least OPT/2 elems. $x \geq \frac{OPT}{2}$ since at least $OPT - x$ elems covered by the opt soln is uncovered by the first subset picked by the greedy, the second subset must cover at least $(OPT - x)/2$ uncovered elems. So greedy covers $x + (OPT - x)/2 = OPT/2 + x/2 \geq 3OPT/4$

Simplex is exponential time

Doubling the min-cut doesn't double the flow

Two polynomial $f(x), g(x)$ given in (point, value) can be multiplied in $O(n)$

Every directed graph can be decomposed into a DAG

Cross edges occur in DFS only if graph directed

Running DFS from a node in a sink SCC yields a SCC of the graph

MST never contains heaviest edge of cycle

MST always contains lightest edge in every cut