さすが

| | | |
|---|---|---|
| <u>Dijkstra</u> SSSP $O(m \log n)$ or $O(m + n \log n)$ (fibheap)<br><br>while nodes unvisited:<br>visit cheapest n, update neighbor costs to $min(old, n + edge)$<br><br>Don't revisit visited nodes => can't handle negative edges | <u>Bellman-Ford</u> SSSP $D_{v,n-1}$ $O(mn)$<br>$0 \quad \text{if } k = 0 \text{ and } v = s$<br>$\infty \quad \text{if } k = 0 \text{ and } v \neq s$<br>$min\{D_{v,k-1}, \min_{x \in N(v)} D_{x,k-1} + len(x,v)\} \text{ else}$<br>"Extend one edge at a time"<br>Can detect negative cycles. | <u>TSP</u> $T(n) = O(n^2 2^n)$, $S(n) = O(n2^n)$<br>$len(x,t) \quad \text{if } S = \{x,t\}$<br>$\min_{t' \in S, t' \neq t, t' \neq x} C(S - t, t') + len(t', t) \text{ else}$ |
| <u>Matrix</u> APSP $O(n^3 \log n)$<br>$B_{ij} = min_k\{A_{ik} + A_{kj}\}$ ($\leq 2$ edges)<br>$C = B \times B$ ($\leq 4$ edges), ...<br>$O(\log n)$ squarings, mult is $O(n^3)$<br><br><u>Floyd-Warshall</u> APSP $O(n^3)$<br>for k in [1,n]: forall i,j:<br>$A_{ij} = min\{A_{ij}, A_{ik} + A_{kj}\}$ | <u>Johnson</u> APSP $O(mn + n^2 \log n)$<br>Add dummy node with len 0 to every other, run Bellman to find shortest path, add that length of shortest path to all so that nonnegative, run Dijkstra from every node | <instructor photos for good luck elided><br>$e^x = \lim_{n \to \infty}(1 + \frac{1}{n})^n$, $e^{-1} = \frac{1}{e} = \lim_{n \to \infty}(1 - \frac{1}{n})^n$<br>Hence $(1 - \frac{1}{m})^m < \frac{1}{e}$ |

**Capacity Constraint** $\forall e, f(e) \leq c(e)$
**Flow Conservation** $\forall v \notin \{s,t\}, \Sigma_u f(u,v) = \Sigma_u f(v,u)$
**s-t Cut** Partition of vertex set into $A$ and $B$ such that $s \in A$ and $t \in B$; cut **capacity** is $\Sigma_{edges\ from\ A\ to\ B}\, c(e)$
**Skew-symmetry** $f(u,v) = -f(v,u)$ allows us to add flows together
**Residual Capacity** $c_f(u,v) = c(u,v) - f(u,v)$
**Augmenting Path** s-t path of positive residual capacity
**Maxflow-Mincut** any s-t flow $\leq$ maximum s-t flow $\leq$ minimum s-t cut $\leq$ any s-t cut [saturated edges], true for (ir)rational caps
**Integral Flow** if all capacities are integers, exists maximum flow in which all flows are integers
**Bipartite Matching** s to L, t to R, all edges capacity 1, use Ford-Fulkerson
**Max-flow Min-cost** Edges have cost $w(e)$ as well as capacities $c(e)$. Modify Ford-Fulkerson, always pick least-cost path

| Algorithm | Description | Runtime | Notes [$F$ = max s-t flow] |
|---|---|---|---|
| Ford-Fulkerson | while $\exists$ augmenting path:<br>push max flow along path | $O(F(m+n))$<br>$F$ iterations, $(m+n)$ DFS | Rational capacities OK<br>Irrational can be wrong/loops |
| Edmonds-Karp1 | Pick largest capacity path | $O(m^2 \log F)$<br>$O(m \log F)$ iterations | $\exists$ s-t path with capacity at least $F/m$;<br>Path found by binary search on answer method of finding max capacity then use dfs |
| Edmonds-Karp2 | Pick shortest path | $O(nm^2)$<br>At most $nm$ iterations | $d(s,t)$ never decreases, increases by $\geq 1$ every $m$ iterations (BFS levels) |

**Linear Programs** {variables, objective, constraints}, either *Infeasible*, *Feasible and Bounded*, or *Feasible and Unbounded*
**Von-Neumann's Theorem**: lb = ub for all **finite, 2 player** zero sum games
**Existence of Stable Strategies**: Every **finite** player game (with each player having a **finite** number of strategies) has at least one (mixed-strategy) Nash equilibrium.

| | | |
|---|---|---|
| By convention, payoff matrix is row-player<br>$\begin{matrix} -1/2 & 3/4 \\ 1 & -3/2 \end{matrix}$ | **Row Player** picks max(min(col))<br>$\begin{matrix} -1/2 & 3/4 \\ 1 & -3/2 \end{matrix}$<br>max(min(-0.5p+(1-p), 0.75p-1.5(1-p))) | **Col Player** picks min(max(row))<br>$\begin{matrix} -1/2 & 3/4 \\ 1 & -3/2 \end{matrix}$<br>min(max(-0.5p+0.75(1-p), p-1.5(1-p))) |

さすが

**Simplex** Jump to better neighboring corners, this works because feasible set is convex, worst case **exponential** number of corners (Klee-Minty cube: fix $\epsilon \in [0, 0.5]$, $0 \le x_1 \le 1$, $\epsilon x_i \le x_{i+1} \le 1 - \epsilon x_i$ creates hypercube with $2^d$ corners)

**Seidel** m constraints, d dimensional variables - RANDOMLY add the constraints one by one, keep track of optimal so far, adjust if necessary. Expected: $O(d!m)$, Worst Case: $O(d!m^2)$

**Ellipsoid** binary search w/ ellipsoids, runs in **polytime** wrt n, d, L (constraints, dimensions, bits to represent coefficients)

**Karmarkar (kumarkar - a n g e r y)** polytime interior-point method

| **Primal** | **Dual** (note: Dual(Dual) = Primal) | P/D I O U | |
|---|---|---|---|
| maximize $c^T x$ | minimize $y^T b$ | I ✓ x ✓ | ✓ possible, x impossible |
| subject to $Ax \le b,\ x \ge 0$ | subject to $y^T A \ge c^T,\ y \ge 0$ | O x ✓ x | I infeasible |
| | | U ✓ x x | O feasible, bounded |
| | | | U unbounded |

**Weak Duality** $x$ feasible for primal, $y$ feasible for dual => $c^T x \le y^T b$

**Strong Duality** primal feasible and bounded => dual feasible and bounded, and $c^T x^* = (y^*)^T b$ (max primal = min dual)

**Polynomial Time** for some constant $c$, runtime is $O(n^c)$ where $n$ is input size

**Poly-Time Reducible** $A \le_p B$ (A polytime reducible to B) if can solve A in polytime given polytime blackbox for B

**Many-one (Karp) Reduction** from A to B: polytime-computable $f$ st $x \in Y_A \Rightarrow f(x) \in Y_B$ and $x \in N_A \Rightarrow f(x) \in N_B$

**P** decision problems solvable in polytime

**NP** decision problems w polytime verifiers, $\exists V(I, X)$ polytime st $I \in Y \Rightarrow \exists X,\ V(I, X) = Y$ and $I \in N \Rightarrow \forall X,\ V(I, X) = N$

**NP-Complete** $Q$ if 1. $Q \in$ **NP** and 2. for all $Q' \in$ **NP**, $Q' \le_p Q$. If only 2. satisfied, we call that **NP-Hard**

**NP-Complete PkProblems :thonk:**

| | | | |
|---|---|---|---|
| Circuit-SAT | Hamilton path | Integer LP | 3-COL |
| 3-SAT | Partition | Binary integer LP | k-COL |
| Vertex cover | TSP | Subset sum | Independent set |
| Clique | | | Set cover |

**Scheduling** Schedule $n$ jobs w times $p_j > 0$ on $m$ machines, $min\ max_i\ \Sigma_{j \in S_i} p_j$ ; greedy 2OPT, sorted greedy 1.5 to 4/3 OPT

**Vertex Cover** Both 2OPT: take both endpoints of arbitrary edge; LP where $0 \le x_i \le 1,\ min\ \Sigma_i x_i$ and round (simply taking highest degree vertex can lead to log(n) times worse solution)

**Set Cover** Greedy is $O(k \ln n)$, more precisely at most $k \ln \frac{n}{k} + k$ sets $[n(1 - \frac{1}{k})^{k \ln n} < n(\frac{1}{e})^{\ln n} = 1]$

**Competitive Ratio (CR)** Worst-case over possible future $\sigma$, i.e. $max_\sigma \frac{ALG(\sigma)}{OPT(\sigma)}$

**Better-Late-Than-Never** has $CR \le 2$, if $p = kr$ then $CR = 2 - \frac{r}{p}$. If modeled as zero-sum game, close to $CR = \frac{e}{e-1}$

**Elevator** CR: 2 - E/S is optimal, If E << S, with randomization can get $CR = \frac{e}{e-1}$

**List Update** 4-competitive

| | |
|---|---|
| $\begin{aligned} &\max(x_1 + 3x_2 - 2x_3) \\ \text{s.t.}\quad & x_1 + x_2 + 2x_3 \le 2 \\ & 7x_1 + 2x_2 + 5x_3 \le 6 \\ & 2x_1 + x_2 - x_3 \le 1 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$ | $\begin{aligned} &\min(2y_1 + 6y_2 + 1y_3) \\ \text{s.t.}\quad & y_1 + 7\ y_2 + 2\ y_3 \ge 1 \\ & y_1 + 2\ y_2 + 1\ y_3 \ge 3 \\ & 2\ y_1 + 5\ y_2 + (-1)\ y_3 \ge -2 \\ & y_1, y_2, y_3 \ge 0 \end{aligned}$ |

$\mathcal{P}_1$

$$\max_x \min_j \sum_i x_i U_{ij}^1 \quad \text{s.t.} \sum_i x_i = 1,\ x_i \ge 0$$

$\mathcal{P}_2$ -- LP

$$\max_{x,v} v \quad \text{s.t. } v \le \sum_i x_i U_{ij}^1, \forall j \quad \sum_i x_i = 1 \quad x_i \ge 0$$