MT2: "Efficient Algorithms II"

- DP = needs DAG-type structure.
  - knapsack
  - TSP
    - $dp[S][i]$ = min. cost. path visiting each vertex in S, from 1 to i.

$dp[\{1\}][1] = 0$

else: $dp[S][i] = \min_{j}\left(dp[S-\{i\}][j] + dist(j,i)\right)$ $\quad O(n2^n)$

  - edit distance → add 1 or both chars.
  - other probs.

- LP
  - want to maximize some linear combo of variables
  - also want to satisfy some constraints

$\Rightarrow$ maximize $c^T x$ s.t. $Ax \geq b$.

— duality
- make some new variables

$\Rightarrow$ maximize $b^T y$ s.t. $Ay \geq c$.

Weak Duality: value of feasible sol $\vec{x}$ to primal LP must be $\leq \vec{y}$ of dual LP.
Strong Duality: same as above, but strictly equal.

- reduction

↳ NP section.

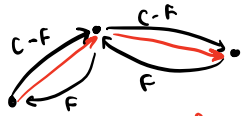
- games (zero-sum)
  - input: a payoff matrix M
  - come up w/ strategies, mixed strategy is just LP over p_i & z where z is total score & u try to maximize z.

Min-Max Thm: $\max_p(\min_q(score(p,q))) = \min_q(\max_p(score(p,q)))$.

- Multiplicative Weights
  - n experts $E_1, \dots, E_n$ → we pick an expert i on day t w/prob $p_i^{(t)}$.
  - each of them has a prediction, either right or wrong.
    - incur a loss $l_j^{(i)}$ depending on day i correctness for expert j.
  - expected loss on t^{th} day is $q^{(t)} = \sum_{i=1}^{n} p_i^{(t)} \cdot l_i^{(t)}$
    - after T days, exp. loss = $A^T = \sum_{t=1}^{T} q^{(t)}$. And loss of expert E_i is $L_i^T = \sum_{t=1}^{T} l_i^{(t)}$.
  - regret = $A^T - \min L_i^T$.

Multiplicative Weights Algo:
- we update weight $p_i^{(t+1)} = p_i^{(t)}(1-\varepsilon)^{l(t)}$ for loss function $l(t)$.
- Regret bounded by $R_T \leq \varepsilon T + \frac{\ln n}{\varepsilon}$

$\Rightarrow R_T \leq 2\sqrt{T \ln n}$ if $\varepsilon = \sqrt{\frac{\ln n}{T}}$.

- Max flow
  - graph w/ edges capacity & you want to max. total capacity
  - want to find max. amt of flow you can send from vertices s to t.
  - 

(residual graph)

capacity-flow sent forward, flow sent backward

  - min-cut max-flow thm: min. cut = max. flow. where cut is edges b/w a set S & a set $T = V \setminus S$.
  - Ford-Fulkerson: just go run DFS on residual graph repeatedly.

Final: "Intractible Problems"

- P vs. NP
  - P: can be solved in poly. time
  - NP: solution can be validated in poly. time.
- NP-hardness & completeness
  - NP-hard. All problems at least as hard as hardest NP problem
  - NP-complete. All problems in both NP & NP hard: the hardest of the NPs.
- Reductions
  - Reduction from A to B
  - ⇒ If we solve B, we have an efficient way to solve A.
  - If ∃ reduction from A to B, then we know A is at least as hard as B.
- Approximation Algorithms
  - Can sacrifice some accuracy for efficiency.

    many NP-C problems can be approx'ed decently well in poly. time.
- (S)treaming
  - have a stream of info
  - can process it entirely only once, from left to right.
  - info can be passed into one or multiple algos or stored.
  - goal's to use around log n mem.

  → randomized algorithms help save memory

  Distinct Values: Makes a random hash function and check t smallest hashes

  Sum: add 2×RV w/ ev of .5.
- Search, Decision, Optimization
  - search: find a solution given inputs, or return none
  - decision: see if solution exists given inputs.
  - optimization: find optimal solution given inputs, or return none
- What's in P / NP?