Classics

Union Find Disjoint Set (UFDS)

$\text{rank}(x)$: height of subtree below node x (excl)

- $\forall x\ \text{rank}(x) \le \text{rank}(\pi(x))$ with equality if and only if $x = \pi(x)$ ($\pi(x) =$ parent of x)
- Any root node of $\text{rank}(k)$ has $\ge 2^k$ nodes in its subtree (excl)
- For n nodes, at most $\frac{n}{2^k}$ nodes of rank k
- [Corollary] All nodes have rank $\le \log n$. Hence, *find* and *union* takes $O(\log n)$

Path compression: a sequence of n *make_sets* and m *union* / *find* takes $O(n + m \cdot \alpha(m, n))$

Proof for $O((m+n)\log^* n)$ ($\log^*(2 \uparrow i) = i$):

- Group nodes by rank; consider intervals $[0], [1], (1,2), \dots, (2 \uparrow i, 2 \uparrow (i+1)], \dots$
- Largest nonempty interval $i = \log^* n - 1$
- # of nodes in $(2 \uparrow i, 2 \uparrow (i+1)] \le \frac{n}{2 \uparrow i}$

Set Cover

Given $U = \{1, \dots, n\}$, subsets $S_1, \dots, S_m \subset U$ s.t $\cup_{i=1}^{m} S_i = U$, find collection of minimal cardinality $J = \{S_i\}$ s.t. $\cup_{i \in J} S_i = U$.

Greedy algorithm: Repeatedly pick S_i that covers the maximum # of uncovered elements

Theorem: If optimal solution uses k sets, Greedy uses at most $k \ln n$ sets. Greedy is the best efficient approximation for Set Cover.

Key idea: Let n_t be number of elements not covered after t iterations of Greedy. Suffices to show $n_{k \ln n} < 1$. Claim: $n_{t+1} \le n_t \left(1 - \frac{1}{k}\right)$

Linear Programming

Definitions:

- Vertex: x s.t. x is **feasible** and n of inequalities are tight (n is dimension of the problem i.e. $x \in \mathbb{R}^n$)
  - For every subset of n constraints, solve for point of intersection; check feasibility
- Unbounded: objective value $p^* = \infty$
  - Primal unbounded ⇒ Dual infeasible and has no vertices
  - Can check for boundedness by using linear combinations of constraints
- Feasible: solution $p^* < \infty$
  - Primal feasible ⇒ Dual feasible and lower-bounded
  - Dual feasible ⇒ Primal upper-bounded

Facts:

- Feasible region of LP is always convex
- ∀LP, $\exists x$ a solution that is a vertex
  - Find all vertices, get optimal value

Duality

Primal LP	Dual LP
$\max_x c^T x$ subject to $Ax \le b$ and $x \ge 0$	$\min_y b^T y$ subject to $A^T y \ge c$, $y \ge 0$

Max Flow

Algorithm:

- Start with zero flow
- Repeat: choose viable path from s to t; increase flow along the edges as much as possible; change weights of residue graph

$G^f = (V, E^f)$ is the residual graph of $G = (V, E)$

$$f(x) = \begin{cases} c_{uv} - f_{uv}, & (u,v) \in E, f_{uv} \le c_{uv} \\ f_{uv}, & (u,v) \in E, f_{uv} > 0 \end{cases}$$

Definitions:

- $s - t$ cut: (L, R) s.t. $L \cup R = V$, $s \in L$, $t \in R$
- Capacity of (L, R): $\sum_{u \to v : u \in L, v \in R} c_{u \to v}$

Theorems and Facts:

- Max-Flow Min-Cut: Size of maximum flow in a network = capacity of smallest $s - t$ cut
- For any flow f and any cut (L, R), $f \le \text{capacity}(L, R)$
- If all capacities $\in \mathbb{Z}$, maximum flow is integral
- No flow of value k from s to t ⇒ min cut has capacity $< k$ with $s \in S, t \in T$
- Flow of value k from s to t ⇒ min cut has capacity $\ge k$

Techniques:

- Phantom source, sink, nodes and edges

$\min_x c^T x$ subject to $Ax \geq b$ and $x \geq 0$	$\max_y b^T y$ subject to $A^T y \leq c$, $y \geq 0$

Duality Theorems:

- [Weak] Objective value of any feasible solution to primal ≤ objective value of any feasible solution to dual i.e. $p^* \leq d^*$
  - Solutions of dual upper bounds primal
  - Solutions of primal lower bounds dual
- [Strong] If $p^* < \infty$, then $p^* = d^*$.
  - If LP has bounded optimum p^*, so does its dual, and $p^* = d^*$

Interpretation of y: Lagrange multiplier for constraints; i.e. weights the importance placed on each constraint.

Primal	Dual
$\max_x c_1x_1 + \cdots + c_nx_n$ $a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i$ for $i \in I$ $a_{i1}x_1 + \cdots + a_{in}x_n = b_i$ for $i \in E$ $x_j \geq 0$ for $j \in N$	$\max_y b_1y_1 + \cdots + b_my_m$ $a_{1j}y_1 + \cdots + a_{mj}y_m \geq c_j$ for $j \in N$ $a_{1j}y_1 + \cdots + a_{mj}y_m = c_j$ for $j \notin N$ $y_i \geq 0$ for $i \in I$

- Bipartite matching: have phantom nodes s, t, find max flow from s to t. ∃matching ⇔ can send n units of flow
- Hall's Marriage Lemma

LP Algorithms (Simplex, Ellipsoid, Interior Point)

SIMPLEX(v): # returns optimal vertex
  while $obj(v') > obj(v)$ and v' neighbor of v:
    return SIMPLEX(v')
  return v

- Two vertices $\in \mathbb{R}^n$ are neighbors if they share $n-1$ inequalities in common
- Number of neighbors of v in LP with n variables and m constraints $\leq (m-n) \cdot n$
- $O\left(\binom{m+n}{n} mn\right)$ (worst case exponential time)
- Problems with Simplex and Solutions
  - Starting vertex (find by transforming LP)
  - Degeneracy (introduce perturbation)
  - Unboundedness (opt increases to ∞; halt and complain)

Ellipsoid algorithm and Interior Point method are polynomial time algorithms

Zero Sum Games

For game matrix $G = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, expected payoff $= \sum_{i,j} G_{ij}\, \mathbb{P}[R = i, C = j]$ (R plays i, C plays j)

Theorems and Facts:

- Once a player's strategy is fixed, there is a optimal pure strategy for the other player.
- [Von Neumann] For $A \in \mathbb{R}^{m \times n}$

$$\max_{x \in X} \min_{y \in Y} x^T A y = \min_{y \in Y} \max_{x \in X} x^T A y$$

$$\max_x \min_y \sum_{i,j} G_{ij} x_i y_j = \min_y \max_x \sum_{i,j} G_{ij} x_i y_j$$

Linear Programs

Game #1: Row announces her strategy p

$$\max_{p_1, p_2 : p_1 + p_2 = 1, p_1, p_2 \geq 0} \min(ap_1 + cp_2, bp_1 + dp_2)$$

$$\max_p \min_{q \text{ mixed}} p^T A q = \max_p \min_{q \text{ pure}} p^T A q$$

Game #2: Column announces her strategy q

$$\min_{q_1, q_2 : q_1 + q_2 = 1, q_1, q_2 \geq 0} \max(aq_1 + bq_2, cq_1 + dq_2)$$

$$\min_q \max_{p \text{ mixed}} p^T A q = \min_q \max_{p \text{ pure}} p^T A q$$

Note: In $\min_c \max_r P(r, c)$, row goes second ⇒ has more information.

Multiplicative Weights Update

Problem: n experts $E_1, \ldots, E_n$. On tth day, expert E_i incurs a loss $l_i^{(t)} \in \{0,1\}$. Pick E_j and incurs loss $l_j^{(t)}$. Want to minimize regret after T days = Total loss – Total loss of best expert in hindsight

At tth day, for distribution $x^{(t)} = \left(x_1^{(t)}, \ldots, x_n^{(t)}\right)$

- Loss of algorithm on day $t = \sum_{i=1}^n x_i^{(t)} \cdot l_i^{(t)}$
- Total loss over T days $= \sum_{i=1}^T \sum_{i=1}^n x_i^{(t)} \cdot l_i^{(t)}$
- $R_T = \sum_{i=1}^T \sum_{i=1}^n x_i^{(t)} \cdot l_i^{(t)} - \min_{1 \leq i \leq n} \sum_{t=1}^T l_i^{(t)}$
- Exists an algorithm s.t. $\lim_{T \to \infty} R = \lim_{T \to \infty} \frac{R_T}{T} = 0$

Key Idea:

- Fix ϵ; $\forall i$, $w_i^{(0)} = 1$
- For each expect, assign weight $w_i^{(t)}$ (weight of expert i on day t)
- On day t, $\mathbb{P}[\text{pick } i] = x_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i=1}^n w_i^{(t)}}$
- At end of day t, update weights via: $w_i^{(t+1)} = w_i^{(t)}(1-\epsilon)^{l_i^{(t)}}$

Results:

Problem Solving	
• Think DP (including change of states), Greedy, Graphs, DnC; on the fly trick • Check LPs; don't be tricked by the direction of inequalities • Is the question asking about dual or primal problem? • Huffman coding: inequality of leaf nodes does not matter	• After T days, regret $R_T \leq \epsilon T + \frac{\ln n}{\epsilon}$ • Average regret $R \in O\left(\sqrt{\frac{\ln n}{T}}\right)$ <u>Variants and Corollaries</u>: • Online algorithm • Constant # of variables per expert (decreasing n lowers memory) • # of timesteps affect runtime of algorithm • ϵ affects calculations of weights If losses $\in [a, b]$, $R_T \leq (b-a)\left(\epsilon T + \frac{\ln n}{\epsilon}\right)$

Appendix of Pseudo-codes and DP Recurrences

(for quick referencing)

All Pairs Shortest Path $O(n^3)$

$\mathrm{d}[i,j,k]$ denotes length of shortest $i \to j$ path where intermediate vertices are in $\{1, \dots, k\}$

- $\mathrm{d}[i,j,0] = w(i,j)$
- $\mathrm{d}[i,j,k] = \min(\mathrm{d}[i,j,k-1], \mathrm{d}[i,k,k-1] + \mathrm{d}[k,j,k-1])$

APSP(G):
  for $(i,j) \in E$: $\mathrm{d}[i,j,0] \leftarrow w(i,j)$
  for $(i,j) \notin E$: $\mathrm{d}[i,j,0] \leftarrow \infty$
  for $k = 1, \dots, n$:
    for $i = 1, \dots, n$:
      for $j = 1, \dots, n$:
        $\mathrm{d}[i,j,k] \leftarrow \min(\mathrm{d}[i,j,k-1], \mathrm{d}[i,k,k-1] + \mathrm{d}[k,j,k-1])$
  return d

Shortest Path in DAG $O(n+m)$

$\mathrm{dp}[i]$ denotes length of shortest $s \to i$ path

$$\mathrm{dp}[v] = \min_{u:(u,v)\in E}\{\mathrm{dp}[u] + w(u,v)\}$$

SP_DAG(G, s):
  $\forall u\ \mathrm{dp}[u] \leftarrow \infty$
  $\mathrm{dp}[s] \leftarrow 0$
  for $v \in G$ in linearized order:
    $\mathrm{dp}[v] \leftarrow \min_{u:(u,v)\in E}\{\mathrm{dp}[u] + w(u,v)\}$
  return $\mathrm{dp}[t]$

Shortest Path Problem $O\big(k(|V|+|E|)\big)$

Given directed $G = (V, E)$, edge weights can be positive or negative, $s, t \in V$, integer k, output length of shortest path using at most k edges

$$dp(v,i) = \min\left(\min_{u:(u,v)\in E}\{dp(u,i-1) + w(u,v)\}, dp(v,i-1)\right)$$

SHORTEST_K_PATH(G, s, t, k):
  $\mathrm{dist}[v,0] \leftarrow \infty$
  $\mathrm{dist}[s,0] \leftarrow 0$
  for $i = 1, \dots, K$:
    for $v \in V$:

$$\mathrm{dist}[v,i] \leftarrow \min\left(\min_{u:(u,v)\in E}\{\mathrm{dist}(u,i-1) + w(u,v)\}, \mathrm{dist}(v,i-1)\right)$$

  return $\mathrm{dist}[t,k]$

Remark: For shortest path, return $\mathrm{dist}[t, n-1]$

Travelling Salesman Problem $O(2^n \cdot n^2)$

Problem: find shortest tour starting at 1 (i.e. visits every node exactly once and ends at 1)

For S s.t. $i \in S$, denote $T[S,i]$ as the length of shortest path that starts at 1, visits every node in subset S, ends at i

$$T[s,i] = \min_{j\in S\setminus\{i\}}\{T[S\setminus\{i\}, j] + d_{ij}\}$$

TSP(G):
  $T[\{1\}, 1] \leftarrow 0$
  for $s = 2, \dots, n$:
    for S s.t. $|S| = s$, $1 \in S$:
      for $i \in S$:
        $T[S,i] \leftarrow \min_{j\in S, j\neq i}\{T[S\setminus\{i\}, j] + d_{ij}\}$

Maximum Independent Set (Trees) $O(|V|+|E|)$

Find S s.t. for $u, v \in S$, $(u,v) \notin E$

$I(v)$ = size of largest independent set in T_v (the subtree rooted v)

$$I(v) = \max\left\{1 + \sum_{w\in grandchild} I(w), \sum_{u\in child} I(u)\right\}$$

Horn SAT

HORN_SAT(X):
  $\forall i,\ x_i \leftarrow \mathrm{False}$
  while $\exists$ unsatisfied $(x_i \wedge \dots \wedge x_j) \Rightarrow x_k$:
    $x_k \leftarrow \mathrm{True}$
  if every $(\bar{x}_i \vee \dots \bar{x}_j)$ satisfied:
    return $(x_1, \dots, x_n)$
  else return "not satisfiable"

Longest Increasing Subsequence $O(|a|^2)$

Denote $L[j]$ as the length of the longest LIS in $a_1, \dots, a_j$ that ends in a_j

$$L[x] = 1 + \max_{i<x}\{L[x]: a_i < a_x\}$$

LIS(a):
- for $x = 1, \dots, n$:
  - if $\exists i < x$ s.t. $a_i < a_x$:
    - $L[x] \leftarrow 1 + \max_{i<x}\{L[x]: a_i < a_x\}$
  - else: $L[x] \leftarrow 1$
- return $\max_{x\in\{1,\dots n\}} L[x]$

Edit Distance $O(|s||t|)$

Edits allowed: insert one character, delete one character, substitute one character

$E[i,j]$ = edit distance of $s[1, \dots, i]$, $t[1, \dots, j]$

$$E[i,j] = \min\{1 + E(i-1,j), 1 + E(i,j-1), E(i-1,j-1) + \text{diff}(i,j)\}$$

$$diff(i,j) = \begin{cases} 0, & s[i] = t[j] \\ 1, & \text{otherwise} \end{cases}$$

EDIT_DISTANCE(s, t):
- for $i = 0, \dots, m$: $E[i,0] \leftarrow i$
- for $j = 0, \dots, n$: $E[0,j] \leftarrow j$
- for $i = 1, \dots, m$:
  - for $j = 1, \dots, n$:
    - $E[i,j] = \min\{E[i-1,j] + 1, E[i,j-1] + 1, E[i-1,j-1] + \text{diff}(i,j)\}$
- return $E[m,n]$

Knapsack with Replacement $O(nW)$

$K(C)$ denote the maximum value that can be achieved with total weight $\leq C$

$$K(C) = \max_{i:w_i\leq C}\{v_i + K(C - w_i)\}$$

KNAPSACK(W):
- $K(0) \leftarrow 0$
- for $C = 1, \dots, W$:
  - $K(C) \leftarrow \max_{i:w_i\leq C}\{v_i + K(C - w_i)\}$
- return $K(W)$

Knapsack without Replacement $O(nW)$

$K(C,j)$ is the max value achievable with total weight C using items $1, \dots, j$

$$K(C,j) = \max\{v_j + K(C - w_j, j-1), K(C, j-1)\}$$

KNAPSACK(W):
- $\forall j\ K(0,j) \leftarrow 0$
- $\forall i\ K(i,0) \leftarrow 0$
- for $j = 1, \dots, n$:
  - for $C = 1, \dots, W$:
    - if $w_j > C$: $K(C,j) \leftarrow K(C, j-1)$
    - else: $K(C,j) \leftarrow \max\{v_j + K(C - w_j, j-1), K(C, j-1)\}$
- return $K(W,n)$