Divide and Conquer: Divide into disjoint subproblems, then combine

- Merge Sort: in $O(n \log n)$
- Count inversion: by augmenting Merge Sort
- Closest Pair: by merging center with delta and 11 cloest y points
- Karatsuba: By replacing $x_1y_2 + x_2y_1$ with $(x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2$; evaluate polynomial at different points
- Strassen: Karatsuba for matrix multiplication

Greedy: Local optimal yields global optimal without memory; Prove optimality by contradiction or induction

- Interval Scheduling: Earliest finish time (replace non EFT by EFT)
- Interval Partitioning: Earliest start time (add one if not capable, argue optimal by depth is precisely the solution)
- Minimizing Lateness: Earliest deadline (argue removing inversion)
- Huffman (Lossless compression): Build prefix free tree by merging two smallest freqs (argue combining 2 smallest freqs remains optimal reversely)

Dynamic Programming: Greedy with memory; Optimal substructure and overlapping subproblems

Let OPT(i) be the optimal solution for the first i items

Let $S(i)$ denote solution for the first i items

Bellman Equation: OPT(i) = Base Case; Recursive Case

Solution: $S(i)$ = emptyset for Base Case; $S(i-1)$ or augmenting depending on Bellman Equation

Top-down: When not all subproblems are needed

Bottom-up: When all subproblems are needed; Prevents recursive call overhead; free memory early

- Weighted Interval Scheduling: Sort by finish time;
- Knapsack: Any sort; 2D array with weight and items; $O(nW)$
- Single Source Shortest Path: 2D array with ending node and number of edges; $O(n^3)$
- Chain Matrix Product: 2D array with starting and ending matrix; $O(n^3)$
- Edit Distance: 2D array with position of matching; $O(nm)$ time and space
- Travelling Salesman: 2D array with remaining node set and starting node; $O(n^2 2^n)$ with space $O(n2^n)$

Network Flow: Max flow = Min cut; Augmenting path; Residual graph; Ford-Fulkerson Source: s; Sink: t; Non-negative capacities c; directed graph

s-t flow is $f : E \to \mathbb{R}_{\geq 0}$; $o \leq f(e) \leq c(e)$; sum of incoming flow equals outgoing flow, $f^{in}(v) = f^{out}(v)$

s-t cut (A, B) is: $s \in A, t \in B$, A and B partition vertices

- Ford-Fulkerson: While residual has path. Let bottlenceck(P) be the minimum capacity of edges in path P of residual graph, Augment flow f by sending bottleneck(P) flow along path P; $O((m + n)C)$;
- Edmons-Karp: Ford-Fulkerson with BFS to find shortest path; $O(nm^2)$

$v(f) := f^{in}(t) = f^{out}(s)$

$v(f) = f^{out}(A) - f^{in}(A)$ for all A (prove by summing each a in A, cancel in and out within A)

$Cap(A, B)$: Sum of capacities of edges from A to B; $v(f) \leq Cap(A, B)$ for any st-cut (A, B) by above

- Max Flow Min Cut: $v(f) = Cap(A, B)$ for some st-cut (A, B) (Prove by letting A be reachable nodes from s and B be remaining for final G_f graph of FF, then outgoing edges are saturated (equal to capacity), and entering edges have zero flow otherwise in A, so $v(f) = f^{out}(A) = Cap(A, B)$)

Integrality Theorem: If all capacities are integers, then there exists an integral max flow, same for variants.

For application of network flow, first show 1-1 correspondence between feasible solution and flow by double implication, then

- Bipartite Matching: For bipartite graph, connect s to left, t to right, all edges have capacity 1; Max flow is max matching; $O(nm)$ (flow with value k corresesopnds to a matching with cardinality k)
- Hall's Marriage Theorem: Bipartite graph G has perfect matching iff neighbor $|N(S)| \geq |S|$ for each $S \subseteq U$ (edges with s or t have capacity 1, otherwise infinity, then cut must isolate s or t)
- Edge-Disjiont Paths (Find max num of edge-disjiont paths):(paths are flows by $f(e) = 1$ if edge e is in a path otherwise 0, then with capacities 1, exists unique flow; reversely construct paths by extracting edges from flow);
- Menger's Theorem: The maximum number of edge-disjoint paths from s to t is equal to the minimum number of edges (resp. vertices) whose removal disconnects s and t
- Multiple sources and sinks: Use master source and sink to connect the sources and sinks (with capacities infinity)
- Circulation: Negative supply node v with value k sends $d(v) = k$ more flow than it receives; positive demand node v with value k receives $d(v) = k$ more flow than it sends; zero node is transshipment node; Connect supply nodes with s, demand nodes with t, then circulation exists iff max flow value equals the sum of supplies.
- Circulation with Lower Bounds: For lower bound k between u and v, add k to $d(u)$ and $-k$ to $d(v)$, subtract k from upper bound
- Survey Design: Connect s to each customer with bounds $[c_i, c_i']$, connect each customer to each survey with $[0, 1]$, connect each survey to t with bounds $[p_j, p_j']$, edge from s to t with $[0, \infty]$ (so that all nodes $d(v) = 0$)
- Image Segmentation: Find min cut to segment image into two parts

Linear Programming: Inputs $c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ with n variables and m constraints

Goal is to minimize $c^T x$ subject to $Ax \leq b, x \geq 0$

Flip sign to change $\geq$ to $\leq$, put both $\geq$ and $\leq$ to get equality, negate minimization problem to turn into maximization, replace x with $x' - x''$ to be unconstrained

LP may be infesible or unbounded, but optimal implies vertex optimal.

Slack form: minimize $z = c^T x$ subject to $s = b - Ax, x, s \geq 0$

- Simplex Algorithm: Start with a vertex v of the fesible region, while there is a neighbor v' of v with better objective value, move to v' (worst case exponential)

Start with $x = 0$ assuming $b \geq 0$, then increase a variable x with **positive** coefficient of z until it hits a constraint (subsitute other non-basic or non-leading variables for constraints, then isolate x and substitute ≥ 0), replace that constraint line with x isolated (then treat this variable as a constraint), and replace all occurences of x to the line

repeat until no more entering variable (positive coefficients on z line)

Convert $b \leq 0$ cases:

Start with $Ax \leq b$; change to $Ax + s = b$; flip signs s.t. $A'x + s' = b'$ for $b' \geq 0$; finally change to $A'x + s' + z = b'$ with minimizing $\sum_i z_i$. If 0 is optimal then substitute back to get solution, otherwise infeasible

We operate the constraints (in different ways) to simulate the objective function to get an upper bound, this shows as a certificate of optimality of the solution

I.e. let y be the multiplers, have something like $(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \leq 200y_1 + 300y_2 + 400y_3$, then make left as the objective function, minimize $200y_1 + 300y_2 + 400y_3 = y^T b$ subject to $y_1 + y_3 \geq 1, y_2 + y_3 \geq 6, y \geq 0$

This is called the dual LP (m variables and n constraints), of the primal LP (original)

Dual solution gives upper bound on primal solution

$c^T x = y^T b$ if one of primal or dual LP has solution.

Complexity: NP-complete: in NP and is NP-hard: reduce from a known NP-complete problem

Approximation Algorithms: approximation ratio for maximization is OPT(I)/ALG(I), worst case $\alpha = \max_I\{OPT(I)/ALG(I)\}$

For minimization, ratio ALG(I)/OPT(I)

PTAS: polynomial time approximation scheme; $\forall \varepsilon > 0$, exists $(1+\varepsilon)$-approximation algorithm runs in polynomial time on size n

FPTAS: Fully PTAS, runs in polynomial time on size n and $1/\varepsilon$

Greedy Makespan gives 4/3-1/3m-approximation (longest job first)

Randomized algorithm: $RP : x \in A, 99\%, x \notin A, 0\%, BPP : x \in A, 99\%, x \notin A, 1\%, NP, x \in A >, 0, x \notin A, 0$

Master Theorem: (O, Theta, Omega)

➢ Theorem: Let $a \geq 1$ and $b > 1$ be constants, $f(n)$ be a function, and $T(n)$ be defined on non-negative integers by the recurrence $T(n) \leq a \cdot T\left(\frac{n}{b}\right) + f(n)$, where n/b can be $\left\lceil \frac{n}{b} \right\rceil$.
Let $d = \log_b a$. Then:

- If $f(n) = O(n^{d-\epsilon})$ for some constant $\epsilon > 0$, then $T(n) = O(n^d)$.
- If $f(n) = O(n^d \log^k n)$ for some $k \geq 0$, then $T(n) = O(n^d \log^{k+1} n)$.
- If $f(n) = O(n^{d+\epsilon})$ for some constant $\epsilon > 0$, then $T(n) = O(f(n))$.

Residual Graph:

- Suppose the current flow is f
- Define the residual graph G_f to be the following
  - The vertices are the same
  - For each edge $e = (u, v)$ in G, G_f has at most two edges
    - A forward edge $e = (u, v)$ with capacity $c(e) - f(e)$
      - How much additional flow can we send on e
    - A backward edge $e^{rev} = (v, u)$ with capacity $\boxed{f(e)}$
      - How much we can reduce the flow on e
    - Only add edges of capacity > 0