Divide & Conquer

$T(n) = aT(n/b) + f(n)$
Master theorem: (polynomially greater)

- $f(n) < n^{\log_b a} \Rightarrow T(n) = \Theta(n^{\log_b a})$
- $f(n) = \Theta(n^{\log_b a} \log^k(n))$, $k \geq 0 \Rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1}(n))$
- $n^{\log_b a} < f(n)$, $af(n/b) \leq cf(n) \Rightarrow T(n) = \Theta(f(n))$

Amortized Analysis

Aggregate method: total cost of k operations, then divide by k
Accounting method: pay ahead of ops, pay for expensive ops with savings
Charging method: distribute blame to past ops
Potential method: quantify karma using potential function ϕ mapping states to energies. $c_{amort} = c_{actual} + \phi_{new} - \phi_{old}$

Randomized Algorithms

Las Vegas: always correct, may take longer time
Monte Carlo: may not be optimal, solution quality random, deterministic running time

Markov's Inequality

If Z is non-negative random variable,
$$P(Z \geq a) \leq \frac{E[Z]}{a}$$
Prove using $Z' = a$ iff $Z \geq a$ and 0 else

Chebyshev's Inequality

If Z has finite mean & variance:
$$P(|Z - E[Z]| \geq \epsilon) \leq \frac{\text{Var}[Z]}{\epsilon^2}$$
Prove using $X = (Z - E[Z])^2$, notice that $E[X] = \text{Var}[Z]$, use Markov ineq.

Chernoff Bound

Suppose $Y_1, Y_2, ..., Y_n \in [0, 1]$ are independent random variables. Then $\forall \beta \in [0, 1]$:
$P(\Sigma Y_i > (1 + \beta)E[\Sigma Y_i]) \leq e^{-\beta^2 E[\Sigma Y_i]/3}$
$P(\Sigma Y_i < (1 - \beta)E[\Sigma Y_i]) \leq e^{-\beta^2 E[\Sigma Y_i]/2}$

Union Bound

If $E_1, E_2, ..., E_n$ are events:
$$P(E_1 \text{ or } E_2 \text{ or} \ldots \text{or } E_n) \leq \sum_{i=1}^{n} P(E_i)$$

Basic Probability

Sample space Ω, distribution $m(x)$
$E[X] = \sum_{x \in \Omega} xm(x)$
$E[X + Y] = E[X] + E[Y]$
$E[cX] = cE[X]$

Let $E[X] = \mu$:
$\text{Var}[X] = E[(X - \mu)^2] = E[X^2] - \mu^2$
$\text{Var}[cX] = c^2 \text{Var}[X]$
$\text{Var}[X + c] = \text{Var}[X]$

If X and Y are independent:
$E[XY] = E[X]E[Y]$
$\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y]$

Hashing

$\Theta(1)$ time per op, $\Theta(n)$ space
Load factor: $n/m = \alpha$ where m is num buckets
Hashing w/ chaining: $\Theta(1 + n/m)$
SUHA: assume random input keys,
$P_{k_1 \neq k_2}(h(k_1) = h(k_2)) = 1/m$
Universal hashing: random h (e.g. random matrix), $P_{k_1 \neq k_2}(h(k_1) = h(k_2)) \leq 1/m$
Static dictionaries: $\Theta(1)$ time worst-case, no insert/delete, $\Theta(n)$ space
Perfect hashing: no collisions, e.g. 2-level hashing if h in universal hash family $P(\Sigma n_i^2 > 4n) \leq 1/2$ (prove with Markov's)

Greedy Algorithm

Greedy choice property: locally optimal choices $\rightarrow$ globally optimal soln
Optimal substructure: optimal solution to problem contains optimal solns to subproblems

Gradient Descent

Locally, functions are $\sim$ linear
Go in direction of negative gradient to minimize, keep step sizes small for accuracy
Iteration: $x^{k+1} = x^k - \alpha \nabla f(x^k)$
Convex: $f(y) \geq f(x) + f'(x) \cdot (y - x)$
Stopping criterion: $|f'(x)| \leq \epsilon$ or $||\nabla f(x)|| \leq \epsilon$
Convergence: $f(x^k) - f^* \leq \frac{L(x^0 - x^*)^2}{2k}$, so in k steps we reduce error by $1/k$. To reduce error to ϵ, we need $O(1/\epsilon)$ steps
Linear approximation (Taylor series): $f(x) = f(a) + \frac{f'(a)}{1!}(x - a) + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(a)}{n!}(x - a)^n$
Linear regression: m vectors of size n. If $\alpha \leq 1/L$, after $O(L/\epsilon)$ iterations, the accuracy is $\epsilon||x^0 - x^*||^2$. Total runtime is $O(mnL/\epsilon)$

Matrix Multiplication

Strassen method: $T(n) = 7T(n/2) + O(n^2)$
Algebraic rules: $A+B = B+A$, $A+(B+C) = (A+B)+C$, $A(B+C) = AB+AC$, $c(AB) = (cA)B = A(cB)$, $A(BC) = (AB)C$, $AB \neq BA$

Interval Scheduling

Unweighted: greedy $O(n \log n)$; weighted: dynamic programming $O(n \log n)$ w/ binary search

Closest Pair

Find min. dist. in left & right halves, check points within δ of x-median. $T(n) = 2T(n/2) + O(n) \Rightarrow O(n \log n)$

Fibonacci Numbers

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_n - 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

$A^n = A \cdot (A^{(n-1)/2})^2$ if n is odd, $(A^{n/2})^2$ otherwise.
Multiplying n-bit numbers: $O(n \log n \cdot 2^{O(\log^* n)})$

Median Finding

Pick x by dividing into groups of $n/5$, find median of each, then find median of $n/5$ medians. Find L and G, partitions of L, s.t. $\text{Rank}(x) = |L| + 1$. Then if $\text{Rank}(x) = i$, done; else recurse.
$T(n) = T(n/5) + T(3n/4) + \Theta(n) = \Theta(n)$

Integer Multiplication

$a = 2^{n/2}X + Y$; $b = 2^{n/2}W + Z$. Compute $XW, YZ, (X + Y)(W + Z)$. $T(n) = 3(n/2) + \Theta(n) = \Theta(n^{1.58})$

Polynomial Multiplication

Evaluate, multiply, interpolate. Pick N to be a sufficiently large power of 2.
DFT, where y_k is value of A at $x_k = \omega^k$:
$y_k = \sum_{j=0}^{N-1} a_j e^{2\pi ikj/N}$
$A_{even}(x) = a_0 + a_2 x + ... + a_{n-1} x^{(n-1)/2}$
$A_{odd}(x) = a_1 + a_3 x + ... + a_{n-2} x^{(n-3)/2}$
Evaluate A_{even} and A_{odd} on the squares of the N-th roots of unity, then evaluate A on the N-th roots of unity. Do the same for B. Point-wise multiply to obtain a point-value representation for C. Perform interpolation.
$c_k = \frac{1}{N} \sum_{t=0}^{N-1} y_t e^{-2\pi itk/N}$
$T(N) = 2T(N/2) + O(N) = \Theta(n \log n)$

Minkowsky Sum

Let X and Y be length n sets of integers in $[0, m-1]$. Construct $P_X(x)$ with x^k if $k \in X$ and $P_Y(x)$ with x^k if $k \in Y$. Multiply to get $P_{X+Y}(x) = P_X(x)P_Y(x)$. Return terms with nonzero coefficients. $O(n \log n)$ using FFT.

Union-Find

$\text{MakeSet}(x)$: create set with x; $O(1)$
$\text{FindSet}(x)$: returns $rep[S(x)]$; $O(1)$
$\text{Union}(x,y)$: $S(x), S(y) \rightarrow S(x) \cup S(y)$; $O(n)$, $O(m + n \log n)$, amort. $O(\log n)$
Forest of trees: $O(\log n)$ or use path compression to flatten tree. Combine rank & path compression has total $O(m\alpha(n))$ where $\alpha(n)$ is inverse Ackermann func.

Mincut

Contract edges; merge into supernode. A cut in G/e correspds to a cut in G; contraction cannot decrease size of mincut; size increases iff contract edge in mincut.
GuessCut: $O(n^2)$, pick edge at random and contract until 2 supernodes left
$P(e_1 \in C) = |C|/|E|$ and $|C| \leq 2|E|/n$ so $P(e_1 \notin C) = 1 - 2/n$. $P(findC) =\geq 2/n^2$ so repeating $k = O(n^2 \log n)$ times gives $P(failure) \leq 1/n^c$. GuessCut with $P(success) \geq 1 - 1/n^c$ is $O(n^4 \log n)$.
Graphs can have at most $n^2/2$ cuts w/ prob. $2/n^2$ each

```
FASTCUT($G$)
1  Do 2 repetitions of
2      do random contractions until $n/\sqrt{2}+1$ nodes left
3      recursively call FASTCUT
4  Return better of 2 cuts
```

FASTCUT: $T(n) = 2T(\frac{n}{\sqrt{2}}+1) + O(n^2) = O(n^2 \log n)$. $P(success) = \Omega(1/\log n)$. Amplification: $c(\log^2 n)$ reruns; overall is $O(n^2 \log^3 n)$ (Karger's is $O(m \log^3 n)$).

Weighted Mincut

For e w/ weight m, make m copies. If $w(e) \leq T$, $O(n^2 \log^3 n + mT)$
Faster: compute total weight, pick by sampling: $O(n^2 \log^3 n(\log T + \log n)a)$

Quicksort

In-place comparison sort. Basic is $O(n^2)$, expected $O(n \log n)$. Randomized expected $O(n \log n)$, deterministic is $O(n \log n)$ worst case. Pick pivot element, partition array, recurse.
Paranoid: repeat pivot-choosing & partition until partition is "good". $E[T(n)] = E[T(n/10)] + E[T(9n/10)] + O(n) = O(n \log n)$

Matrix Multiplication Checking

Matrix multiplication is $O(n^{2.37})$. Frievald's algorithm is $O(n^2)$. If $M \neq M_1M_2$, then $P(correct) \geq 1/2$. For error probability $1/n^c$, $O(n^2 \log n)$.

```
FRIEVALD'S($M_1, M_2, M$)
1  Pick random vector $v \in 0, 1^n$
2  Return $Mv == M_1M_2v$
```

Prim's Algorithm

Start with $|S| = 1$ and grow from there. Maintain priority queue Q on $V - S$ where $v.key = \min\{w(u, v)|u \in S\}$. Initially $Q = V$. Pick an arbitrary start vertex. Add the minimum-weight edge connecting to a vertex in $V - S$; repeat until $S = V$. Time is $O(|E| + |V| \log |V|)$ using Fibonacci heap

Kruskal's Algorithm

Maintain connected-so-far MST in Union-Find structure.

```
KRUSKAL'S($G$)
1  $T = \emptyset$
2  for $v \in V$: MAKESET($v$)
3  Sort $E$ by $w(e)$
4  for $e = (u, v) \in E$:
5      if FINDSET($u$)$\neq$FINDSET($v$):
6          Add $e$ to MST
7          UNION($u, v$)
```

$T_{sort}(E) + \Theta(|V|)T_{makeset} + \Theta(|E|)(T_{find} + T_{union}) = O(m \log m + n + m\alpha(n)) = O(m \log m) = O(m \log n)$
Proof: Let P ={picked edges}. Greedy choice: e belongs to sme MST of G/P. Optimal substrcture: $e \cup P$ belong to some MST of G

Maxflow

Directed graph with source and sink. Flow is a function f satisfying:
$f(u, v) \leq c(u, v)$ and inflow = outflow.

Ford-Fulkerson

```
FORD-FULKERSON($G$)
1  $f = 0$
2  while there is an augmenting path $p$:
3      send max possible flow through $p$
4  Return $f$
```

Residual graph G_f: shows how much extra flow (u, v) can take. If f' is a flow in G_f, then $f + f'$ is a flow in G. Then, maxflow in G = f + maxflow in G_f.
Augmenting path: $s \to t$ path along which flow can be sent. If augmenting path exists in G_f, then f is not max flow.
Edmonds-Karp: use FF but w/ BFS to choose augmenting path. $O(VE^2)$ while FF is $O(c|V||E|)$.

Maxflow-Mincut Theorem

$|f^*| = c(S^*)$, f^* is maxflow, S^* is mincut. Prove by supposing f is maxflow. Define $S = \{v \in V | v$ reachable from s in $G_f\}$. For all $u \in S$, $v = V/S$, must have $c_f(u, v) = 0$ (since no augmenting paths). For $(u, v) \in E$: $c_f(u, v) = c(u, v) - f(u, v) = 0$. For $(v, u) \in E$: $c_f(u, v) = f(u, v) = 0$. $|f| = f(S) = c(S)$. We know $|f| \leq c(S^*)$ so f is maxflow.

Minimum Vertex Cover

Vertex cover: set of vertices covering all edges. Generally NP-hard but polynomial in bipartite graphs.
Make G' by adding s connected to all $v \in L$ and t connected to all $v \in R$, set capacity of new edges to 1. Every vertex cover Q defines (s, t)-cut of same value. Find mincut (s, t) in G'. $c(s, t)$. $(S \cap R) \cup (T \cap L)$ is optimal vertex cover.

Matching

Set M of edges sharing no endpoints; perfect if $|M| = |V|/2$.
Bipartite matching: add (s, A) and (B, t) with capacity 1 (inside edges cap 1 also). Run FF to find maxflow: $O(|V||E|)$. Or Hopcroft-Karp is $O(\sqrt{|V|}|E|)$, Macha-Sankowski is $O(|V|^{2.37})$.
Konig's Theorem: size of max matching = size of min vertex cover in bipartite graphs

Linear Programming

Minimize/maximize linear objective function subject to linear inequalities & equations. Constraints form polytope; if bounded, optimal soln is vertex of polytope. Commonly transformed from min to max by negating, all nums to positive by adding x^+ and x^- variables.
Standard LP form has nonnegative vars, max, and $\leq$ constraints.
Strong duality:

$$\begin{array}{cc} \max\ \mathbf{c} \cdot \mathbf{x} & \min\ \mathbf{b} \cdot \mathbf{y} \\ A \cdot \mathbf{x} \leq \mathbf{b} & A^T \cdot \mathbf{y} \geq \mathbf{c} \\ \mathbf{x} \geq 0 & \mathbf{y} \geq 0 \end{array}$$

If one is unbounded, other is infeasible.
Weak duality: f is objective for primal min LP, g is objective for dual max LP. $g(\mathbf{y}) \leq f(\mathbf{x})$ if $\mathbf{x}$ and $\mathbf{y}$ are feasible solns.
Taking dual: rewrite objective as min if needed. Rewrite each inequality as $\leq$, rearrange each so that right-hand side is 0. Define a nonnegative dual variable for each constraint and an unrestricted dual variable for each equality constraint. For each constraint, remove it and add the term (dual variable)*(left-hand side) to the objective. Minimize the result over the dual vars. Rewrite objective to factor out primal vars. Remove each term of te form (primal var)*(exp w/ dual vars) and replace it with a constraint (exp $\geq$ 0 if primal is nonnegative and exp $\leq$ 0 if primal is nonpositive).

Solving LPs

Simplex: $\mathbf{x}$ walks from vertex to vertex in direction of $\mathbf{c}$; worst-case exponential
Ellipsoid: guarantees OPT$\in$ ellipsoid
Interior-point: $\mathbf{x}$ moves inside polytope vaguely in $\mathbf{c}$, runtime $O(m^{3.5}L^2)$. m is # variables & L is # bits required to describe.
ILP is NP-complete.

Game Theory

Thought exps to help predict rational behavior in conflict (players max own utility and actions affect others' utility)
Nash equilibrium: no player can improve her exepcted payoff by unilaterally changing strategy. Exists in every finite game (prove for 2-player zero sum using LPs). Can be computed in polynomial time in 2-player LPs

Minimax Theorem

Finite zero-sum 2-player game. A has m strategies, B has n. A's payoff matrix is $A[i, j]$, B's is negative of A's. With A playing fixed strategy $[p_i]$, B chooses $[q_j^\pi] = \arg\min_{[q_j]} \sum_{i,j} p_i q_j A[i, j]$. A's optimal payout is $v_A^* = \max_{[p_i]} \min_{[q_j]} \sum_{i,j} p_i q_j A[i, j]$; B's is $v_B^* = -\min_{[q_j]} \max_{[p_i]} \sum_{i,j} p_i q_j A[i, j]$. These two payouts are opposite $\to$ existence of Nash equilibrium.

Dynamic Programming

Clever brute-force reusing subproblem solns. Total # subproblems, # guesses per subproblem are all polynomial. Subproblem dependencies need to be acyclic. Two approaches: memoizing (recursive alg & lookup table), iterative (understand dependencies). **E.g.** max contiguous subsequence, making change, longest increasing subsequence, box stacking, noncrossing bridges, knapsack, balanced partition, edit distance, counting boolean parenthesizations, alternating coins

Single Pair Shortest Paths

Unweighted - BFS, $O(V+E)$
Nonneg. edge weights - Dijkstra, $O(E+V\log V)$
General - Bellman-Ford, $O(VE)$
Acyclic DAG - topological sort + 1 iter BF, $O(V+E)$
Dijkstra's: use priority queue to select closest unvisited vertex, keep unvisited set, use triangle inequality to relax edges.
Bellman-Ford: like Dijkstra's but relax all edges $|V|-1$ times.

All Pairs Shortest Paths

Unweighted - $V\times$BFS, $O(VE)$
Nonneg. edge weights - $V\times$Dijkstra, $O(VE+V^2\log V)$
General - $V\times$B-F, $O(V^2E)$
General - see below, $O(VE+V^2\log V)$
Dynamic programming: $d_{uv}^{(m)}$ is weight of shortest $u\to v$ path using $\leq m$ edges. Time is $O(V^4)$; bottom-up using relaxation
min,+ matrix multiplication: define $D_{ij}^{(m)}$ as shortest i,j path using $\leq m$ edges and $W_{ij}=w(i,j)$. $D^{(m)}=W^m$ where multiplication is replaced with addition and addition is replaced with min. Can't use Strassen etc. so $O(V^3\log V)$.
Floyd-Warshall: faster DP, define subproblem $c_{uv}^{(k)}$ as weight of shortest path using only first k vertices. $O(V^3)$
Johnson's: find shift of edge weights to make all nonnegative then run Dijkstra's. Define cost function h solving system of difference constraints s.t. $w_h(u,v)=w(u,v)-h(u)+h(v)\geq 0$, then run Dijkstra's on G_h. $O(VE+V^2\log V)$. Note that BF can solve any system of difference constraints in $O(VE)$ where V = variables, E = constraints.

Decision, Search, Optimization

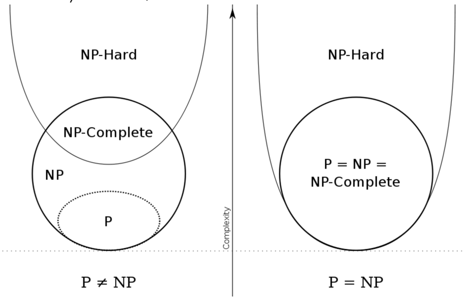
Optimization: find optimal soln. **Search**: find soln better than k. **Decision**: does soln better than k exist? Solve optimization in poly time → search and decision in poly time, etc.

Easier, Harder

Shortest path, longest path. MST, Hamiltonian cycle. Linear programming, integer programming. Bipartite matching, tripartite matching.

Complexity

P = problems solvable in polynomial time. $\Pi\in$ P if there is a poly-time A s.t. $\forall x:(\Pi(x)$ is "yes") $\Leftrightarrow$ ($A(x)$ outputs "yes")
NP = problems whose solutions can be verified in polynomial time. $\Pi\in$ NP if there is a poly-time verification V_Π & $c>0$ s.t. $\forall x:(\Pi(x)$ is "yes") $\Leftrightarrow$ ($\exists$ certificate y s.t. $|y|\leq|x|^c$ & $V_\Pi(x,y)$ is "yes")
NP-hard = as hard as every problem in NP. $\Pi\in$ NP-hard iff $\forall\Pi'\in$ NP, $\Pi'\leq_p\Pi$
NP-complete = in NP & NP-hard
If $P\neq NP$, then NP-hard $\Rightarrow$ not in P



NP-Complete Problems

Clique: given $G=(V,E)$ and integer k, does G have a complete subgraph of size $\geq k$?
Independent set: given $G=(V,E)$ and integer k, does G have a set of vertices of size $\geq k$ s.t. no two vertices are adjacent?
Vertex cover: given $G=(V,E)$ and integer k, does G have a vertex cover of size $\leq k$?
3-coloring: given $G=(V,E)$ and 3 colors, can be assign a color to every vertex s.t. adjacent vertices have different colors?
Hamiltonian cycle: given $G=(V,E)$, does it have a cycle passing through each vertex once?
3SAT: Given $X=\{x_1,x_2,...\}$ binary variables and set of clauses of 3 variables, is there an assignment of values s.t. there is at least one true variable in every clause?
Subset sum: given a set S of integers, is there a subset $T\subseteq S$ s.t. $\sum_{x\in T}x=0$?
Partiton: given a set S of integers, can it be partitioned into S_1 and S_2 s.t. $\sum_{x\in S_1}x=\sum_{x\in S_2}x$?

Reductions

A poly-time reduction from Π_1 to Π_2 is a poly-time alg R s.t. if x is input to Π_1, $R(x)$ is input to Π_2 and $\Pi_1(x)=$"yes" $\Leftrightarrow$ $\Pi_2(R(x))=$"yes". Iff there is a poly-time reduction from Π_1 to Π_2 we know $\Pi_1\leq_p\Pi_2$. Reduce X to Y = if you can solve Y, you can solve X given a black box solver for Y
3SAT → Vertex Cover → Hamiltonian Path → Travelling Salesman. 3SAT → 3D Matching → Subset Sum → Partition → Rectangle Packing.
Cook's theorem: circuit-SAT is NP-complete. Karp reduced circuit-SAT to many other problems.

Approximation Algorithms

α-approximation alg: computes output APX whose cost is within $\alpha\geq 1$ factor of OPT, or $\frac{1}{\alpha}c(OPT)\leq c(APX)\leq\alpha c(OPT)$
Approximation ratio: $p(n)\geq\max(\frac{C}{C^*},\frac{C^*}{C})$

Online Algorithms

Competitive analysis: α-competitive if for all input sequences σ: $c_A(\sigma)\leq\alpha c_{OPT}(\sigma)$ or $\alpha=\max_\sigma c_A(\sigma)/c_{OPT}(\sigma)$
Ex: ski rental (better late than never), linked list updates (move-to-front), load balancing, robot navigation, secretary problem

Streaming Algorithms

Have limited memory or can make limited passes over input, output result at end. Alg A computes function f in the streaming model if there exists constant c and $0\leq\delta<1$ s.t. for any input x, $P(\frac{f(x)}{c}\leq A(x)\leq cf(x))\geq 1-\delta$
Probably approximately correct: compute an (ϵ,δ)-approximation to F_0, i.e. a number $\hat{F_0}$ s.t. w/ prob. $\geq 1-\delta$, $(1-\epsilon)F_0\leq\hat{F_0}\leq(1+\epsilon)F_0$
Counting distinct elements: for input elements x_i find $z=\max zeros(h(x_i))$, output 2^z. Prove with $W_{z,i}$ as indicator var for event that $zeros(h(x_i))\geq z$, $I_z=\sum W_{z,i}$. $E[W_{z,i}]=1/2^z$, $E[I_z]=d/2^z$, $\mathrm{Var}[I_z]\leq d/2^z$, $P(I_z=0)\leq 2^z/d$. $P(2^{z_2}\leq\hat{d}\leq 2^{z_1})\geq 1/2$. Amplify by repeating k times to make prob of failure $2^{-\Omega(k)}$.

Markov Chains

If G is strongly connected and has self-loops on every vertex, G has a unique stationary distribution.

Sublinear-time Algorithms

Take $o(n)$ time. **Ex:** diameter of point set, sampling of connected components (estimate n_v by visiting up to $2/\epsilon$ nodes), min weight spanning tree.
Connected components: we know $|\tilde{C}-C|\leq\epsilon n/2$ and that $P(|\hat{C}-\tilde{C}|\geq\epsilon n/2)\leq 2\exp(-2k\epsilon^2/4)$ so choose $k\geq 2\ln(2/\delta)/\epsilon^2)$ for runtime of $O(k/\epsilon^2)$
Average degree: sample $8/\epsilon$ sets of vertices, return minimum average degree of sets.

Linear Program Examples

Shortest path: $\min w_{ij}x_{ij}$ s.t.
$\sum_j x_{ij}-\sum_j x_{ji}=1$ for $i=s$
$\sum_j x_{ij}-\sum_j x_{ji}=-1$ for $i=t$
$\sum_j x_{ij}-\sum_j x_{ji}=0$ for $i\neq s,t$
$\forall(i,j)\in E:x_ij\in\{0,1\}$
Minimum vertex cover: $\min\sum_i x_i$ s.t.
$\forall(i,j)\in E:x_i+x_j\geq 1$
$\forall i:x_i\in\{0,1\}$

LP Dual Example

$$\begin{array}{rl} \max\ x+2y & \min\ 4a+5b+8c \\ x+y \le 8 & a+c \ge 2 \\ x \le 5 & b+c \ge 1 \\ y \le 4 & a,b,c \ge 0 \\ x,y \ge 0 & \end{array}$$

Dynamic Programming Examples

Longest increasing subsequence: $L[i] = \max_{j|j<i,A[j]<A[i]} L[j] + 1 \rightarrow O(n^2)$. $M^{(i)}[l] = k$, index of smallest value $A[k]$ s.t. $k < i$ and $\exists$ increasing subseq. of length l ending at $A[k]$; return index of largest nonempty entry of $M^{(n+1)}$. $M^{(i+1)}$ differs from $M^{(i)}$ on at most one index, where $M^{(i+1)}[l] = i \rightarrow O(n \log n)$.

Alternating coin game: n coins, $v(i,j) =$ value player 1 can guarantee if remaining coins are $v_i, ..., v_j$. If $j-i$ even, player 2 plays. If $j-i$ odd: $v(i,j) = \max(v_i + v(i+1,j), v_j + v(i,j-1))$. If $j-i$ even: $v(i,j) = \min(v(i+1,j), v(i,j-1))$. $O(n^2)$.

Reduction Examples

SAT to 3SAT: pad short clauses so they have 3 literals; break long clauses into shorter clauses, e.g. $(l_1 \vee ... \vee l_n)$ into $(l_1 \vee l_2 \vee x_2) \wedge (\neg x_2 \vee l_3 \vee x_3) \wedge ... \wedge (\neg x_{n-2} \vee l_{n-1} \vee l_n)$

3SAT to ILP: for every literal x_i in SAT, we define y_i in the ILP s.t. $y_i = 1$ iff $x_i = T$ and $y_i = 0$ iff $x_i = F$. Add constraint $0 \le y_i \le 1$ so $y_i \in \{0,1\}$; note that $\neg x_i \Rightarrow 1 - y_i$. Transform clauses in SAT to inequalities > 0.

3SAT to Vertex Cover: create connected pair of p_{x_i}, n_{x_i} for each variable, connected 3-node cluster for each clause. Connect nodes in each clause cluster to appropriate variable nodes. Pick assignment for each variable and remaining nodes in each clause cluster.

SAT to Clique: create 3-node cluster for each clause. Add edge between all pairs of nodes in different clusters except for $(x_i, \neg x_i)$.

Clique to Independent Set: invert G.

3SAT to Hamiltonian Cycle: n paths corresponding to x_i, each path is $2k$ nodes (k clauses), edges in both directions in path, add edges connecting endpoints of paths, add s and t nodes where s is connected to x_1 path and t connected to x_n path plus backpath $t \rightarrow s$. Add nodes corresponding to clauses and connect clauses in direction depending on whether x_i is inverted, skipping gap between clauses in paths.

Approximation Examples

Vertex cover: pick any $(u,v) \in E$, add u, v to S, delete u, v, and incident edges from G, repeat until all edges gone $\rightarrow$ 2-approximation. (Greedy vertex cover is $O(\log n)$-approximation, pick max degree)

Set cover: pick set covering most points (greedy) $\rightarrow$ $(\log n + 1)$-approximation

TSP: reduce Hamiltonian cycle to α-approximation of TSP. Create G' with same vertices, edge for every pair of vertices, $c(u,v) = 1$ if $\{u,v\} \in E$ and $c(u,v) = \alpha$ otherwise. Metric TSP with preorder walk of MST is 2-approximation.

Pset 1-1: Solving Recurrences

A: $T(n) = 8T(n/3) + \sqrt{n} \log n = \Theta(n^{\log_3 8})$. Master theorem case 3.

B: $T(n) = T(n-1) + 2T(n-2) + 1 = \Theta(2^n + (-1)^n)$. Quad. roots are -1 and 2.

C: $T(n) = T(n/2) + 2T(n/4) + n = \Theta(n \log n)$. Recursion tree.

D: $T(n) = 6T(\sqrt[3]{n}) + \log^2 n$. Let $n = 2^m$, set $S(m) = T(2^m)$, $S(m) = \Theta(m^2) \rightarrow T(n) = \Theta(\log^2 n)$.

Pset 1-2: Drawing Trees on the Plane

Bijective map from points on plane to balanced tree. For each node in tree, compute subtree size in $O(n)$. Sort pts by increasing x-coord. Pick leftmost pt as root, sweep through point angles, partition by subtree size. $O(n \log^2 n)$.

Pset 1-3: Extensions of FFT

FFT on polynomials w/ 2 vars in $O(n^2 \log n)$. Define matrix $M_{ab} = \omega^{ab}$. $p_{ab} = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} p(\omega^i, \omega^j) \omega^{-(ai+bj)}$.

Pset 1-4: Firing Neurons

CHARGE operation with neurons. CHARGE$_3$: $\Phi = 2\sum_i x_i$, $\Delta\Phi = 1$ if no fire and $-2(k_l + k_r)$ if fire, time $T = 1$ if no fire and $2(k_l + k_r) + c_2$ if fire; amortized $O(1)$. CHARGE$_2$: accounting method with coins; amortized cost $\Theta(n^2)$.

Pset 2-1: Reviewing Mincut

r-way cut set. Stop when $2(r-1)+1$ vertices left, similar to analysis of repeated GUESS-CUT, $O(\frac{n^{2r}}{(2r-2)!} \log n)$. Construct graph with constant prob. of picking edge in mincut to antagonize VERTEX-GUESSCUT.

Pset 2-2: Working w/ Broken Scale

Broken scale, make each comparison $O(\log n)$ times. $P(\text{weighing fails}) = P(t \ge k/2) \le P(t \ge E[t] + k/6) \le \exp(-k/18)$; pick $k = 36 \ln n$ so error becomes $1/n^2$. Union bound: total prob. of any wrong comparison is $O(\log n/n)$.

Pset 2-3: Social Investigator

Check if friend sets are duplicates. Iterate through matrix rows and hash in $O(n^2)$, switch diagonal bits to 1 in part B (where people can be friends with each other to be doubles).

Pset 2-4: Conquer the Summit

Local search walk $\not\rightarrow$ global maximum. Divide & conquer into quadrants for $O(n)$. 4D search $\rightarrow$ cube into 8 sections. Randomization: $O(n^2) \rightarrow O(n^{1.5})$

Pset 2-5: Younger Brother Problem

Almost sorted & not sorted test. Construct antagonistic input w/ completely sorted list then similar but not sorted list. Binary search: $\text{FOO}(x, 0, n-1) < \text{FOO}(y, 0, n-1) \Rightarrow x < y$, true b/c x cannot branch right while y branches left. Prove that if array is Not Sorted, $\ge n/4$ elements fail binary search test.

Pset 3-1: Second Minimum Spanning Tree

A: prove 2nd-MST of G differs from MST at exactly two edges. Take edge in MST but not in SMST; prove replacement edge has $> w(e)$.

B: find SMST in $O(m \log mn^2)$: find MST T in $O(m \log m)$, for every $(u,v) \in E$ find max weight edge $R(e)$, replace edge that minimizes $w(e) - w(R(e))$

C: Build component tree using coloring

D: Find SMST using c), $O(m \log m + n \log^2 n)$

Pset 3-2: Gradient Descent

A: Prove $g_k \le \frac{1}{\sqrt{k+1}}[2L(f(x^0) - f(x^*))]^{1/2}$

$$\frac{1}{2L}||\nabla f(x_k)||^2 \le f(x_k) - f(x_{k+1})$$

$$\frac{1}{2L}\sum_{k=0}^{n}||\nabla f(x_k)||^2 \le f(x_0) - f(x_{n+1})$$

$$\frac{k+1}{2L}g_k^2 \le \frac{1}{2L}\sum_{k=0}^{K}||\nabla f(x_k)||^2$$

$$\frac{k+1}{2L}g_k^2 \le f(x_0) - f(x_{K+1})$$

$$\frac{k+1}{2L}g_k^2 \le f(x_0) - f(x^*)$$

so grad. descent steps $\le \frac{2L}{\epsilon^2}(f(x_0) - f(x^*))$

B: strictly convex func. that never converges: $f(x) = x^2$, $x_0 = 1$, step size 1

C: Hessian: square matrix of second-order partial derivative

Pset 3-3: Musical Office Hours

Graph G where V is set of classrooms, E is set of hallways. Build a flow network: for each $v \in V$, create v_i and v_f and add edge $(v_i, v)f)$ w/ $c = \infty$; for each $(u,v) \in E$, add (u_i, v_f) w/ $c = \infty$; add source s and edges to v_i w/ $c = f(v_i)$, add sink t and edges from v_f w/ $c = f'(v_f)$. Edmonds-Karp $\rightarrow O(VE^2)$

Pset 3-4: Greedy Eater

```
GreedyEater(C[])
1   days = 0
2   while True:
3       for each of the $k$ largest counts in $C$:
4           if count is 0: return days
5           Decrement the count
6       days++
```

Store repeated counts as tuple (value, multiplicity) and use linked list $\to O(n)$. If $m << n$, binary search for max # of days

Pset 4-1: Efficient Max Flow

Run FF in phases on parameter D, which starts at c_{max}. $G_f(\delta)$ is subgraph of G_f w/ edges w/ cap $\geq \delta$.
A: Show maxflow from $s \to t$ in each phase $\leq 2|E|D$. $c_{max} \leq 2D$, any cut has $\leq |E|$ edges, size of any cut $\leq 2|E|D$.
B: Prove # augmentations for each phase is $\leq 2|E|$. Every edge has cap $\geq D$, so each augmentation sends $\geq D$ units of flow. $2|E|D/D = 2|E|$ augmentations.
C: Runtime: # phases is $O(\log(c_{max}))$. Each phase is $O(|E|^2)$: $2|E|$ augmentations + BFS for each. Overall is $O(|E|^2 \log(c_{max})) \to$ poly.

Pset 4-2: Round Table Matching

A: Alg to find arrangement. Lemma $H = (V, E)$ is 2-regular graph $\to$ decomposes to disjoint cycles. Create new graph w/ cap 1 b/t sponsors & businessmen, cap 2 b/t s & sponsors and businessmen & t. Maxflow, $O(VE)$
B: Prove union of 2 disjoint perfect matchings forms covering subgraph w/ disjoint cycles. Every vertex has degree 2; 2-regular graph consists of disjoint cycles; every vertex included so covering.
C: Find faster alg in $O(\sqrt{V}E)$. Find 2 different perfect matchings by removing first matching from G, run Hopcroft-Karp. Use b)

Pset 4-3: Linear Programming

A: Solve LP by checking all 2 corners of polytope & find objective values. Or, find coefficients a, b s.t. a(constraint$_1$) + b(constraint$_2$) = $c_1x_1 + c_2x_2$ in objective. **B**: Convert into standard form.

$$\min_x \sum_i |c_i x_i| \qquad\qquad \min_t \sum_i t_i$$
$$Ax \leq b \qquad\qquad Ax \leq b$$
$$\forall i : c_i x_i \leq t_i$$
$$\forall i : c_i x_i \geq -t_i$$

$$\min_x \frac{c^T x + d}{e^T x + f} \qquad\qquad \min c^T y + dz$$
$$Ax \leq b \qquad\qquad Gy = hz$$
$$Gx = h \qquad\qquad e^T y + fz = 1$$
$$Ay \leq bz$$
$$z \geq 0$$

D: Use oracle to binary search for objective
E: Strict inequalities $\to$ maybe no soln (e.g. $\max x$, $x < 5$). Replace max with sup.

Pset 5-1: Reductions

A1: "Permutation of matrix M s.t. $\prod_{i=1}^n M_{i\sigma(i)}$ is not 0?" to "Bipartite graph has perfect matching?". Create vertex for each row, column; add edge if entry is nonzero.
A2: "Does G have clique of size $\geq k$?" to "Does G have a clique of size $\geq |V|/2$?". If $k = |V|/2$, run on G. If $k > |V|/2$, add $2k - |V|$ dummy nodes. If $k < |V|/2$, add $|V| - 2k$ fully connected nodes.
B: Solve LP in 1 oracle query. Find dual of LP, combine constraints from both together, satisfied at exactly 1 pt (optimal soln to both LPs).

$$\min 1$$
$$\sum_j c_j x_j = \sum_i b_i y_i$$
$$\sum_j A_{ij} x_j \leq b_i, \forall i$$
$$\sum_i A_{ij} y_i \geq c_j, \forall j$$
$$x_j, y_i \geq 0, \forall i, j$$

Pset 5-2: Designing Games

Each player has n strategies; k pairs of strict Nash equilibria: $\{(i_t, j_t\}_{t \in [1,k]}$. Check if there are t_1, t_2 s.t. $i_{t_1} = i_{t_2}$ and $j_{t_1} \neq j_{t_2}$ or $j_{t_1} = j_{t_2}$ and $i_{t_1} \neq i_{t_2}$ (if there are, return none). Set $(a_{i,j}, b_{i,j}) = (1,1)$ if $(a_{i,j}, b_{i,j}) = (a_{i_t,j_t}, b_{i_t,j_t})$ and $= (0,0)$ otherwise; return payoff matrix.

Pset 5-3: Roads Taken Too Many Times

A: One-way roads $s \to t$: $\binom{m}{n-1}$ ways.
B: $S[i,j]$ is # ways to arrive at i in j days. $S[1,j] = S[1,j-1] + S[2,j-1]$, $S[n,j] = S[n,j-1] + S[n-1,j-1]$, or $S[i,j] = S[i-1,j-1] + S[i,j-1] + S[i+1,j-1]$.
C: Travel up to k in a day; modify b) so that each $S[i,j]$ sums predecessor values.
D: Travel by bus to cities; modify b) to use adjacency matrix to compute $S[i,j]$.
E: $O(n^3 \log m)$: M is adjacency matrix, compute M^m using $O(\log(m))$ squarings.

Pset 5-4: Octopus's Garden

A: DP: $T_{i,j}$ contains the width and height of the maximum contiguous subarray without coral whose bottom-right corner is (i,j). **B**: Reduce CLIQUE to CORAL. Construct G with adjacency matrix M and $|V| - k$ indices to remove coral. If G has clique of size k, removing non-clique vertices yields submatrix of M with no coral in any cell. If there are at most $|V| - k$ indices, removing vertices corresponding to indices will leave clique of size k.

Pset 5-5: Free Food Calendar

A: In $S_i \subseteq [1,n]$ days, free food $\geq c_i$ times; in $S'_j \subseteq [1,n]$ days, free food $\leq c'_j$ times.

$$\max \sum_j x_j$$
$$\sum_{j \in S_{1,i}} x_j \geq c_{1,i}, \forall i \in [1, k_1]$$
$$\sum_{j \in S_{2,i}} x_j \leq c_{2,i}, \forall i \in [1, k_2]$$
$$x_j \in \{0,1\}, \forall j$$

B: Reduce SAT to FOOD: every clause of SAT is constraint with $c_{1,i} = 1$, also set $x_i + x_j = 1$ for each variable in SAT. **C**: $\min(\max_k T_k - \min_k T_k)$ s.t. $\forall (i,j) : T_i - T_j \leq r_{i,j}$, run Bellman-Ford in $O(n(m+n))$.

Pset 6-1: Vacation Planning

A: Visit every island using $c-1$ ferries between connected components.
B: Kruskal streaming alg w/ union-find structure to track connected components in $O(n \log n)$ space (store $n-1$ edges).
C: If mincut has size $\geq k$, then b/t any two vertices there are at least k edge disjoint paths; prove using flow network w/ all edges at capacity 1.
D: k bridges closed; decide whether graph is still strongly connected in $O(kn \log n)$ space. Prove that mincut is $> k$. Track k possible subgraphs; fill up by adding edges sequentially when u, v not connected; check if mincut of union of graph is $> k$.

Pset 6-2: Waiting for the Bus

A: T min to walk and t min by bus. Wait until $t_0 = T - t$; $\beta = t/T$. Competitive ratio is $2 - \beta$ & is optimal. If bus arrives before t_0, both opt and online have same time. If bus arrives after, competitive ratio is $2 - \beta$. Prove optimality with antagonistic input s.t. bus arrives right after t'_0 for both $t'_0 > t_0$ and $t'_0 < t_0$.
B: If $t_1 < \alpha t_0$, $CR = 1$. If $\alpha t_0 \leq t_1 < t_0$, $E[CR]$ max at $t_1 = \alpha t_0$ and $E[CR] = \frac{\alpha(1-\beta) + \frac{1}{2} + \frac{\beta}{2}}{\alpha(1-\beta) + \beta}$. If $t_1 \geq t_0$, $E[CR] = \frac{1}{2}\alpha(1-\beta) + \frac{1}{2} + \frac{1}{2}(2-\beta)$. **C**: Find α for strategy with best CR. Case 2 decreasing while case 3 increasing, so set their CR's equal.

Pset 6-3: Can we Approximate?

A: 2-approximation: prove $\sum_{i=1}^k p_i \leq p^* < \sum_{i=1}^{k+1} p_i$. Show that $p^* < \sum_{i=1}^{k+1} p_i = p_{k+1} + \sum i = 1^k p_i$, $\to$ output is greater than $p^*/2$.
B: Special case where $\forall i : w_i = p_i$; problem equivalent to minimizing remaining weight in knapsack.
C: SUBSET SUM $\to$ POSITIVE SUBSET SUM $\to$ this problem. Add $X = 2\sum_{i=1}^n |x_i|$

Pset 6-4: Manhattan Traffic

A: define ILP:

$$\min z$$
$$\sum_{i\in P(e)} x_i + \sum_{i\in N(e)} (1-x_i) \le z$$
$$x_i \in \{0,1\}$$

B: Chernoff bound show $P(\sum_{i=1}^{k} X_i \ge \alpha t) \le \exp(-\alpha t/4)$

C: Prob $\ge 2/3$ approx. factor is $O(\log n)$. Let $t = \max_e z_e^*$ and $\alpha = O(\log n)$, use union bound.

True/False

- Floyd-Warshall cannot be improved with Strassen's b/c no subtraction
- If $E = O(V^{1.5})$, Johnson's faster than F-W
- Subproblem graph of DP is DAG, not directed tree
- Every lowest weight edge always in some MST
- n vertices & edges $\to$ find MST in $O(n)$
- FF is $O((|V|+|E|)|f|)$
- Need to increase capacity of all mincuts to increase maxflow
- LPs can have many or no optimal solns
- If $P = NP$, then 3SAT solvable in poly time
- Paranoid quicksort only requires constant fraction to get $\theta(n\log n)$ bound
- Binary min-heap n elements supports INSERT in $O(\log n)$ and DELETE-MIN in 0 amortized.
- Matrix multiplication shortest paths, replace $(*,+)$ with $(+,*)$ does not compute product of weights of all paths between vertices (since cycles)
- Negating all edge weights in a weighted undirected graph G and then finding MST gives us the maximum-weight spanning tree of the original graph G
- Unique edge weights can produce multiple SMSTs
- Floyd-Warshall recursion contains only first k intermediate nodes, not up to k edges
- Determine if hashes in universal family by checking if prob. of collision $> 1/m$
- If there's poly time reduction from the general vertex cover problem to bipartite matching, there is a poly time algorithm that solves SAT
- A randomized algorithm that is always correct and has expected linear running time can be converted to a randomized algorithm that always runs in linear time, but is correct only with probability 99%
- For every NP-hard minimization problem there is a constant c ¿ 1, such that approximating the problem to within a factor c is NP-hard (false, since $(1+\epsilon)$-approx.)
- Given a maximization linear program LP1 and its dual LP2, the objective value of every feasible solution to LP1 is not larger than the objective value of any feasible solution to LP2
- Transpose for self-organizing lists is not 2-competitive

Review Sheet

Divide & conquer: FFT. Tiling problem (place L-tile at center of subsquare; $T(n) = 4T(n/2) + C = O(n^2)$)
Amortized analysis: binary counter (aggregate method: flip last bit n times, next $n/2$, $n/4$, etc. $\le 2n$ so amortized cost is 2). Union find path compression.
Randomized algorithms: pairwise independence ($P(a\cap B) = P(A)P(B)$), mutual independence is stronger. Coupon collector $E(T) = \frac{1}{p_1} + ... + \frac{1}{p_n} = n(1 + \frac{1}{2} + ... + \frac{1}{n}) = O(n\log n)$
Hashing: universal hashing expected collisions n/m. Matrix method: pick h to be a random $b\times u$ 0/1 matrix ($m = 2^b$ and u is bit lenght of keys). $h(x) = hx$ with vector addition mod 2.
Greedy: cut-and-paste: "The proof is by contradiction. Suppose you came up with an optimal solution to a problem by using suboptimal solutions to subproblems. Then, if you were to replace ("cut") those suboptimal subproblem solutions with optimal subproblem solutions (by "pasting" them in), you would improve your optimal solution. But, since your solution was optimal by assumption, you have a contradiction". Balloon-ray problem: start at direction which does not intersect balloons, then identify first balloon and shoot tangent to it on other side.
Gradient descent: convergence, step size.
Flow/matching: source to all sinks $\to$ graph's mincut. For multiple source/sinks, introduce dummy nodes. Series-parallel graph—find limiting edges in all parallel paths.
Linear programming: duality, game theory with minimax
Dynamic programming: Bellman-Ford, edit distance ($E(i,j)$ is edit distance from first i chars of first string to first j chars of second string; $E(i,j) = \min(1 + E(i-1,j), 1 + E(i,j-1), \text{diff}(i,j) + E(i-1,j-i)))$
Complexity: classes, reductions
Approximation algs: vertex cover, metric TSP (see above)
Online algs: ski rental, cow path (alternate directions for 2^i distance, CR is 9)
Streaming algs: probably approximately correct, majority element (track guess and count, decrement count if does not match guess), bipartite test (check if edges so far create odd cycle, if so, output no)
MCMC: stationary distribution, eigenvalues (eigenvector for eigenvalue 1 is steady state), periodicity (period $k = \gcd(n : P(X_n = i|X_0 = i) > 0)$ needs to be ≥ 2). Every Markov Chain with a finite state space has a unique stationary distribution unless the chain has two or more closed communicating classes.
Sublinear: PAC learning: A class of functions F is Probably Approximately (PAC) Learnable if there is a learning algorithm L that for all f in F, all distributions D on X, all ϵ ($0 < \epsilon < 1$) and δ ($0 < \delta < 1$), will produce an hypothesis h, such that the probability is at most δ that $error(h) > \epsilon$.