Strategies

Divide and Conquer

Solve a subproblems of size $\frac{n}{b}$ and recombine them in time $O(n^d)$

$$T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d) \implies T(n) = \begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

How to Recognize

- Problem is easily broken into substantially smaller subproblems

How to Prove

- Induction

Tips and Tricks

- It may be easier to solve a more general problem (instead of finding median, find kth smallest)

Greedy Algorithms

Build a solution piece by piece by choosing whatever is optimal in the moment

How to Prove

- Swapping Arguments

Dynamic Programming

Identify a collection of subproblems and tackle them one by one, using smaller problems to solve larger ones.

How to Recognize

Input	Subproblem	Function	Number of subproblems
$x_1, x_2, ..., x_n$	$x_1, x_2, ..., x_i$	$F(i)$	$O(n)$
$(x_1, x_2, ..., x_n)$ $(y_1, y_2, ..., y_m)$	$(x_1, x_2, ..., x_i)$ $(y_1, y_2, ..., y_j)$	$F(i, j)$	$O(mn)$
$x_1, x_2, ..., x_n$	$x_i, x_{i+1}, ..., x_j$	$F(i, j)$	$O(n^2)$

How to Prove

1. Define a base case
2. Define Recurrence Relation
3. Prove recurrence accurately solves the problem via induction.

Approximation

Find an algorithm which can approximate a solution to an NP-Complete problem

How to Prove

- Find a Greedy Algorithm which provides a lower bound on the optimal solution
- Find a different structure which can provide a lower bound on optimal (e.g a matching for set cover, a MST for TSP)

P vs NP

Definitions

- $A \to B$ means a subroutine solving B can be used to solve Q
- **Search Problem:** A problem with an algorithm that checks if solution S to instance I is valid in polynomial time.
- **NP:** The class of all search problems
- **P:** The class of all search problems solvable in polynomial time
- **NP Complete:** NP-Hard and all other search problems reduce to it
- **NP Hard:** $\forall B \in NP, B \to A$

Properties

- $A \to B$ means B is at least as hard as A
- If A is NP-Complete, then B is NP-Complete if it is NP-Hard and $A \to B$

NP-Complete Problems

Problem	**Inputs**	**Objective**
Traveling Salesman	- n vertices - $\frac{n(n-1)}{2}$ distances - Budget b	Find a cycle which uses each vertex exactly once with cost $\leq b$
Rudrata/Hamiltonian Cycle	Graph $G = (V, E)$	Find a cycle which visits each vertex exactly once
Balanced Cut	- n vertices - Budget b	Partition vertices into T and S such that $\|S\|, \|T\| > \frac{n}{3}$ with at most b edges connecting S and T
3D Matching	n boys, girls, and pets	Find n disjoint triples which follow edge preferences
Independent Set	- Graph $G = (V, E)$ - Goal g	Find g vertices where no two have an edge between them
Vertex Cover	- Graph $G = (V, E)$ - Budget b	Find b vertices which touch every edge
Set Cover	- Universe E - $S_1, ..., S_m \subseteq E$ - Budget b	Select b subsets whose union is E
Clique	- Graph $G = (V, E)$ - Goal g	Find g vertices with all possible edges between them
Longest Path	- Graph $G = (V, E)$ - Goal g - Vertices s, t	Find a path between s and t of weight $\geq g$
Knapsack	- Weights $w_1, ..., w_n$ - Values $v_1, ..., v_n$ - Capacity W - Goal g	Find a subset of weights with total weight $\leq W$ and total value at least g
Subset Sum	- Integers $s_1, ..., s_n$ - Goal S	Find a subset of integers which add up to S
k-Cluster	Points $x_1, ..., x_n$ distance metric $d(x, y)$ integer k	Find k clusters of points which minimizes the diameter of the clusters $\max_j \max_{x_a, x_b \in C_j} d(x_a, x_b)$

Graphs

Definitions

- **Tree Edge:** DFS traversed edge $\rightarrow$ [u, v, v, u]
- **Forward Edge:** Connects a vertex with a descendent $\rightarrow$ [u, v, v, u]
- **Cross Edge:** Connects unrelated vertices $\rightarrow$ [u, u, v, v]
- **Back Edge:** Connects a vertex to an ancestor $\rightarrow$ [v, u, u, v]

Properties

- In a DAG, each edge leads to a lower post number
- In a Dag, there is always at least one source and one sink
- Two vertices are part of the same strongly connected component $\Leftrightarrow$ There is a cycle in the Graph $\Leftrightarrow$ There is a backedge in DFS
- Every Directed Graph is a DAG of Strongly Connected Components
- The highest post number vertex returns by DFS is in a source SCC
- If c and c' are strongly connected components and there exists an edge fromo c to c', then the highest post in c is higher than the highest post in c'
- $(u, v) \in E \implies post(u) > post(v)$
- When explore terminates, all reachable vertices are visited
- Running Bellman Ford $|V|$ times can detect negative cycles
- Deleting a cycle edge can never disconnect the graph
- **Cut Property:** Let e be the lightest edge across any partition of the graph. Then e belongs to an MST
- **Cycle Property of MSTs:** Let $C \subseteq G$ be a cycle and e be the heaviest edge in c. Then e cannot be part of any MST
- **Dijkstra's Invariant:** Computed distance values are always either correct or overestimated

Algorithms

Algorithm	**Runtime**	**Objective**
Depth First Search	$O(\|V\| + \|E\|)$	What parts of the graph are reachable from a given vertex
Strongly Connected Components:	$O(\|V\| + \|E\|)$	Finds the strongly connected component that each vertex belongs to
Topological Sort:	$O(\|V\| + \|E\|)$	Returns vertices in a DAG in decreasing post order
Breadth First Search:	$O(\|V\| + \|E\|)$	Process vertices according to distance from start vertex
Dijkstra's Algorithm:	$O((\|V\| + \|E\|) \log \|V\|)$	Find shortest paths tree from start vertex. Only works with non-negative edges
Bellman Ford:	$O(\|V\|\|E\|)$	Finds shortest paths from s in a graph with no negative cycles
Kruskals Algorithm	$O(\|E\| \log \|E\|)$	Finds an MST of the graph

Linear Programming

Primal	**Dual**
$\min \vec{c}^T \vec{x}$	$\max \vec{b}^T \vec{y}$
$A\vec{x} \geq \vec{b}$	$A^T \vec{y} \leq \vec{c}$
$\vec{x} \geq 0$	$\vec{y} \geq 0$

Theorems

- **Weak Duality:** Any feasible solution of the primal upper bounds any feasible solution to the dual
- **Strong Duality:** The optimum of the primal is equal to the optimum of the dual

LP Conversions

- **Max to Min:** Multiply coefficients in objective by -1
- **Inequality to Equality:** $\sum_{i=1}^{n} a_i x_i \leq b \Leftrightarrow s + \sum_{i=1}^{n} a_i x_i = b, s \geq 0$
- **Equality to Inequality** $ax = b \Leftrightarrow ax \leq b$ and $ax \geq b$
- **Unrestricted Sign** Create $x^+, x^- \geq 0$ and replace x with $(x^+ - x^-)$

Max Flow

Given a directed graph $G = (V, E)$ with edge capacities c_e, a source vertex s, and a sink vertex t, find the maximum flow from s to t

Definitions

- **Flow:** a numeric assignment to each edge such that $\forall e \in E, 0 \leq f_e \leq c_e$ and $\forall u \neq s, t, \sum f_{uw} = \sum f_{uw}$
- **Flow Size:** Total flow leaving s $\sum f_{su}$
- **ST Cut:** A partition of the vertices with S on one side and T on the other.
- **Residual Graph:** A graph whose edges have capacity

$$\begin{cases} c_{uv} - f_{uv} & \text{if } (u, v) \in E \text{ and } f_{uv} \leq c_{uv} \\ f_{vu} & \text{if } f_{vu} \notin E \text{ and } f_{uv} > 0 \end{cases}$$

- **Cut Capacity:** The total capacity of the edges leaving S and entering T

Properties

- The size of the maximum flow equals the capacity of the smallest $s - t$ cut

---

Multiplicative Weights

At time $t = 1, 2, ...$, choose from n choices to produce vector $x^{(t)} = (x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)})$ such that $x_i^{(t)} \geq 0, \sum_{i=1}^{n} x_i^{(t)} = 1$ After choosing $x^{(t)}$, we see losses $l^{(t)} = (l_1^{(t)}, l_2^{(t)}, ..., l_n^{(t)})$. Maintain weights $w^{(t)} = (w_1^{(t)}, w_2^{(t)}, ..., w_n^{(t)})$ where $w_i^{(t)} = 0$ and $w_i^{(t)} = w_i^{(t)}(1 - \epsilon)^{l_i^{(t)}}$. Choose $x^{(t)}$ by letting $x_i^{(t)} = \frac{w_i^{(t)}}{\sum w_j^{(t)}}$

Definitions

- **Player's Loss at t:** $\sum x_i^{(t)} l_i^{(t)}$
- **Regret after T steps:** $R = \sum x_i^{(t)} l_i^{(t)} - \min_i \sum l_i^{(t)}$

Properties

- $R_T \leq \epsilon T + \frac{\ln n}{\epsilon}$, so if $T > 4 \ln n, \epsilon = \sqrt{\frac{\ln n}{T}}$, then $R_T \leq 2\sqrt{T \ln n}$
- Strategies producing high losses in the pst will have small weight, Strategies with low losses in the passt will have high weights

---

Computation Models

Comparison Model

How many comparisons does an algorithm perform?

Circuit Complexity

n bits of input are fed into a circuit of AND, OR, and NOT gates.

- **Depth:** The longest path from an input to the output
- **Size:** The number of wires in the circuit

Total number of circuits is $s = 2^{O(s \log s)}$

Cell Probe

Memory M using S words of space. Each word is w bits. Count the number of memory reads and writes

Word RAM

Same idea as Cell Probe, but now any machine operation (+, -, etc) also adds to the cost

Branching Program

Model a function $f : 0, 1^n \to Y$. It is a DAG with one source and several sinks. Each sink corresponds to an output in Y. Each non-sink node is labeled $1...n$ and has a 0 edge and a 1 edge, so input $x \in 0, 1^n$ defines a path. L is the length of the program, a.k.a the maximum number of edges from source to sink.

Communication Complexity

The player Alice has $x \in X$ and player Bob has $y \in Y$. They want to compute $f(x, y)$. They communicate with each other, and the complexity is how many bits they send.

Math

Probability

- $\forall \lambda > 0, \mathbb{P}[Z > \lambda] < \frac{E[Z]}{\lambda}$
- $\forall \lambda > 0, \mathbb{P}[|Z - \mathbb{E}[Z]| > \lambda] < \frac{Var[Z]}{\lambda^2}$
- $\mathbb{P}[U_i X_i] \leq \sum \mathbb{P}[X_i]$

Polynomials

$A(x) = \sum_{i=0}^{d} a_i x^i$ and $B(x) = \sum_{i=0}^{d} b_i x^i$, then $C(x) = A(x)B(x) = \sum_{i=0}^{2d} c_i x^i$ such that $c_k = \sum_{i=0}^{k} a_i b_{k-i}$

$$A(x) = A_e(x^2) + xA_o(x^2) \Rightarrow \begin{cases} A(x_i) = A_e(x_i^2) + x^i A_o(x_i^2) \\ A(-x_i) = A_e(x_i^2) - x^i A_o(x_i^2) \end{cases}$$

Let $\omega_n = e^{i\frac{2\pi}{n}}$, then $\omega_n^{\frac{n}{2}+j} = -\omega^j$ and $\omega_n^2 = \omega_{\frac{n}{2}}$

Bounds

$$n! \approx \sqrt{\pi(2n + \frac{1}{3})} * n^n e^- n \implies \log(n!) = \Theta(n \log n)$$

$$1 - x \leq e^{-x}$$

$$(1 - \epsilon)^z \leq 1 - \epsilon z \text{ for } 0 \leq z \leq 1, 0 \leq \epsilon \leq 1$$

$$-z - z^2 \leq \ln 1 - z \leq -z \qquad \forall 0 \leq z \leq \frac{1}{2}$$

$$\sum_{t=1}^{n} \frac{1}{t} \leq \ln n$$