# Greedy Algorithms

**Stay Ahead Argument:** Eg. for interval scheduling, show it stays ahead of any other compatible set with some number of jobs (a) Show the r-th interval in $S$ finishes as soon as possible.

**Exchange Argument:** Prove at each step the subset $S$ "is promising". Use induction on $L(k)$: After the $k$-th step, there exists some optimal sol. OPT s.t. $J(i) \in S$ iff $J(i) \in OPT$

**Interval Scheduling IS:** $OPT$ extends $S_k$ iff $S_k \subseteq OPT \subseteq S_k \cup \{A_{k+1}, \dots, A_n\}$

[IH] For some $k \geq 0$, suppose $\exists$ OPT extending $S_k$

IS: WTP $\exists OPT'$ extending $S_{k+1}$

Consider $S_{k+1}$ in terms of $S_k$:

① $S_{k+1} = S_k$ : Let $OPT' = OPT$ then, $S_{k+1} = S_k \subseteq OPT = OPT'$
Also $OPT' = OPT \subseteq S_k \cup \{A_{k+1}, \dots, A_n\}$
$A_{k+1} \notin OPT$ b/c it overlaps with $A \in S_k \subseteq OPT$ ⇒ contradict with OPT being a sol.

② $S_{k+1} = S_k \cup \{A_{k+1}\}$
Does $A_{k+1} \in OPT$?

2.1 If $A_{k+1} \in OPT$ then $OPT' = OPT$ already extending $S_{k+1}$

2.2 If $A_{k+1} \notin OPT$, then consider $OPT \cup \{A_{k+1}\}$
- $OPT \cup \{A_{k+1}\}$ contains overlapping $A_{k+1}$ and $A_j$ for some $j \in \{k+2, \dots, n\}$
- By sorting $f_{k+1} \leq f_j$ so there is exactly one $A_j$
- Let $OPT' = OPT \cup \{A_{k+1}\} - \{A_j\}$, then $|OPT'| = |OPT|$ and $OPT'$ contains no overlapping activities ⇒ $OPT'$ is optimum and extends $S_{k+1}$

**Coin Change:** Let $C_0, C_1, \dots$ be partial solutions generated by the algo. Say $C_i$ is extending iff it can be extended by the Optimum sol. OPT

[IH] Suppose $C_i$ is promising, with optimum solution OPT

IS: Consider $C_{i+1} = C_i + d[k_{i+1}]$

WTS that $OPT - C_i$ must already contain one coin with value $d[k_{i+1}]$, so that OPT already extends $C_{i+1}$

Assume for a contradiction, that $OPT - C_i$ does NOT contain on coin with value $d[k_{i+1}]$

- $d[k_{i+1}] = 25$
  Since $C_{i+1} = C_i + d[k_{i+1}]$ the amount remain must be $\geq 25$. If $OPT - C_i$ does not contain any coin with value 25, then $A_i$ must be made up of 1, 5, 10. But any combination of these coins that adds up to more than 25 can be replaced by 25 and uses fewer coins (*)
  ...
- $d[k_{i+1}] = 1$ : It is impossible to make up amounts of 1 w/o using any coins

In every case, we get a contradiction

**Dijkstra's:** For $cost(e) \geq 0$, return s-v cost for every $v \in V$

initialize $S = \{s\}$, $d(s) = 0$

while $S$ not empty:

- loop invariant: $d(u)$ is min s-u distance for $u \in S$
- Define $R = \{v \mid v \in V \setminus S$ and exists $(u,v) \in E$ with $u \in S\}$
- Greedily choose $v \in R$ which minimizes
$$\pi(v) = \min_{e=(u,v): u \in S} (d(u) + w_e)$$
- Set $d(v) = \pi(v)$ and add $v$ to $S$

Proof by induction: $k = |S|$ from 1 to $|V|$ using LI

**Cut Set:** A cut is a subset of vertices $S$. The corresponding cut set $D$ is the subset of edges with exactly one endpoint in $S$.

**Properties of MST:**
- Any undirected graph with distinct edge weights has a unique MST
- Max spanning tree can be computed by negating edges
- Adding a constant to each edge does not change MST
- Cheapest edge in cycle is not neccessarily in an MST

# Dynamic Programming

Step 0: Define recursive structure for optimum solution
Step 1: Define table to store optimum
Step 2: Recurrence for the table values
Step 3: Bottom Up Algorithm
Step 4: Use table values to reconstruct solution

**Knapsack:** if $w < w[i]$ then $dp[i,w] = dp[i-1,w]$
otherwise: $dp[i,w] = \min(dp[i-1,w], w[i] + dp[i-1, w - w[i]])$

**Sequence Alignment:** Let $dp[i,j]$ = min cost to align $x[1..i]$, $y[1..j]$
$dp[i,j] = \min(\alpha[x[i], y[j]] + dp[i-1, j-1], \delta + dp[i-1, j], \delta + dp[j-1])$

# DP on Graphs

**APSP:** $OPT(v,w,i)$ = min cost of any v-w path consisting of paths of restricted form $(v, x(j_1), \dots, x(j_\ell), w)$ where each $j_q \in \{1, \dots, i\}$

$$OPT(v,w,i) = \begin{cases} c_{vw} & i = 0 \\ \min\left(OPT(v,w,i-1),\ OPT(v,x(i),i-1) + OPT(x(i),w,i-1)\right) & i > 0 \end{cases}$$

**Largest Indep Set:** $OPT(w)$ = max size of indep set for subtree rooted at $w$

Case 1: $w \in S$. The children of $w$ cannot be in $S$ but grandchild can

Case 2: $w \notin S$. Children of $w$ can be in $S$

$$OPT(w) = \max\left(1 + \sum_{v \in c(c(w))} OPT(v),\ \sum_{v \in c(w)} OPT(v)\right)$$

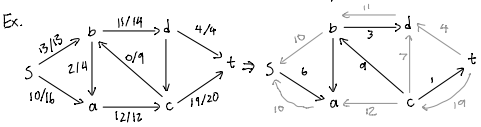
# Network Flow

**Cut:** An s-t cut is a partition $(A, B)$ of $V$
$cap(A,B) = \sum_{e \text{ out of } A} c(e)$

net-flow: $NF(A)$ across an s-t cut $(A,B)$ is sum of flow on edges A to B minus the sum on edges B to A

**Residual Graph:** Given flow graph, the res graph but $E'$ has:
- Forward Edges: for each $e=(u,v)$ on which $f(e) < c(e)$ include same edge: for each $e=(u,v)$ but $c'(e) = c(e) - f(e)$
- Backward: for each $e(u,v)$ on which $f(e) > 0$, include a backedge $e^R(v,u)$ with capacity of $f(e)$

Ex.




**Ford Fulkerson Theorem:** If $f$ is a feasible flow, then:
(i) There $\exists$ cut $(A,B)$ s.t. $v(f) = cap(A,B)$
(ii) The flow $f$ is a max flow
(iii) There is no augmenting path relative to $f$

**Max Flow – Min cut Theorem:** Max Flow < min cut
Proof: Follows from (i) ⇔ (ii) along with weak duality

**Constructing Cuts:**
1) Start $V_s = \{s\}$ $V_t = V - V_s$
2) For any $(u,v) \in E$ with $u \in V_s$, $v \in V_t$
$f(u,v) < c(u,v)$ let $V_t = V_t - \{v\}$ $V_s = V_s \cup \{v\}$
3) For any $(u,v) \in E$ with $u \in V_t$, $v \in V_s$
$f(u,v) > 0$, let $V_t = V_t - \{u\}$, $V_s = V_s \cup \{u\}$

**Properties:**
- There may be cycles containing positive flow
- There always exist a maxflow w/o cycles carry +flow
- Unique mincut does NOT mean unique max-flow
- Distinct capacities does NOT imply unique max flow
- Multiply all edge capacities by $L$ does not change min-cut
- Adding positive number to all capacities does not necessarily change max flow

# Linear Programming

**Initial Feasible Vertex:** Given $(A,b,c)$ consider startup LP maximize $-z_1 - z_2 - \dots - z_m$ (i.e. minimize $z_1 + z_2 + \dots + z_m$) subject to $x, z \geq 0$ and $Ax - z \leq b$. For this, use initial guess $x = 0$, $z = b^-$ where $b_i^- = -b_i$ and 0 otherwise. This startup has solution $(x_0, 0)$ (with $z = 0$ and the objective function equal to 0) iff $x_0$ is feasible solution of the original LP. Simplex will return a feasible vertex $x_0$ on startup as long as feasible set $F$ is non-empty

**Max Flow as LP:** For each edge $(u,v)$, add LP variable $x(u,v)$ to represent flow from $(u,v)$. The first $|E|$ rows of $A$ consists of flow capacity constraint $x(e) \leq cap(e)$. The next $|V|$ rows consists of flow conservation constraints, for $v \in V$:
$\sum(x(e)$ for $e$ entering $v) = \sum(x(e)$ for $e$ leaving $v$.
Objective function: maximize $\sum(x(e)$ for $e$ entering sink)

**Integer LP:** NP-hard. We can use LP relaxation by converting the constraints $x(i) \in \{0, 1\}$ into a pair of linear constraints $0 \leq x(i) \leq 1$. Then rounding is used to obtain some info. The approx. ratio is usually based on the integrality gap.
$$I_G = \frac{M_{int}}{M_{frac}} = \frac{\text{round minimum}}{\text{real minimum}}$$

# Decision Problems

**Indep Set IS(G,k):** Given graph $G$ and int $k$, is there a subset of vertices $S \subseteq V$ s.t. $|S| \geq k$ and for each edge at most one of its endpoints is in $S$

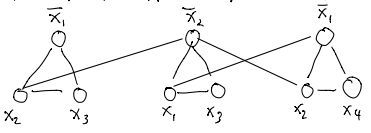
**Vertex Cover:** Given $G$, $k$, is there a subset $S \subseteq V$ st. $|S| \leq k$ and for each edge at least of its endpoints is in $S$

**Set cover:** Given universe $U$ of elem, collection $S = \{s_1, \dots, s_m\}$ of subsets of $U$, and int $k$, does $\exists$ a collection of subsets of $\leq k$ whose union = $U$

**Circuit SAT:** Given combinational circuit built with AND, OR, NOT, can the circuit inputs be set so that output=1

**SET PACK:** Given $U$ and $S = \{s_1, \dots, s_m\}$, $k$. Does $\exists$ a collection $C \subseteq S$ of size $\leq k$ s.t. no two distinct elem $S_i, S_j \in C$ intersect

**Reduction via gadgets:** $3SAT \to_p$ Indep set
Given instance $\Phi$ of 3SAT, construct instance $(G,k)$ of IS where IS size $|S| \geq k = |\Phi|$ iff $\Phi$ is satisfyable

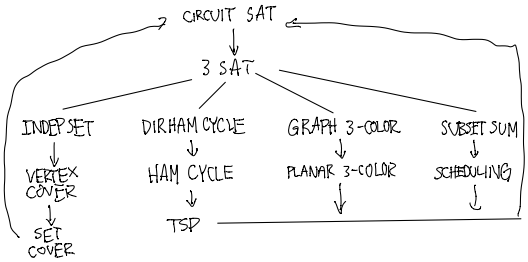



$\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$

# Complexity Class

**Proving NP-ness:** 1. Note $X$ is decision problem and define $|s|$ to input size
2. Define certificate $t$ corresponding to each instance $s$ of $X$, show certifier $C(s,t) =$ "yes" iff $s \in X$
3. Show certificate length is bound by poly $pc(|s|)$

**Notes:** PRIMES $\in P$. FACTOR $\in NP$ and $\in coNP$. Factor $\equiv_p$ Factorize (reduction by binary search)




**INDEP SET $\leftrightarrow_p$ CLIQUE:** Suppose $G$ is indep set $S$ of size $k$, for every distinct edge $(u,v) \in S$; edge $(u,v) \notin G$. Then, $e$ is an edge in the complement graph $G_c$ (graph $G$ but with all and only edges that are not in $E$). Thus $S$ is a clique. Similary, it follows that if $S$ is a clique in $G_c$, then $S$ is in indep set in $G$.

**3-colour $\to_p$ k-color:** Show k-colour $\to_p$ k+1 colour
On input $G$, $G' = (V \cup \{v_0\}, E')$ where $E'$ has edges with $v_0$ connected to all vertices in $V$
(⇒) If $G$ can be coloured with at most k-colours, then $G'$ can be coloured with at most k+1 colours. Add one more colour to $G'$
(⇐) If $G'$ can be coloured with at most k+1 colours, then colour of $v_0$ is different from all other vertices. This means $G$ is k colourable.

**Proving Self reducibility:** Construct search algo using decision algo. Show your algo runs in polytime and prove that it is correct by LI

# Approximations

**$\rho$ approximation:**
- Guaranteed to solve arbitrary instances of problem in polytime
- Finds solution with ratio $\rho$ of true optimum
  Let $C$ be computed value, $C^*$ be opt
- For maximization: $C \geq C^*/\rho$ with $\rho \geq 1$
- For minimization: $C \leq C^*/\rho$ with $\rho \geq 1$

**Weighted VC:** Find min weight subset $S$ (each vertex has $w$)
- Can use LP+ rounding relaxation (edge $e$ pays price $p(e) \geq 0$ to use vertex $i$ and edges incident to vertex $i$ pays $\leq w(i)$ in total)

**Load Balance:** Minimize makespan = $\min(L(i))$ where $L(i) = \sum(t_j$ across jobs $J(i)$ assigned to machine$i)$ of schedule
Greedy by assigning to smallest-load machine by 2-approx.
Pf. Consider load $L_i$ of bottleneck of machine $i$
- Let $j$ be last job scheduled on machine $i$
- When job $j$ assigned to machine $i$, $i$ has smallest load. Its load before assignment is $L_i - t_j \Rightarrow L_i - t_j \leq L_k$ for all $1 \leq k \leq m$
- Sum inequalities over all $k$ and divide by $m$
$$L_i - t_j \leq \frac{1}{m}\sum_k L_k = \frac{1}{m}\sum_j t_j \leq L^* \quad \text{(Lemma 2)}$$
- Now $L = L_i = \underbrace{(L_i - t_j)}_{\text{by Lemma 2}} + t_j \leq 2L^*$ ← by Lemma 1

Lemma 1: optimal makespan $L^* \geq \max_j t_j$

Lemma 2: optimal makespan $L^* \geq \frac{1}{m}\sum_j t_j$

**Properties:**
- There can be no $\rho$-approx if $P \neq NP$
- There is $\rho(n)$ that grows with input size (setcover $\rho(n) = \Omega(\log n)$)
- Polytime approx schemes: For any $\varepsilon > 0$, there is $(1+\varepsilon)$-approx algo (Load Balance, Euclidean TSP, etc), produces high quality sol. Runtime grows quickly with $1/\varepsilon$
- Fully polytime approx schemes: For $\varepsilon > 0$, there is a $(1+\varepsilon)$-approx algo whose runtime is poly in both input size and $1/\varepsilon$ (Knapsack)

Reducing weighted 3-SAT to ILP: Each true clause gains weight $w(k)$.

Sol: make the first row of A equal to $[a^1, a^2, a^3, 0]$ where $a(i) = -1$ if $x_i$ appears in the clause $x_i$ and $a_i$ appears as $\neg x_i$. Add another row $(-a^1, -a^2, -a^3, -l)$ where $l$ is 1 if satisfied given $y^1, y^2, y^3$ as input, 0 otherwise. Let K denote the number of literals negated in the clause and $b = (-1+K, -1+K, 2-K)$. Let $c = (0, 0, 0, w^1)$. Eg

The CNF clause $(x^1, x^2, \neg x^3)$ becomes $y^1 + y^2 + (1 - y^3) \geq 1$