Recall: Bipartite Matching

- Given an undirected graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a matching if each node appears in at most one edge in $M$.
- The maximum matching problem is to find the matching with the most edges.
- We’ll design an efficient algorithm for maximum matching in a bipartite graph. Recall, a graph is bipartite if the nodes $V$ can be partitioned into two sets $V = L \cup R$ such that all edges have one endpoint in $L$ and one endpoint in $R$.

Formulating it as a network flow problem

- Given an instance $G = (L \cup R, E)$ of maximum matching, create a directed graph with nodes $L \cup R \cup \{s, t\}$.
- For each undirected edge $(i, j) \in E$, add a directed edge from $i \in L$ to $j \in R$ with capacity 1.
- Add an edge with capacity 1 from $s$ to each of the nodes in $L$.
- Add an edge with capacity 1 from each of the nodes in $R$ to $t$.
- Claim: The size of the maximum matching in $G$ equals the value of the maximum flow in $G'$.

Reductions

- We just showed how to reduce Matching to NetworkFlow.
  - Given algorithm for NetworkFlow (e.g., Ford-Fulkerson) we can easily solve Matching.
  - Therefore, matching is “no harder” than network flow.
- Definition: Problem $Y$ is poly-time reducible to problem $X$ if:
  - We can solve $Y$ using polynomially many computations + polynomially many calls to black-box algorithm for $X$.
  - Or, if we can solve $X$ in polynomial time, we can solve $Y$ in polynomial time as well.
  - Write $Y \leq_P X$.
- Matching $\leq_P$ NetworkFlow

Reducibility and Intractability

- Claim 1. If $Y \leq_P X$ and $X$ poly-time solvable, so is $Y$.
  - Can use to design algorithms.
- Claim 2. If $Y \leq_P X$ and $Y$ not poly-time solvable, then $X$ is not either.
  - Contrapositive of above.
  - Can be used to prove hardness.
- The catch: we do not know of any problem $Y$ that provably cannot be solved in polynomial time.
A first reduction

**Definition.** \( S \subseteq V \) is an independent set in a graph \( G = (V, E) \) if no nodes in \( S \) share an edge.

**Problem.** Does \( G \) have independent set of size at least \( k \)?

**Example:**
- \( U \) is the set of all skills.
- Each \( S_i \) is a person.
- Want to find a small team that has all skills.

**Theorem.** \( \text{VERTEXCOVER} \leq_P \text{SETCOVER} \)

Reduction #2: Set cover

**Problem.** Given a set \( U \) of \( n \) elements, subsets \( S_1, \ldots, S_m \subseteq U \), and a number \( k \), does there exist a collection of at most \( k \) subsets \( S_i \) whose union is \( U \)?

- **Example:**
  - \( U \) is the set of all skills.
  - Each \( S_i \) is a person.
  - Want to find a small team that has all skills.

**Theorem.** \( \text{VERTEXCOVER} \leq_P \text{SETCOVER} \)

Set cover reduction

**Reduction.** Given \( G = (V, E) \) make set cover instance with \( U = E \) and \( S_v \) is all edges incident to \( v \). \( U \) is independent if and only if \( V \setminus S \) is a vertex cover.

**Proof.**
- If \( S_{v_1}, \ldots, S_{v_k} \) covers \( U \), then every edge adjacent to one of \( \{v_1, \ldots, v_k\} \).
- If \( S_{v_1}, \ldots, S_{v_k} \) is a vertex cover, then \( S_{v_1}, \ldots, S_{v_k} \) covers \( U \).

Common Confusions

- \( Y \leq_P X \) means:
  - \( Y \) is “no harder” than \( X \)
  - \( X \) is “at least as hard” as \( Y \).

- To show \( Y \) is easy, show \( Y \leq_P X \) for easy \( X \).
- To show \( X \) is hard, show \( Y \leq_P X \) for hard \( Y \).

For decision problem \( Y \), need to show two things.

- Correctly outputs Yes and No.
A bad reduction.

Given VertexCover instance \((G, k)\), make SetCover instance with \(U = E\), \(S_v = \text{edges incident to } v\), \(S_0 = U\), and integer \(k\).

- If \(G\) has VC of size at most \(k\), then \(U\) has cover of size at most \(k\).
- But if \(U\) has cover of size \(k\), \(G\) might not!

If \((G, k)\) is a No instance, the reduction does not correctly return No.

Reduction #3: Satisfiability

- Can we determine if a boolean formula has a satisfying assignment?
- Let \(X = \{x_1, \ldots, x_n\}\) be boolean variables
  - A literal is \(x_i\) or \(\bar{x}_i\).
  - A clause is or of several literals \((t_1 \lor t_2 \lor \ldots \lor t_k)\).
  - A formula is and of several clauses
  - An assignment \(v : X \to \{0, 1\}\) gives T/F to each variable.
- \(v\) satisfies formula if all clauses evaluate to True.

Example.
\[(x_1 \lor \bar{x}_2) \land (x_1 \lor x_4 \lor \bar{x}_3) \land (\bar{x}_1 \lor x_4) \land (x_3 \lor x_2)\]

Satisfiability Proof

Claim Graph has IS of size \(n + m\) if and only if formula satisfiable.

- If formula satisfiable, select correct literal on the left and one per clause on the right.
- If graph has IS, 
  - At most one node per clause on the right
  - At most one node per variable on the left.
  - If node selected in clause, its negation cannot be selected.