**3.1.57.** (*) For all $n \in \mathbb{N}$, construct an $n$-vertex tree with domination number 2 in which the minimum size of an independent dominating set is $\lfloor n/2 \rfloor$.

**3.1.58.** (*) Prove that a $K_{1,r}$-free graph $G$ has an independent dominating set of size at most $(r-2)\gamma(G) - (r-3)$. (Hint: Generalize the argument of Theorem 3.1.34.) (Bollobás–Cockayne [1979])

**3.1.59.** (*) In a graph $G$ of order $n$, prove that the minimum size of a connected dominating set is $n$ minus the maximum number of leaves in a spanning tree.

**3.1.60.** (*) For $k \leq 5$, every graph $G$ with $\delta(G) \leq k$ has a connected dominating set of size at most $3n(G)/(k+1)$ (Kleitman–West [1991], Griggs–Wu [1992]). Prove that this is sharp using a graph formed from a cyclic arrangement of $3m$ pairwise-disjoint cliques by making each vertex adjacent to every vertex in the clique before it and the clique after it. Let the clique sizes be $\lceil k/2 \rceil, \lfloor k/2 \rfloor, 1, \lceil k/2 \rceil, \lfloor k/2 \rfloor, 1, \cdots$.

# 3.2. Algorithms and Applications

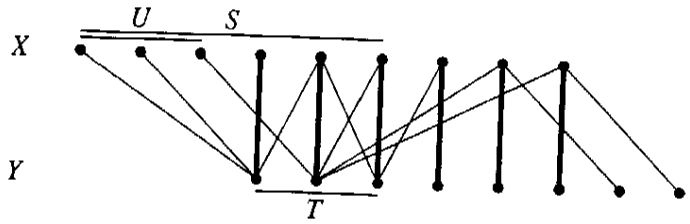
## MAXIMUM BIPARTITE MATCHING

To find a maximum matching, we iteratively seek augmenting paths to enlarge the current matching. In a bipartite graph, if we don't find an augmenting path, we will find a vertex cover with the same size as the current matching, thereby proving that the current matching has maximum size. This yields both an algorithm to solve the maximum matching problem and an algorithmic proof of the König–Egerváry Theorem.

Given a matching $M$ in an $X,Y$-bigraph $G$, we search for $M$-augmenting paths from each $M$-unsaturated vertex in $X$. We need only search from vertices in $X$, because every augmenting path has odd length and thus has ends in both $X$ and $Y$. We will search from the unsaturated vertices in $X$ simultaneously. Starting with a matching of size 0, $\alpha'(G)$ applications of the Augmenting Path Algorithm produce a maximum matching.

**3.2.1. Algorithm.** (Augmenting Path Algorithm).
**Input**: An $X,Y$-bigraph $G$, a matching $M$ in $G$, and the set $U$ of $M$-unsaturated vertices in $X$.
**Idea**: Explore $M$-alternating paths from $U$, letting $S \subseteq X$ and $T \subseteq Y$ be the sets of vertices reached. *Mark* vertices of $S$ that have been explored for path extensions. As a vertex is reached, record the vertex from which it is reached.
**Initialization**: $S = U$ and $T = \emptyset$.
**Iteration**: If $S$ has no unmarked vertex, stop and report $T \cup (X - S)$ as a minimum cover and $M$ as a maximum matching. Otherwise, select an unmarked $x \in S$. To explore $x$, consider each $y \in N(x)$ such that $xy \notin M$. If $y$ is unsaturated, terminate and report an $M$-augmenting path from $U$ to $y$. Otherwise, $y$ is matched to some $w \in X$ by $M$. In this case, include $y$ in $T$ (reached from $x$)

and include $w$ in $S$ (reached from $y$). After exploring all such edges incident to $x$, mark $x$ and iterate. ∎




When exploring $x$ in the iterative step, we may reach a vertex $y \in T$ that we have reached previously. Recording $x$ as the previous vertex on the path may change which $M$-augmenting path we report, but it won't change whether such a path exists.

**3.2.2. Theorem.** Repeatedly applying the Augmenting Path Algorithm to a bipartite graph produces a matching and a vertex cover of equal size.

**Proof:** We need only verify that the Augmenting Path Algorithm produces an $M$-augmenting path or a vertex cover of size $|M|$. If the algorithm produces an $M$-augmenting path, we are finished. Otherwise, it terminates by marking all vertices of $S$ and claiming that $R = T \cup (X - S)$ is a vertex cover of size $|M|$. We must prove that $R$ is a vertex cover and has size $|M|$.

To show that $R$ is a vertex cover, it suffices to show that there is no edge joining $S$ to $Y - T$. An $M$-alternating path from $U$ enters $X$ only on an edge of $M$. Hence every vertex $x$ of $S - U$ is matched via $M$ to a vertex of $T$, and there is no edge of $M$ from $S$ to $Y - T$. Also there is no such edge outside $M$. When the path reaches $x \in S$, it can continue along any edge not in $M$, and exploring $x$ puts all other neighbors of $x$ into $T$. Since the algorithm marks all of $S$ before terminating, all edges from $S$ go to $T$.

Now we study the size of $R$. The algorithm puts only saturated vertices in $T$; each $y \in T$ is matched via $M$ to a vertex of $S$. Since $U \subseteq S$, also each vertex of $X - S$ is saturated, and the edges of $M$ incident to $X - S$ cannot involve $T$. Hence they are different from the edges saturating $T$, and we find that $M$ has at least $|T| + |X - S|$ edges. Since there is no matching larger than this vertex cover, we have $|M| = |T| + |X - S| = |R|$. ∎

In addition to studying the correctness of algorithms, we are concerned about the time (number of computational steps) they use. We measure this as a function of the size of the input. For graph problems, we usually use the order $n(G)$ and/or size $e(G)$ to measure the input size.

**3.2.3. Definition.** The **running time** of an algorithm is the maximum number of computational steps used, expressed as a function of the size of the input. A **good algorithm** is one that has polynomial running time.

Running time is often expressed as "$O(f)$", where $f$ is a function of the