size of the input. Here $O(f)$ denotes the set of functions g such that $|g(x)|$ is bounded by a constant multiple of $|f(x)|$ when x is sufficiently large (that is, there exist c, a such that $|g(x)| \leq c\,|f(x)|$ when $|x| \geq a$).

Many problems we study in Chapters 1-4 have good algorithms; other notions of complexity (Appendix B) need not trouble us yet. Since we don't know how long a particular operation may take on a particular computer, constant factors in running time have little meaning. Hence the "Big Oh" notation $O(f)$ is convenient. When f is a quadratic polynomial, we typically abuse notation by writing $O(n^2)$ instead of $O(f)$ to describe functions that grow at most quadratically in terms of n.

3.2.4. Remark. Let G be an X, Y-bigraph with n vertices and m edges. Since $\alpha'(G) \leq n/2$, we find a maximum matching in G by applying Algorithm 3.2.1 at most $n/2$ times. Each application explores a vertex of X at most once, just before marking it; thus it considers each edge at most once. If the time for one edge exploration is bounded by a constant, then this algorithm to find a maximum matching runs in time $O(nm)$. Theorem 3.2.22 presents a faster algorithm, with running time $O(\sqrt{n}m)$. Section 3.3 discusses a good algorithm for maximum matching in general graphs. ∎

WEIGHTED BIPARTITE MATCHING

Our results on maximum matching generalize to weighted X, Y-bigraphs, where we seek a matching of maximum total weight. If our graph is not all of $K_{n,n}$, then we insert the missing edges and assign them weight 0. This does not affect the numbers we can obtain as the weight of a matching. Thus we assume that our graph is $K_{n,n}$.

Since we consider only nonnegative edge weights, some maximum weighted matching is a perfect matching; thus we seek a perfect matching. We solve both the maximum weighted matching problem and its dual.

3.2.5. Example. *Weighted bipartite matching and its dual.* A farming company owns n farms and n processing plants. Each farm can produce corn to the capacity of one plant. The profit that results from sending the output of farm i to plant j is $w_{i,j}$. Placing weight $w_{i,j}$ on edge $x_i y_j$ gives us a weighted bipartite graph with partite sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$. The company wants to select edges forming a matching to maximize total profit.

The government claims that too much corn is being produced, so it will pay the company not to process corn. The government will pay u_i if the company agrees not to use farm i and v_j if it agrees not to use plant j. If $u_i + v_j < w_{i,j}$, then the company makes more by using the edge $x_i y_j$ than by taking the government payments for those vertices. In order to stop all production, the government must offer amounts such that $u_i + v_j \geq w_{i,j}$ for all i, j. The government wants to find such values to minimize $\sum u_i + \sum v_j$. ∎

3.2.6. Definition. A **transversal** of an n-by-n matrix consists of n positions, one in each row and each column. Finding a transversal with maximum sum is the **Assignment Problem**. This is the matrix formulation of the **maximum weighted matching** problem, where nonnegative weight $w_{i,j}$ is assigned to edge $x_i y_j$ of $K_{n,n}$ and we seek a perfect matching M to maximize the total weight $w(M)$.

With these weights, a **(weighted) cover** is a choice of labels $u_i, \dots, u_n$ and $v_j, \dots, v_n$ such that $u_i + v_j \geq w_{i,j}$ for all i, j. The **cost** $c(u, v)$ of a cover (u, v) is $\sum u_i + \sum v_j$. The **minimum weighted cover** problem is that of finding a cover of minimum cost.

Note that the problem of minimum weight perfect matching can be solved using maximum weight matching; simply replace each weight $w_{i,j}$ with $M - w_{i,j}$ for some large number M.

The next lemma shows that the weighted matching and weighted cover problems are dual problems.

3.2.7. Lemma. For a perfect matching M and cover (u, v) in a weighted bipartite graph G, $c(u, v) \geq w(M)$. Also, $c(u, v) = w(M)$ if and only if M consists of edges $x_i y_j$ such that $u_i + v_j = w_{i,j}$. In this case, M and (u, v) are optimal.

Proof: Since M saturates each vertex, summing the constraints $u_i + v_j \geq w_{i,j}$ that arise from its edges yields $c(u, v) \geq w(M)$ for every cover (u, v). Furthermore, if $c(u, v) = w(M)$, then equality must hold in each of the n inequalities summed. Finally, since $c(u, v) \geq w(M)$ for every matching and every cover, $c(u, v) = w(M)$ implies that there is no matching with weight greater than $c(u, v)$ and no cover with cost less than $w(M)$. ■

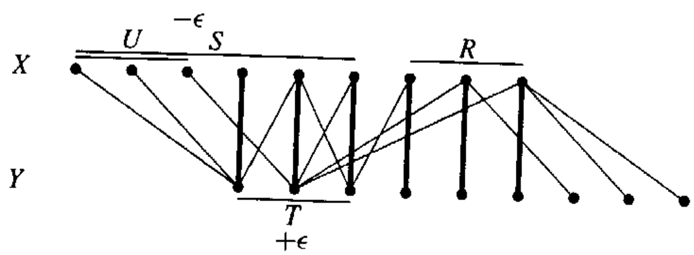
A matching and a cover have the same value only when the edges of the matching are covered with equality. This leads us to an algorithm.

3.2.8. Definition. The **equality subgraph** $G_{u,v}$ for a cover (u, v) is the spanning subgraph of $K_{n,n}$ having the edges $x_i y_j$ such that $u_i + v_j = w_{i,j}$.

If $G_{u,v}$ has a perfect matching, then its weight is $\sum u_i + \sum v_j$, and by Lemma 3.2.7 we have the optimal solution. Otherwise, we find a matching M and a vertex cover Q of the same size in $G_{u,v}$ (by using the Augmenting Path Algorithm, for example). Let $R = Q \cap X$ and $T = Q \cap Y$. Our matching of size $|Q|$ consists of $|R|$ edges from R to $Y - T$ and $|T|$ edges from T to $X - R$, as shown below. To seek a larger matching in the equality subgraph, we change (u, v) to introduce an edge from $X - R$ to $Y - T$ while maintaining equality on all edges of M.

A cover requires $u_i + v_j \geq w_{i,j}$ for all i, j; the difference $u_i + v_j - w_{i,j}$ is the **excess** for i, j. Edges joining $X - R$ and $Y - T$ are not in $G_{u,v}$ and have positive excess. Let ϵ be the minimum excess on the edges from $X - R$ to $Y - T$. Reducing u_i by ϵ for all $x_i \in X - R$ maintains the cover condition for these edges while bringing at least one into the equality subgraph. To maintain the cover condition for the edges from $X - R$ to T, we also increase v_j by ϵ for $y_j \in T$.

We repeat the procedure with the new equality subgraph; eventually we obtain a cover whose equality subgraph has a perfect matching. The resulting algorithm was named the **Hungarian Algorithm** by Kuhn in honor of the work of König and Egerváry on which it is based.



3.2.9. Algorithm. (Hungarian Algorithm—Kuhn [1955], Munkres [1957]).
Input: A matrix of weights on the edges of $K_{n,n}$ with bipartition X, Y.
Idea: Iteratively adjusting the cover (u, v) until the equality subgraph $G_{u,v}$ has a perfect matching.
Initialization: Let (u, v) be a cover, such as $u_i = \max_j w_{i,j}$ and $v_j = 0$.
Iteration: Find a maximum matching M in $G_{u,v}$. If M is a perfect matching, stop and report M as a maximum weight matching. Otherwise, let Q be a vertex cover of size $|M|$ in $G_{u,v}$. Let $R = X \cap Q$ and $T = Y \cap Q$. Let

$$\epsilon = \min\{u_i + v_j - w_{i,j} : x_i \in X - R,\ y_j \in Y - T\}.$$

Decrease u_i by ϵ for $x_i \in X - R$, and increase v_j by ϵ for $y_j \in T$. Form the new equality subgraph and repeat. ∎

We have presented the algorithm using bipartite graphs, but repeatedly drawing a changing equality subgraph is awkward. Therefore, we compute with matrices. The initial weights form a matrix A with $w_{i,j}$ in position i, j. We associate the vertices and the labels (u, v) with the rows and columns, which serve as X and Y, respectively. We subtract $w_{i,j}$ from $u_i + v_j$ to obtain the **excess matrix**: $c_{i,j} = u_i + v_j - w_{i,j}$. The edges of the equality subgraph correspond to 0s in the excess matrix.

3.2.10. Example. *Solving the Assignment Problem.* The first matrix below is the matrix of weights. The others display a cover (u, v) and the corresponding excess matrix. We underscore entries in the excess matrix to mark a maximum matching M of $G_{u,v}$, which appears as bold edges in the equality subgraphs drawn for the first two excess matrices. (Drawing the equality subgraphs is not necessary.) A matching in $G_{u,v}$ corresponds to a set of 0s in the excess matrix with no two in any row or column; call this a **partial transversal**.

A set of rows and columns covering the 0s in the excess matrix is a **covering set**; this corresponds to a vertex cover in $G_{u,v}$. A covering set of size less than n yields progress toward a solution, since the next weighted cover costs less. We study the 0s in the excess matrix and find a partial transversal and a covering set of the same size. In a small matrix, we can do this by inspection.