2.3

Definition

A **weighted graph** G is a graph alongside a function $w : E(G) \to \mathbb{R}^+$.

If G is a weighted graph and $H \subseteq G$, then, $w(H) := \sum_{e \in E(H)} w(e)$.

A **minimum weight spanning tree** (or MWST) is a spanning tree T such that $w(T)$ is minimized among all possible spanning trees. In other words, $w(T) \leq w(T')$ $\forall T' \subseteq G$ where T' is a spanning tree.

Kruskal's Algorithm

INPUT Weighted graph G with n vertices

OUTPUT A MWST, T^* if G is connected, otherwise a message "G is not connected"

STEP 1 Create a list of edges, L_E, in order from smallest weight to largest weight. Start T^* with no edges but all vertices of G.

STEP 2 If the number of edges in T^* is strictly less than $n-1$ AND if there are still edges in L_E, examine the first edge in L_E (i.e., the edge with smallest weight), $e = \{a, b\}$

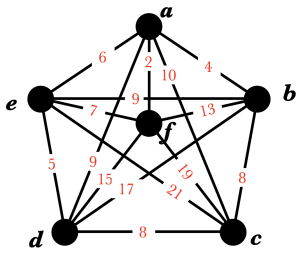
SUBSTEP 2.1 If a and b are in different components of T^*, add e to T^* and remove e from L_E. Return to STEP 2.

SUBSTEP 2.2 If a and b are in the same component, remove e from L_E and do not add to T^*. Return to STEP 2.

STEP 3 If the number of edges in T^* is $n-1$, then output T^*, which is the MWST for G. Otherwise, the number of edges in T^* is strictly less than $n-1$ and G was not connected.

Example

We will find a MWST for the following graph:



First, we create the following table of all the edges in G.

Edge	Cost
af	2
ab	4
de	5
ae	6
ef	7
bc	8
cd	8
ad	9
be	9
ac	10
bf	13
df	15
bd	17
cf	19
ce	21

We can read the following table describing the steps in Kruskal's algorithm from left to right (i.e., we check the edge of lowest weight, then we check the components, then we select our substep).

Edge	Components Of T′	Substep to be used
af	$\{a\},\{b\},\{c\},\{d\},\{e\},\{f\}$	2.1
ab	$\{a,f\},\{b\},\{c\},\{d\},\{e\}$	2.1
de	$\{a,b,f\},\{c\},\{d\},\{e\}$	2.1
ae	$\{a,b,f\},\{c\},\{d,e\}$	2.1
ef	$\{a,b,d,e,f\},\{c\}$	2.2
bc	$\{a,b,d,e,f\},\{c\}$	2.1
—	$\{a,b,c,d,e,f\}$	—

Proof of Kruskal's Algorithm

If G is connected, then Kruskal's Algorithm produces a minimum weight spanning tree.

Let G be a connected graph, and let T_K be the output from Kruskal's algorithm on G. It is easy to check that T_K is a spanning tree, since it is acyclic and has $n(G) - 1$ edges.

Suppose T^* is a MWST with largest edge intersection with T_K — in other words, $|E(T_K) \cap E(T^*)| \geq |E(T_K) \cap E(T')|$ for any other MWST T'.

If $T^* = T_K$, then we are done, since T_K is assumed to be a minimum weight spanning tree. Otherwise, assume toward contradiction that $T^* \neq T_K$. Let e be the first edge chosen by Kruskal's algorithm that is not in T^*. Then, by a previous result, $\exists e' \in E(T^*) - E(T_K)$ such that $T' = T^* + e - e'$ is a spanning tree.

We are assuming, however, that Kruskal's algorithm would choose e over e'. Let $e_1, \ldots, e_k$ be edges in $E(T_K)$ before e. Since e_i was selected before e, we know that $e_i \in E(T^*)$ for each $i \in [k]$.

Let $G_k = (V, \{e_1, \ldots, e_k\})$. we are assuming that Kruskal's algorithm would not choose e' for two reasons:

- If e' shows up before e, then e' would connect two vertices in $G_j = (V, \{e_1, \ldots, e_j\})$. Since $G_j + e' \subseteq T^*$, then T^* contains a cycle and isn't a spanning tree.
- If e' shows up after e in L_E, then $w(e') \geq w(e)$, meaning $w(T') \leq w(T^*)$, meaning that $w(T_K) \leq w(T^*)$, implying that T_K is of a lower weight than T^*.

Dijkstra's Algorithm

We want to find an algorithm to find the shortest path between two vertices in a weighted graph — Dijkstra's Algorithm can solve this.

We know that if P is the shortest u, v path, and $x \in P$, then following P from u to x will yield the shortest u, x path.

Input A weighted graph, G, and $u \in V(G)$

Output For each $z \in V(G)$, the distance $d(u, z)$.

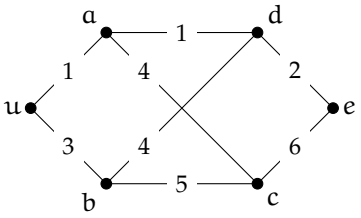
Initialization Extend the weight function such that if $xy \notin E(G)$, then $w(xy) = \infty$. Create S that contains all vertices whose distances from u are known. Let $S := \{u\}$. Let $t : V \to \mathbb{R}^+ \cup \{0, \infty\}$ which will keep track of the tentative distance between u and z. Let $t(z) := w(uz)$ for all $z \neq u$, and $t(u) := 0$.

Condition to Terminate Loop If $t(z) = \infty$ for all $z \notin S$ **or** $S = V$, then go to end.

Loop Else, pick $v \in V - S$ such that $t(v) = \min_{z \notin S} t(z)$. Add v to S. Explore the edges from v to update tentative distances; for each edge vz with $z \notin S$, $t(z) := \min\{t(z), t(v) + w(vz)\}$.

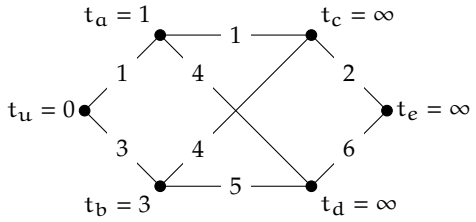
End Set $d(u, v) = t(v)\ \forall v \in V$.

On the following weighted graph, we can do Dijkstra's algorithm as follows:

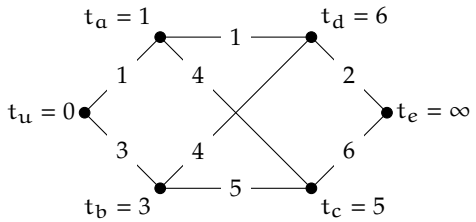


Dijkstra's Algorithm: Worked Example

We let $S = \{u\}$, and include our tentative distances for the first step.

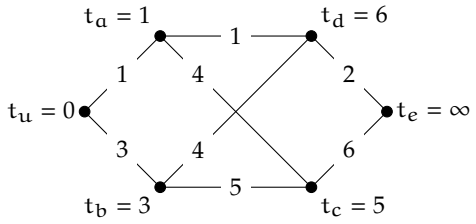


Now, $S = \{u, a\}$ because $t_a \leq t_b$. We now start from a and include our tentative distances.

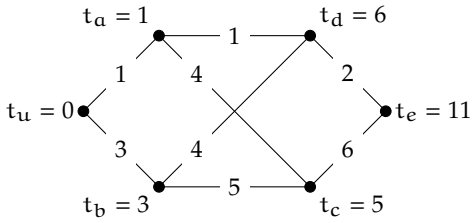


Next, we include b into S, making it $\{u, a, b\}$. We then check our tentative distances from b, where

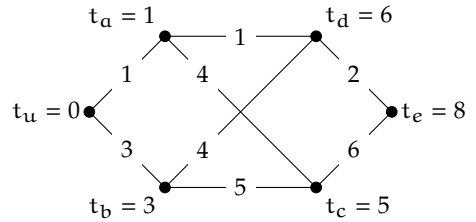
we find that they are no better than tentative distances from a, so we keep the tentative distances on c and d as what they were with a.



Next, we include c into S, making it $\{u, a, b, c\}$, and we update our tentative distances.



Finally, we include d and update tentative distances:



In order to find the direct paths, we can add arrows along the edges that were selected by Dijkstra's algorithm.

The proof can be outlined as follows:

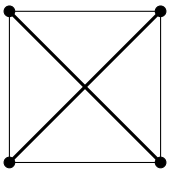
If $z \in S$: then, $t(z) = d(u, z)$.

Else if $z \notin S$: then, $t(z)$ is the length of a shortest u, z path P such that $V(P - z) \subseteq S$.

3.1

Matchings

In a simple graph, a **matching** M is a set of pairwise disjoint edges. In other words, $\forall e_i, e_j \in M$, then $e_i \cap e_j = \emptyset$. In an arbitrary graph, a matching is a set of non-loop edges with no shared endpoints. For example, the thick edges are a matching.



If v is incident to an edge, then v is **saturated** by M, otherwise it is unsatured by M.

A **perfect matching** is a matching that saturates every vertex. We know that all graphs with perfect matchings have even number of vertices, but the alternative case may not be true (for example, we might consider a graph with an isolated vertex).

A **maximal matching** is a matching that cannot be enlarged by adding any other edges. A **maximum** matching is a matching of maximum size among all matchings. In other words, if M^* is a maximum matching, then $|M^*| \geq |M| \; \forall M \in G$.

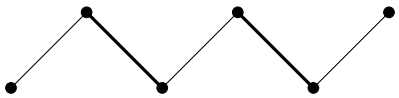
- Not every maximal matching is a maximum matching.
- However, every maximum matching is a maximal matching (because you cannot extend a maximum matching by definition).

Alternating and Augmenting Paths

Given a matching M, an M-**alternating path** is a path that alternates between edges in M and edges not in M. An M-alternating path whose endpoints are unsaturated by M is an M-**augmenting** path.

An M-alternating path:

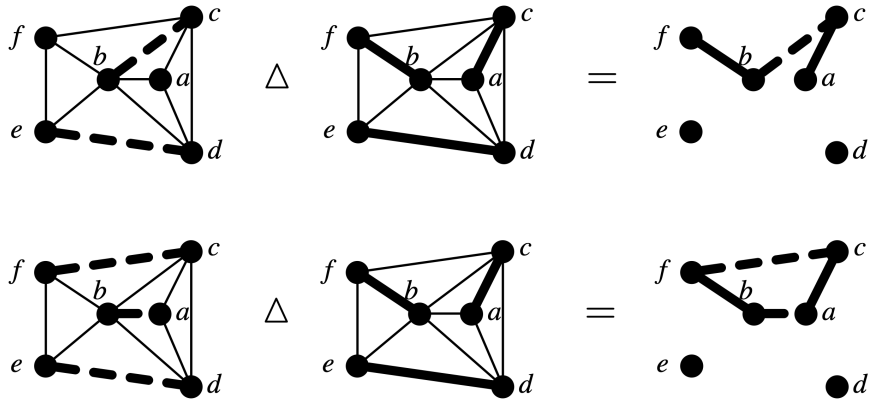
An M-augmenting path:



If M is a matching and there exists an M-augmenting path in G, then M is not a maximum matching (as in the path P that contains M, you can switch the matching edges as follows)

Symmetric Difference

The **symmetric difference** $G \triangle H$ is the subgraph of $G \cup H$ whose edges are the edges of $G \cup H$ that appear in exactly one of G and H. A picture of an example of a symmetric difference between matchings is shown below.



Theorems and Lemmas

Lemma 3.1.9

Every component of the symmetric difference of two matchings is a path or an even cycle.

Let M and M′ be two matchings. Each vertex is incident to at most 1 edge in M and at most one edge in M′. Thus, $d(v) \leq 2$ for each vertex in M and M′. Therefore, each component is either a path or a cycle.

If a component is a cycle, then it must be even because the edges of the cycle must alternate between M and M′.

Theorem 3.1.10

A matching M in a graph G is a *maximum* matching if and only if G has no M-augmenting path.

($\Rightarrow$) Suppose G has an M-augmenting path, P. Exchange the edges of P in M with the edges of P not in M. This action increases the size of the matching by 1, meaning that M was not a maximum matching initially.

($\Leftarrow$) Suppose M is not a maximum matching. Let M′ be a matching with more edges than M (i.e., $|M'| > |M|$). Let $H = (V(G), M \triangle M')$. By Lemma 3.1.9, we know that each component of H is either a path or an even cycle. Since $|M'| > |M|$, there is a component of H that contains more edges from M′ than edges from M, which means it cannot be an even cycle — therefore, this component must be a path that alternates between M′ and M. Because there are more edges from M′ than M in this path, meaning this path is an M-augmenting path.

Perfect Matchings

Let $G = (X, E, Y)$ be a bipartite graph. A matching that saturates X is an X-**perfect matching**.

If there exists a set of vertices $S \subseteq X$ such that $|S| > |N(S)|$, then it is impossible for G to have an

X-perfect matching.

Theorem 3.1.11 (Hall, 1935)

A bipartite graph $G = (X, E, Y)$ has an X-perfect matching if and only if $|S| \leq |N(S)|$ for all $S \subseteq X$.

($\Rightarrow$) If G has an X-perfect matching, then each vertex in X is matched to a distinct vertex in $N(S)$. Thus, $|S| \leq |N(S)|$.

($\Leftarrow$) We will prove via the contrapositive. Suppose G does not have an X-perfect matching. Let M be a maximum matching in G, and let $u \in X$ be an M-unsaturated vertex. Let $S = \{x \in X : \exists\, M\text{-alternating } u, x \text{ path}\}$, and $T = \{y \in Y : \exists M\text{-alternating } u, y \text{ path}\}$.

We can partition S into $\{u\}, X_1, \ldots, X_k$ and T into $Y_1, \ldots, Y_k$, where $|X_i| = |Y_i|$ for $1 \leq i \leq k$. Our partition works as follows: $Y_1 = N(u), X_1 = N_M(Y_1), Y_2 = N(X_1) - Y_1, X_2 = N_M(Y_2)$, and the general form is $Y_i = N(X_{i-1}) - (Y_1 \cup \cdots \cup Y_{i-1})$ and $X_i = N_M(Y_i)$.

Since M is a maximum matching, we know that there are no M-augmenting paths, meaning $|X_i| = |Y_i|$ (otherwise, if $|X_i| < |Y_i|$, we would be able to find a M-augmenting u, y path). Therefore, $|S| = |\{u\}| + |X_1| + \cdots + |X_k|$ while $|T| = |Y_1| + \cdots + |Y_k|$, so $|S| = |T| + 1$.

Corollary 3.1.13

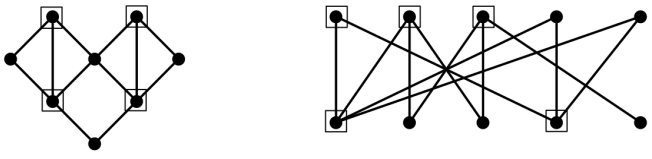
For $k > 0$, every k-regular bipartite graph has a perfect matching.

Let $k > 0$ and let $G = (X, E, Y)$ be a k-regular bipartite graph. We will show that G satisfies Hall's condition. Let $S \subseteq X$ and $[S, N(S)] = \{xy \in E(G) : x \in S \text{ and } y \in N(S)\}$, and $[X, N(S)] = \{xy \in E(G) : x \in X \text{ and } y \in N(S)\}$. Since $[S, N(S)] \subseteq [X, N(S)]$. Additionally, $\sum_{x \in S} d(X) = |[S, N(S)]|$ and $\sum_{y \in X} d(y) = |[X, N(S)]|$.

Since G is k=regular, we find that $|[S, N(S)]| = k|S|$ and $|[X, N(S)]| = k|N(S)|$, meaning that $|S| \leq |N(S)|$

Vertex Covers

A **vertex cover** is a set $Q \subseteq V(G)$ such that Q contains at least one endpoint of every edge. A picture of two vertex covers is shown below:



Of course, a very simple way to find a vertex cover is to select every vertex into Q. However, that isn't particularly useful, so we are focused on *small* vertex covers.

- The size of the **minimum vertex cover** is $\beta(G)$.

- The size of the **maximum matching** is $\alpha'(G)$ (the edge correspondent to $\alpha(G)$ for the size of the maximum independent set).

Alternatively, the minimum vertex cover is denoted $\tau(G)$ and the maximum matching is denoted $\nu(G)$.

An example of minimum vertex covers is shown below:

For any graph G, we can see that $\alpha'(G) \leq \beta(G)$ because to cover the edge of the maximum matching, we will need at least one vertex per edge in the maximum matching. Therefore, the minimum vertex cover must contain within it every vertex corresponding to an edge in the maximum matching, so $\beta(G) \geq \alpha'(G)$.

Theorem 3.1.16 (König-Egervary Theorem)

If G is a bipartite graph, then $\beta(G) = \alpha'(G)$.

Let $G = (X, E, Y)$ be a bipartite graph, and let M be a maximum matching in G. We are going to show that there exists a Q such that $|Q| = |M|$.

For each e in M, with endpoints $x \in X$ and $y \in Y$, select either x or y to be in Q as follows:

Rule: If there exists an M-alternating path that starts at an unsaturated vertex $u \in X$ and ends at y, then $y \in Q$ — else, $x \in Q$.

Claim: Q is a vertex cover. We will use cases to show that every edge has at least one endpoint in Q.

Case 1: Suppose $xy \in M$. Then, $x \in Q \veebar y \in Q$.

Case 2: Suppose $xy \notin M$. If x is M-unsaturated, then y must be M-saturated (or else we would add the edge to the matching). We can create an M-alternating path that starts at x and ends at y, meaning that $y \in Q$.

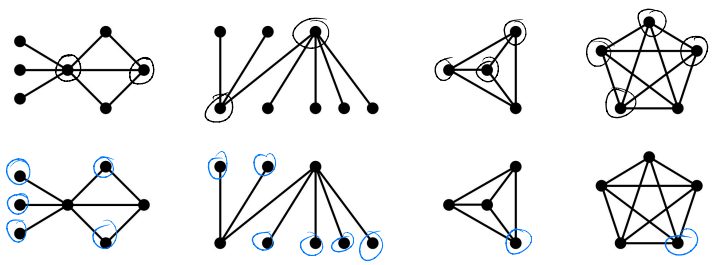
Otherwise, if x is M-saturated, let $xv \in M$ — then, we can find that $v \in Q$ because we create an M-alternating path that starts at $u \in X$ and ends at v. If P does not contain y, then we extend by going from v to x to y.

Edge Covers

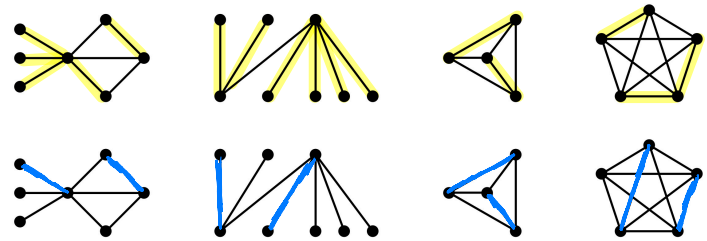
An **edge cover** is a set of edges A such that every vertex is incident to some edge in A. If A is an edge cover, then $\bigcup_{e \in A} e = V(G)$.

The **edge cover number** is the size of a minimum edge cover of G. It is denoted $\beta'(G)$.

We can see a relationship between vertex covers and independent sets as follows, where the top graph shows the minimum vertex cover and the bottom graph shows the maximum independent set.



Similarly, we can see a relationship between minimum edge covers and maximum matchings.



Theorems and Lemmas

Following from the previous result regarding independent sets and vertex covers, we might be able to state the following propostions:

Lemma 3.1.21

In a graph G, $S \subseteq V(G)$ is an independent set if and only if $\overline{S}$ is a vertex cover. Therefore, we get the following result:

$$\alpha(G) + \beta(G) = n(G)$$

S is an independent set.

$\Leftrightarrow \forall u, v \in S, u \nleftrightarrow v$.

$\Leftrightarrow \forall e \in e(G)$, both endpoints of e are not in S.

$\Leftrightarrow$ Both endpoints of e are in $\overline{S}$.

$\Leftrightarrow \overline{S}$ is a vertex cover.

Thus, if S is a maximum independent set, then $\overline{S}$ is a minimum vertex cover, and since $|S| + |\overline{S}| = n(G)$, this means $\alpha(G) + \beta(G) = S$.

Theorem 3.1.22 (Gallai's Theorem)

If G is a graph without isolated vertices, then $\alpha'(G) + \beta'(G) = n(G)$. (i.e., if G is a graph without isolated vertices, then the size of a minimum edge cover summed with the size of the maximum matching equals the number of vertices in G).

We will prove the forward direction first, as follows:

Let M be a maximum matching, and let $A = V(M)$ and $B = \overline{A}$. Since M is a maximum matching, $\nexists e \in E(G)$ such that e connects two edges in B (or else you could simply add it to the matching). Additionally, since there are no isolated vertices in G, $\forall v \in B, \exists w \in A$ such that $v \leftrightarrow w$.

Select one edge such that $e = vw$ for each $v \in B$, and let the set of these edges be N. Then, $C = M \cup N$ is an edge cover, as every vertex in A is covered by M, and every vertex in B is covered by N. So, we have the following:

$$\begin{aligned} |M| + |C| &= |M| + (|M| + |N|) && \text{Since } M \text{ and } N \text{ are disjoint} \\ &= \left(\frac{|A|}{2}\right) + \left(\frac{|A|}{2} + |B|\right) && \text{By the definition of } M \text{ and } N \\ &= n(G) \end{aligned}$$

So, we have that $|M| + |C| = n(G) \geq \alpha'(G) + \beta'(G)$.

Now, we will prove the reverse direction:

Let C be a minimum edge cover. The components of C are stars (there are no cycles or copies of P_4). Let M be a matching formed by taking one edge from each edge in each component of C. Assign each "center" vertex in with an edge from M and assign each "leaf" edge with an edge from C.

Each vertex in $V(G)$ appears exactly once in each star, we have that $n(G) = |M| + |C|$. So, since $|C| = \beta'(G)$ and $|M| \leq \alpha'(G)$, we have that $|M| + |C| = n(G) \leq \alpha'(G) + \beta'(G)$.

Since $n(G) \geq \alpha'(G) + \beta'(G)$ and $n(G) \leq \alpha'(G) + \beta'(G)$, we have that $n(G) = \alpha'(G) + \beta'(G)$.

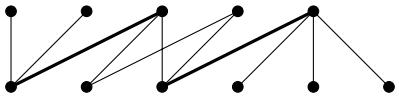
Corollary 3.1.24

If G is a bipartite graph with no isolated vertices, then $\alpha(G) = \beta'(G)$.

We previously showed that $\alpha(G) + \beta(G) = n(G)$ and $\alpha'(G) + \beta'(G) = n(G)$, and by the König-Egerváry theorem, we know that $\alpha'(G) = \beta(G)$ in bipartite graphs with non-isolated vertices, so $\alpha(G) + \beta(G) - \beta(G) = \alpha'(G) - \alpha'(G) + \beta'(G)$, so $\alpha(G) = \beta'(G)$.

Our Hungarian Method for Maximum Matchings

This is a method for finding a maximum matching in a bipartite graph that builds off the augmenting path algorithm. Consider the graph below:



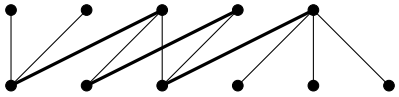
We will find the maximum matching by creating a "Hungarian Forest" as follows:

Initialization In our graph, we set $X_0 = \{x_1, x_2, x_4\}$ and $Y_0 = \{y_2, y_4, y_5, y_6\}$, where $X_0 \cup Y_0$ are the set of M-unsaturated vertices in G.

Iteration Set $X_1 = X_0$ and $Y_1 = N(X_1)$. Then, we do $X_2 = N_M(X_1)$, and $Y_2 = N(X_2) - Y_1$, and so on. The general case is $X_n = N_M(Y_{n-1})$, and $Y_n = N(X_n) - Y_{n-1}$.

End condition If Y_n is empty, then we stop the algorithm.

Let $Y^* = \bigcup Y_i$ (the vertices of Y that are in our Hungarian Forest). If $Y_0 \cap Y^* = \emptyset$, then M is a maximum matching. Otherwise, select $y \in Y_0 \cap Y^*$, and follow the path from y to the root of the tree to create an M-augmenting path, and "flip" the path (exchange matching edges with non-matching edges). In this case, after flipping the path starting between x_4 and y_2, yielding the following graph.



After building our Hungarian forest in the same process, we find that $Y_0 \cap Y^* \neq \emptyset$, so we pick an M-augmenting path starting at y_4, and flip the path starting at y_4 following up to x_1. Afterwards, we find the graph as follows: