VI GRAPHS

A graph is a set of vertices with pairs connected by edges. The information in a particular graph is contained in the choice of vertices that are connected by edges. This simple combinatorial structure is surprisingly versatile and conveniently models situations in which the relationship between parts is important.

20 Trees

Graphs can be used to model and solve many problems. Trees are special graphs. Today, we look at various properties of graphs as well as of trees.

Party problem. Suppose we choose six people at random from a party. We call the people A through F. Then, one of the following must be true:

I. three of the people mutually know each other; or

II. three of the people mutually do not know each other.

We reformulate this claim using a graph representing the situation. We draw six vertices, one for each person, and we draw an edge between two vertices if the two people know each other. We call this a *simple graph*. The *complement graph* consists of the same six vertices but contains an edge between two vertices iff the graph does not have an edge between them. Property I says the graph contains a triangle and Property II says the complement graph contains a triangle. In Figure 24, we see two graphs on six vertices. On the left, the graph contains a triangle and on the right, the complement graph contains a triangle. We can now reformulate the above claim.

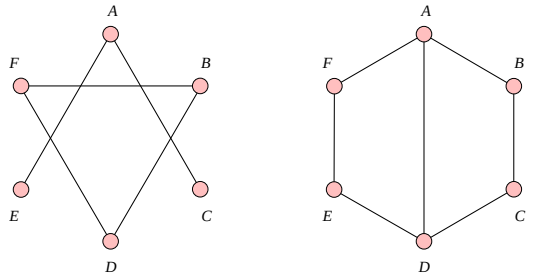

Figure 24: The two cases we consider in the party problem.

PARTY THEOREM. If a simple graph on six vertices does not have a triangle then the complement graph on the same six vertices has a triangle.

PROOF. We distinguish the case in which A knows two or fewer people from the case in which A knows three or more people. For the first case, we assume that A possibly knows C and E but nobody else. If A knows both and C and E know each other, then we have a triangle. Otherwise, consider B, D, F. If they do not all mutually know each other, then without loss of generality, we can say that B does not know D. Thus, we have a triangle in the complement graph since A, B, and D mutually do not know each other. For the second case, assume that A knows B, D, F, and possibly other people. If any two of the three know each other then we have a triangle in the graph. Otherwise, we have a triangle in the complement graph since B, D, and F mutually do not know each other. □

Simple graphs. In the above problem, we used graphs to help us find the solution. A *(simple) graph*, $G = (V, E)$, is a finite set of vertices, V, together with a set of edges, E, where an edge is an unordered pair of vertices. Two vertices connected by an edge are *adjacent* and they are *neighbors* of each other. Furthermore, we say the edge is *incident* to the vertices it connects. Sometimes we refer to the vertices as nodes and the edges as arcs. For example, Figure 25 shows the *complete graph* of five vertices, which we denote as K_5. This graph is complete since we cannot add any more edges. A *subgraph* of a graph $G = (V, E)$ is

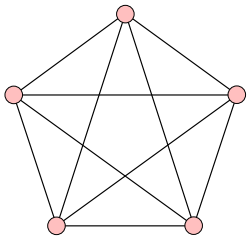

Figure 25: The complete graph of five vertices. It has $\binom{5}{2} = 10$ edges.

a graph $H = (W, F)$ for which $W \subseteq V$ and $F \subseteq E$. For example, a *clique* in G is a subgraph that is a complete graph if considered by itself. With this terminology, the Party Theorem says that a graph on six vertices contains a clique of three vertices or its complement graph contains such a clique.

Connectivity. Suppose we have a graph where the nodes are cities and an edge $\{a, b\}$ exists if there is a train that goes between these two cities. Where can you go if you start at a city x? We need some definitions to study this question. A *walk* is an alternating sequence of vertices and edges such that

1. the sequence starts and ends with a vertex;
2. each edge connects the vertex that precedes the edge with the vertex that succeeds the edge.

Furthermore, a *path* is a walk such that no vertex appears twice. If there exists a walk from a to b, then we know that there also exists a path from a to b. Indeed, if a vertex x appears twice, we can shorten the walk by removing all edges and vertices between the two copies as well as one copy of the vertex x. If the walk is finite, we get a path after finitely many operations as described.

CLAIM. Having a connecting path is an equivalence relation on the vertices of a graph.

PROOF. Let a, b, c be three vertices of a graph G. The relation is reflexive because (a) is a path from a to a. The relation is symmetric because reversing a path from a to b gives a path from b to a. Finally, the relation is transitive because concatenating a path from a to b with a path from b to c gives a path from a to c. □

A graph is *connected* if there is a path between every pair of its vertices. A *connected component* of a not necessarily connected graph is a maximal connected subgraph. Equivalently, a connected component is an equivalence class of vertices together with the edges between them.

Cycles and trees. A *closed walk* is a walk that starts and ends at the same vertex. A *cycle* is a closed walk in which no vertices are repeated. Alternatively, we can say that a cycle is a path in which the start and the end vertices are the same. A *tree* is a connected graph that does not have any cycles.

PROPERTIES OF TREES. If $T = (V, E)$ is a tree with n vertices and m edges, then we have the following properties:

1. there is exactly one path between every pair of vertices;
2. removing an edge disconnects the tree;
3. the number of edges is $m = n - 1$;
4. there exists at least one vertex that has precisely one neighboring vertex.

Spanning trees. Sometimes the whole graph gives us more information than we need or can process. In such a case, we may reduce the graph while preserving the property of interest. For example, if we are interested in connectivity, we may remove as many edges as we can while preserving the connectivity of the graph. This leads to the concept of a *spanning tree*, that is, a subgraph that contains all vertices and is itself a tree.

SPANNING TREE THEOREM. Every finite connected graph contains a spanning tree.

PROOF. If there is a cycle in the graph, remove one edge of the cycle. Repeat until no cycles remain. □

There are a number of different algorithms for constructing a spanning tree. We may, for example, begin with a vertex $u_0 \in V$ and grow a tree by adding an edge and a vertex in each round. This is called Prim's Algorithm.

$W = \{u_0\}$; $F = \emptyset$; $X = V - \{u_0\}$;
`while` $\exists$ edge $\{w, x\}$ with $w \in W$ and $x \in X$ `do`
  move x from X to W; add $\{w, x\}$ to F
`endwhile`;
`if` $V = W$ `then` (W, F) is spanning tree
  `else` (V, E) is not connected
`endif`.

At the end of this algorithm, we have determined if G is connected. If it is connected, we have found a spanning tree. Otherwise, (W, F) is a spanning tree of the connected component that contains u_0.

Rooted trees. In many situations, it is convenient to have the edges of a tree directed, from one endpoint to the other. If we have an edge from a to b, we call a a *parent* of b and b a *child* of a. A particularly important such structure is obtained if the edges are directed such that each vertex has at most one parent. In a tree, the number of edges is one less than the number of vertices. This implies that each vertex has exactly one parent, except for one, the *root*, which has no parent. Holding the root and letting gravity work on the rest of the graph, we get a picture like in Figure 26 in which the root is drawn at the top. The

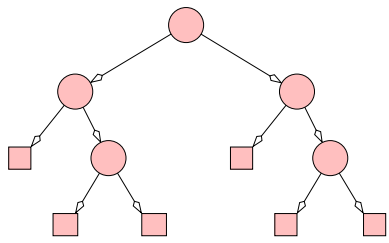

Figure 26: The top vertex is the root and the square vertices are the leaves of the rooted tree.

directions of the edges are now determined, namely from top to bottom, leading away from the root. Each maximal directed path starts at the root and ends at a *leaf*, that is,

a vertex without children. Rooted trees are often used to model or to support a search process. We start at the root and choose an outgoing edge to direct the search to one of the available subtrees. We then recurse, treating the new vertex like the root. Repeating this step, we eventually arrive at a leaf of the rooted tree. Similarly, we can give a recursive definition of a rooted tree. We phrase this definition of the case of a *binary tree*, that is, a rooted tree in which every vertex has at most two children. We thus talk about a left child and a right child.

- an empty graph is a binary tree;
- a vertex (the root) with a binary tree as left subtree and a binary tree as right subtree is a binary tree.

While uncommon in Mathematics, recursive definitions are suggestive of algorithms and thus common in Computer Science.

Summary. Today, we looked at graphs, subgraphs, trees, and rooted trees. We used Prim's algorithm to find a spanning tree (if one exists).

21 Tours

In this section, we study different ways to traverse a graph. We begin with tours that traverse every edge exactly once and end with tours that visit every vertex exactly once.

Bridges of Königsberg. The Pregel River goes through the city of Königsberg, separating it into two large islands and two pieces of mainland. There are seven bridges connecting the islands with the mainland, as sketched in Figure 27. Is it possible to find a closed walk that traverses

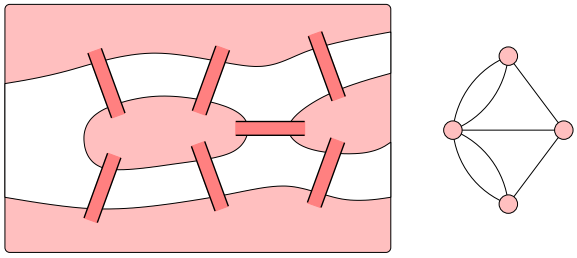

Figure 27: Left: schematic picture of the bridges connecting the islands with the mainland in Königsberg. Right: representation by a graph with four vertices and seven edges.

each bridge exactly once? We can formalize this question by drawing a graph with four vertices, one for each island and each piece of the mainland. We have an edge for each bridge, as in Figure 27 on the right. The graph has *multi-edges* and is therefore not simple. More generally, we may also allow *loops* that are edges starting and ending at the same vertex. A *Eulerian tour* of such a graph is a closed walk that contains each edge exactly once.

Eulerian graphs. A graph is *Eulerian* if it permits a Eulerian tour. To decide whether or not a graph is Eulerian, it suffices to look at the local neighborhood of each vertex. The *degree* of a vertex is the number of incident edges. Here we count a loop twice because it touches a vertex at both ends.

EULERIAN TOUR THEOREM. A graph is Eulerian iff it is connected and every vertex has even degree.

PROOF. If a graph is Eulerian then it is connected and each vertex has even degree just because we enter a vertex the same number of times we leave it. The other direction is more difficult to prove. We do it constructively. Given a vertex $u_0 \in V$, we construct a maximal walk, W_0, that leaves each vertex at a yet unused edge. Starting at u_0, the walk continues until we have no more edge to leave the last vertex. Since each vertex has even degree, this last vertex can only be u_0. The walk W_0 is thus necessarily closed. If it is not a Eulerian tour then there are still some unused edges left. Consider a connected component of the graph consisting of these unused edges and the incident vertices. It is connected and every vertex has even degree. Let u_1 be a vertex of this component that also lies on W_0. Construct a closed walk, W_1, starting from u_1. Now concatenate W_0 and W_1 to form a longer closed walk. Repeating this step a finite number of times gives a Eulerian tour. □

All four vertices of the graph modeling the seven bridges in Königsberg have odd degree. It follows there is no closed walk that traverses each bridge exactly once.

Hamiltonian graphs. Consider the *pentagon dodecahedron*, the Platonic solid bounded by twelve faces, each a regular pentagon. Drawing the corners as vertices and the sides of the pentagons as edges, we get a graph as in Figure 28. Recall that a cycle in a graph is a closed walk

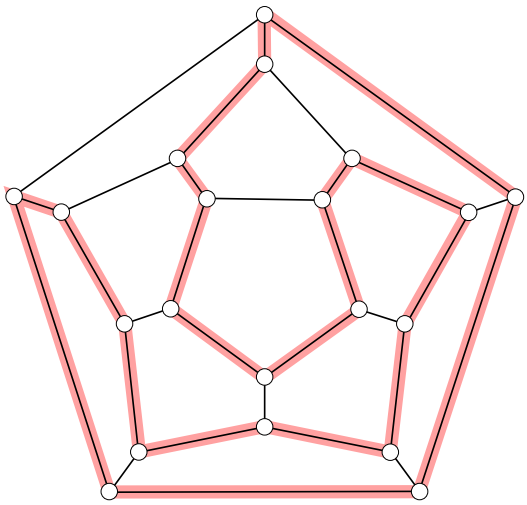

Figure 28: A drawing of a pentagon dodecahedron in which the lengths of the edges are not in scale.

in which no vertex is repeated. A *Hamiltonian cycle* is a closed walk that visits every vertex exactly once. As indicated by the shading of some edges in Figure 28, the graph of the pentagon dodecahedron has a Hamiltonian cycle. A graph is *Hamiltonian* if it permits a Hamiltonian cycle. Deciding whether or not a graph is Hamiltonian turns out to be much more difficult than deciding whether or not it is Eulerian.

A sufficient condition. The more edges we have, the more likely it is to find a Hamiltonian cycle. It turns out that beyond some number of edges incident to each vertex, there is always a Hamiltonian cycle.

DIRAC'S THEOREM. If G is a simple graph with $n \geq 3$ vertices and each vertex has degree at least $\frac{n}{2}$ then G is Hamiltonian.

PROOF. Assume G has a maximal set of edges without being Hamiltonian. Letting x and y be two vertices not adjacent in G, we thus have a path from x to y that passes through all vertices of the graph. We index the vertices along this path, with $u_1 = x$ and $u_n = y$, as in Figure 29. Now suppose x is adjacent to a vertex u_{i+1}. If y is

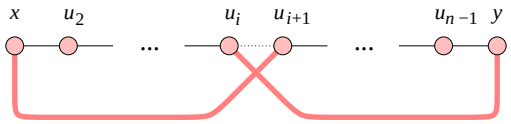

Figure 29: If x is adjacent to u_{i+1} and y is adjacent to u_i then we get a Hamiltonian cycle by adding these two edges to the path and removing the edge connecting u_i to u_{i+1}.

adjacent to u_i then we have a Hamiltonian cycle as shown in Figure 29. Thus, for every neighbor u_{i+1} of x, we have a non-neighbor u_i of y. But x has at least $\frac{n}{2}$ neighbors which implies that y has at least $\frac{n}{2}$ non-neighbors. The degree of y is therefore at most $(n-1) - \frac{n}{2} = \frac{n}{2} - 1$. This contradicts the assumption and thus implies the claim. □

The proof of Dirac's Theorem uses a common technique, namely assuming an extreme counterexample and deriving a contradiction from this assumption.

Summary. We have learned about Eulerian graphs which have closed walks traversing each edge exactly once. Such graphs are easily recognized, simply by checking the degree of all the vertices. We have also learned about Hamiltonian graphs which have closed walks visiting each vertex exactly once. Such graphs are difficult to recognize. More specifically, there is no known algorithm that can decide whether a graph of n vertices is Hamiltonian in time that is at most polynomial in n.

22 Matching

Most of us are familiar with the difficult problem of finding a good match. We use graphs to study the problem from a global perspective.

Marriage problem. Suppose there are n boys and n girls and we have a like-dislike relation between them. Representing this data in a square matrix, as in Figure 30 on the left, we write an 'x' whenever the corresponding boy and girl like each other. Alternatively, we may represent the data in form of a graph in which we draw an edge for each 'x' in the matrix; see Figure 30 on the right. This graph, $G = (V, E)$, is *bipartite*, that is, we can partition the vertex set as $V = X \dot\cup Y$ such that each edge connects a vertex in X with a vertex in Y. The sets X and Y are the *parts* of the graph.

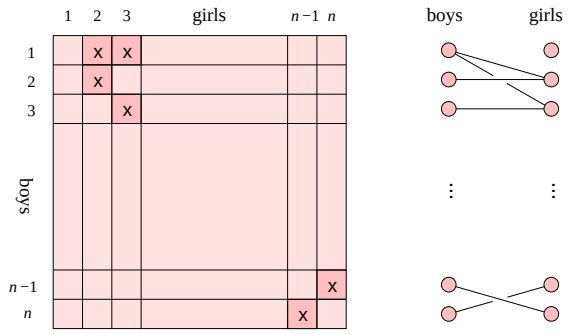

Figure 30: The matrix on the left and the bipartite graph on the right both represent the same data.

The goal is to marry off the boys and girls based on the relation. We formalize this using the bipartite graph representation. A *matching* is a set $M \subseteq E$ of vertex-disjoint edges. The matching is *maximal* if no matching properly contains M. The matching is *maximum* if no matching has more edges than M. Note that every maximum matching is maximal but not the other way round. Maximal matchings are easy to find, eg. by greedily adding one edge at a time. To construct a maximum matching efficiently, we need to know more about the structure of matchings.

Augmenting paths. Let $G = (V, E)$ be a bipartite graph with parts X and Y and $M \subseteq E$ a matching. An *alternating path* alternates between edges in M and edges in $E - M$. An *augmenting path* is an alternating path that begins and ends at unmatched vertices, that is, at vertices that are not incident to edges in M. If we have an augmenting path, we can switch its edges to increase the size of the matching, as in Figure 31.

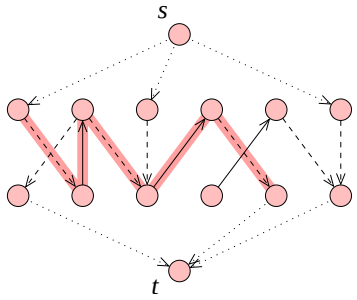

Figure 31: The solid edges form a matching. The shaded edges form an augmenting path. Trading its dashed for its solid edges, we increase the size of the matching by one edge. If we add s and t and direct the edges, the augmenting path becomes a directed path connecting s to t.

BERGE'S THEOREM. The matching M is maximum iff there is no augmenting path.

PROOF. Clearly, if there is an augmenting path then M is not maximum. Equivalently, if M is maximum then there is no augmenting path. Proving the other direction is more difficult. Suppose M is not maximum. Then there exists a matching N with $|N| > |M|$. We consider the symmetric difference obtained by removing the duplicate edges from their union,

$$M \oplus N \quad = \quad (M \cup N) - (M \cap N).$$

Since both sets are matchings, the edges of M are vertex-disjoint and so are the edges of N. It follows that each connected component of the graph $(V, M \oplus N)$ is either an alternating path or an alternating cycle. Every alternating cycle has the same number of edges from M and from N. Since N has more edges than M, it also contributes more edges to the symmetric difference. Hence, at least one component has more edges from N than from M. This is an augmenting path. □

Constructing a maximum matching. Berge's Theorem suggests we construct a maximum matching iteratively. Starting with the empty matching, $M = \emptyset$, we find an augmenting path and increase the size of the matching in each iteration until no further increase is possible. The number of iterations is less than the number of vertices. To explain

how we find an augmenting path, we add vertices s and t to the graph, connecting s to all unmatched vertices in X and t to all unmatched vertices in Y. Furthermore, we direct all edges: from s to X, from X to Y if the edge is in $E-M$, from Y to X if the edge is in M, and from Y to t; see Figure 31. An augmenting path starts and ends with an edge in $E-M$. Prepending an edge from s and appending an edge to t, we get a directed path from s to t in the directed graph. Such a path can be found with breadth-first search, which works by storing active vertices in a queue and marking vertices that have already been visited. Initially, s is the only marked vertex and the only vertex in the queue. In each iteration, we remove the last vertex, x, from the end of the queue, mark all unmarked successors of x, and add these at the front to the queue. We halt the algorithm when t is added to the queue. If this never happens then there is no augmenting path and the matching is maximum. Otherwise, we extract a path from s to t by tracing it from the other direction, starting at t, adding one marked vertex at a time.

The breadth-first search algorithm takes constant time per edge. The number of edges is less than n^2, where $n = |V|$. It follows that an augmenting path can be found in time $O(n^2)$. The overall algorithms takes time $O(n^3)$ to construct a maximum matching.

Vertex covers. Running the algorithm to completion, we get a maximum matching, $M \subseteq E$. Let Y_0 contain all vertices in Y reachable from s and X_0 all vertices in X from which t is reachable; see Figure 32. No edge in M

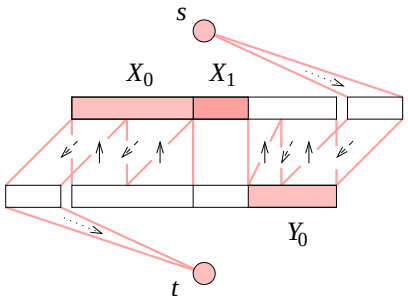

Figure 32: Schematic picture of the vertex set D consisting of the shaded portions of X and of Y. The vertices are ordered so that all edges in M are vertical.

connects a vertex in X_0 with a vertex in Y_0, else we would have an augmenting path. Furthermore, $|X_0 \cup Y_0| \leq |M|$ because each vertex in the union is incident to an edge in the matching. Letting X_1 contain the endpoints of the yet untouched edges in M, we set $D = X_0 \cup Y_0 \cup X_1$ and note that $|D| = |M|$. Furthermore, we observe that D *covers* all edges in E, that is, each edge has at least one endpoint in D.

We generalize this concept. Given a graph $G = (V, E)$, a *vertex cover* is a set $C \subseteq V$ such that each edge in E has at least one endpoint in C. It is *minimal* if it does not properly contain another vertex cover and *minimum* if there is no vertex cover with fewer vertices. Finding a minimal vertex cover is easy, eg. by greedily removing one vertex at a time, but finding a minimum vertex cover is a difficult computational problem for which no polynomial-time algorithm is known. However, if G is bipartite, we can use the maximum matching algorithm to construct a minimum vertex cover.

KÖNIG'S THEOREM. If $G = (V, E)$ is bipartite then the size of a minimum vertex cover is equal to the size of a maximum matching.

PROOF. Let X and Y be the parts of the graph, $C \subseteq V = X \dot\cup Y$ a minimum vertex cover, and $M \subseteq E$ a maximum matching. Then $|M| \leq |C|$ because C covers M. Since M is maximum, there is no augmenting path. It follows that the set $D \subseteq V$ (as defined above) covers all edges. Since C is minimum, we have $|C| \leq |D| = |M|$, which implies the claim. □

Neighborhood sizes. If the two parts of the bipartite graph have the same size it is sometimes possible to match every last vertex. We call a matching *perfect* if $|M| = |X| = |Y|$. There is an interesting relationship between the existence of a perfect matching and the number of neighbors a set of vertices has. Let $S \subseteq X$ and define its *neighborhood* as the set $N(S) \subseteq Y$ consisting of all vertices adjacent to at least one vertex in S.

HALL'S THEOREM. In a bipartite graph $G = (V, E)$ with equally large parts X and Y, there is a perfect matching iff $|N(S)| \geq |S|$ for every $S \subseteq X$.

PROOF. If all vertices of X can be matched then $|N(S)| \geq |S|$ simply because $N(S)$ contains all matched vertices in Y, and possibly more. The other direction is more difficult to prove. We show that $|N(S)| \geq |S|$ for all $S \subseteq X$ implies that X is a minimum vertex cover. By König's Theorem, there is a matching of the same size, and this matching necessarily connects to all vertices in X.

Let now $C \subseteq X \dot\cup Y$ be a minimum vertex cover and consider $S = X - C$. By definition of vertex cover, all

neighbors of vertices in S are in $Y \cap C$. Hence, $|S| \leq |N(S)| \leq |Y \cap C|$. We therefore have

$$\begin{aligned} |C| &= |C \cap X| + |C \cap Y| \\ &\geq |C \cap X| + |S| \\ &= |C \cap X| + |X - C| \end{aligned}$$

which is equal to $|X|$. But X clearly covers all edges, which implies $|C| = |X|$. Hence, X is a minimum vertex cover, which implies the claim. □

Summary. Today, we have defined the marriage problem as constructing a maximum matching in a bipartite graph. We have seen that such a matching can be constructed in time cubic in the number of vertices. We have also seen connections between maximum matchings, minimum vertex covers, and sizes of neighborhoods.

23 Planar Graphs

Although we commonly draw a graph in the plane, using tiny circles for the vertices and curves for the edges, a graph is a perfectly abstract concept. We now talk about constraints on graphs necessary to be able to draw a graph in the plane without crossings between the curves. This question forms a bridge between the abstract and the geometric study of graphs.

Drawings and embeddings. Let $G = (V, E)$ be a simple, undirected graph and let $\mathbb{R}^2$ denote the two-dimensional real plane. A *drawing* maps every vertex $u \in V$ to a point $\varepsilon(u)$ in $\mathbb{R}^2$, and it maps every edge $uv \in E$ to a curve with endpoints $\varepsilon(u)$ and $\varepsilon(v)$; see Figure 33. The drawing is an *embedding* if

1. vertices are mapped to distinct points;
2. edge are mapped to curves without self-intersections;
3. a curve does not pass through a point, unless the corresponding edge and vertex are incident, in which case the point is an endpoint of the curve;
4. two curves are disjoint, unless the corresponding edges are incident to a common vertex, in which case the curves share a common endpoint.

Not every graph can be drawn without crossings between the curves. The graph G is *planar* if it has an embedding in the plane.

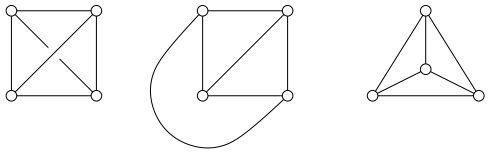

Figure 33: Three drawings of K_4. From left to right a drawing that is not an embedding, an embedding with one curved edge, and a straight-line embedding.

Euler's Formula. Think of the plane as an infinite piece of paper which you cut along the curves with a pair of scissors. Each piece of the paper that remains connected after the cutting is called a *face* of the embedding. We write $n = |V|$, $m = |E|$, and ℓ for the number of faces. Euler's Formula is a linear relation between the three numbers.

EULER'S FORMULA. For an embedding of a connected graph we have $n - m + \ell = 2$.

PROOF. Choose a spanning tree (V, T) of (V, E). It has n vertices, $|T| = n - 1$ edges, and one face. We have $n - (n - 1) + 1 = 2$, which proves the formula if G is a tree. Otherwise, draw the remaining edges, one at a time. Each edge decomposes one face into two. The number of vertices does not change, m increases by one, and ℓ increases by one. Since the graph satisfies the claimed linear relation before drawing the edge it satisfies the relation also after drawing the edge. □

We get bounds on the number of edges and faces, in terms of the number of vertices, by considering *maximally connected* graphs for which adding any other edge would violate planarity. Every face of a maximally connected planar graph with three or more vertices is necessarily a triangle, for if there is a face with more than three edges we can add an edge without crossing any other edge. Let $n \geq 3$ be the number of vertices, as before. Since every face has three edges and every edge belong to two triangles, we have $3\ell = 2m$. We use this relation to rewrite Euler's Formula: $n - m + \frac{2m}{3} = 2$ and $n - \frac{3\ell}{2} + \ell = 2$ and therefore $m = 3n - 6$ and $\ell = 2n - 4$. Every planar graph can be completed to a maximally connected planar graph, which implies that it has at most these numbers of edges and faces.

Note that the sum of vertex degrees is twice the number of edges, and therefore $\sum_u \deg(u) \leq 6n - 12$. It follows that every planar graph has a vertex of degree less than six. We will see uses of this observation in coloring planar graphs and in proving that they have straight-line embeddings.

Non-planarity. We can use the consequences of Euler's Formula to prove that the complete graph of five vertices and the complete bipartite graph of three plus three vertices are not planar. Consider first K_5, which is drawn in Figure 34, left. It has $n = 5$ vertices and $m = 10$ edges,

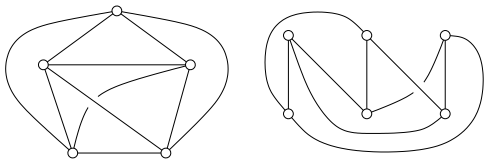

Figure 34: K_5 on the left and $K_{3,3}$ on the right.

contradicting the upper bound of at most $3n-6 = 9$ edges for maximally connected planar graphs. Consider second $K_{3,3}$, which is drawn in Figure 34, right. It has $n = 6$ vertices and $m = 9$ edges. Each cycle has even length,

which implies that each face has four or more edges. We get $4\ell \leq 2m$ and $m \leq 2n - 4 = 8$ after plugging the inequality into Euler's Formula, again a contradiction.

In a sense, K_5 and $K_{3,3}$ are the quintessential non-planar graphs. Two graphs are *homeomorphic* if one can be obtained from the other by a sequence of operations, each deleting a degree-2 vertex and merging its two edges into one or doing the inverse.

KURATOWSKI'S THEOREM. A graph G is planar iff no subgraph of G is homeomorphic to K_5 or to $K_{3,3}$.

The proof of this result is a bit lengthy and omitted. We now turn to two applications of the structural properties of planar graphs we have learned.

Vertex coloring. A *vertex k-coloring* is a map $\chi : V \to \{1, 2, \ldots, k\}$ such that $\chi(u) \neq \chi(v)$ whenever u and v are adjacent. We call $\chi(u)$ the *color* of the vertex u. For planar graphs, the concept is motivated by coloring countries in a geographic map. We model the problem by replacing each country by a vertex and by drawing an edge between the vertices of neighboring countries. A famous result is that every planar graph has a 4-coloring, but proving this fills the pages of a thick book. Instead, we give a constructive argument for the weaker result that every planar graph has a 5-coloring. If the graph has five or fewer vertices then we color them directly. Else we perform the following four steps:

`Step 1.` Remove a vertex $u \in V$ with degree $k = \deg(u) \leq 5$, together with the k incident edges.

`Step 2.` If $k = 5$ then find two neighbors v and w of the removed vertex u that are not adjacent and merge them into a single vertex.

`Step 3.` Recursively construct a 5-coloring of the smaller graph.

`Step 4.` Add u back into the graph and assign a color that is different from the colors of its neighbors.

Why do we know that vertices v and w in Step 2 exist? To see that five colors suffice, we just need to observe that the at most five neighbors of u use up at most four colors. The idea of removing a small-degree vertex, recursing for the remainder, and adding the vertex back is generally useful. We show that it can also be used to construct embeddings with straight edges.

Convexity and star-convexity. We call a region S in the plane *convex* if for all points $x, y \in S$ the line segment with endpoints x and y is contained in S. Figure 35 shows examples of regions of either kind. We call S *star-convex*

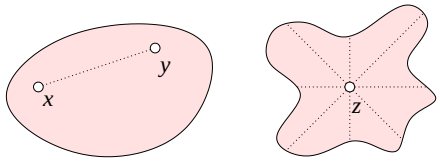

Figure 35: A convex region on the left and a non-convex star-convex region on the right.

if there is a point $z \in S$ such that for every point $x \in S$ the line segment connecting x with z is contained in S. The set of such points z is the *kernel* of S.

It is not too difficult to show that every pentagon is star-convex: decompose the pentagon using two diagonals and choose z close to the common endpoint of these diagonals, as shown in Figure 36. Note however that not every hexagon is star-convex.

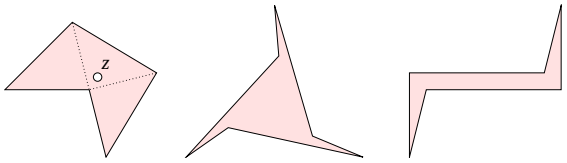

Figure 36: A (necessarily) star-convex pentagon and two non-star-convex hexagons.

Straight-line embedding. A *straight-line embedding* maps every (abstract) edge to the straight line segment connecting the images of its two vertices. We prove that every planar graph has a straight-line embedding using the fact that it has a vertex of degree at most five. To simplify the construction, we assume that the planar graph G is maximally connected and we fix the 'outer' triangle abc. Furthermore, we observe that if G has at least four vertices then it has a vertex of degree at most 5 that is different from a, b and c. Indeed, the combined degree of a, b, c is at least 7. The combined degree of the other $n-3$ vertices is therefore at most $6n - 19$, which implies the average is still less than 6, as required.

`Step 1.` Remove a vertex $u \in V - \{a, b, c\}$ with degree $k = \deg(u) \leq 5$, together with the k incident

edges. Add $k-3$ edges to make the graph maximally connected again.

`Step 2.` Recursively construct a straight-line embedding of the smaller graph.

`Step 3.` Remove the added $k-3$ edges and map u to a point $\varepsilon(u)$ inside the kernel of the k-gon. Connect $\varepsilon(u)$ with line segments to the vertices of the k-gon.

Figure 37 illustrates the recursive construction. It would be fairly straightforward to turn the construction into a recursive algorithm, but the numerical quality of the embeddings it gives is not great.

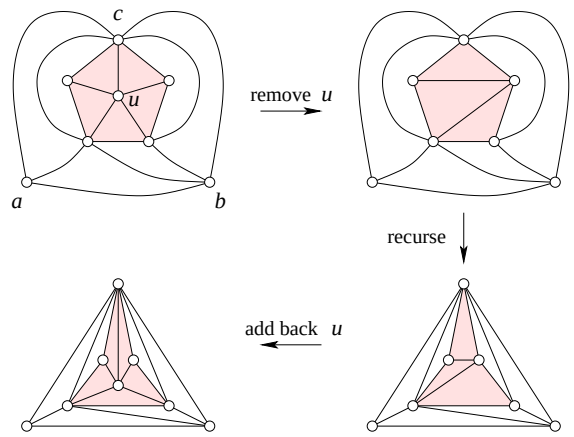

Figure 37: We fix the outer triangle abc, remove the degree-5 vertex u, recursively construct a straight-line embedding of the rest, and finally add the vertex back.

Sixth Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is 22 April 2009.

Question 1. ($20 = 5 + 5 + 5 + 5$ points). Choose ten of your friends, and make a graph where the edges represent two friends being Facebook friends. (Do not include yourself in the graph). Order your friends alphabetically, and label the vertices $v_1, v_2, \ldots, v_{10}$ respectively. This will be most interesting if all of your friends know each other. Now, answer the following questions about the graph that you drew.

(a) What is the size of the largest clique?

(b) Find the shortest and longest paths from v_1 to v_{10}.

(c) Which vertex has the highest degree?

(d) Use Prim's algorithm to find the minimum spanning tree, and draw that tree.

Question 2. (20 points). (Problem 6.1-14 in our textbook). Are there graphs with n vertices and $n - 1$ edges and no cycles that are not trees? Give a proof to justify your answer.

Question 3. (20 points). Call a simple graph with $n \geq 3$ vertices an *Ore graph* if every pair of non-adjacent vertices has a combined degree of at least n. Is it true that every Ore graph is Hamiltonian? Justify your answer.

Question 4. ($20 = 10 + 10$ points). (Problems 6.4-12 and 13 in our textbook). Prove or give a counterexample:

(a) Every tree is a bipartite graph.

(b) A bipartite graph has no odd cycles.

Question 5. ($20 = 5 + 15$ points). Suppose you have n pennies which you arrange flat on a table, without overlap.

(a) How would you arrange the pennies to maximize the number of pennies that touch each other?

(b) Prove that the number of touching pairs cannot exceed $3n$.