Graph Theory Fall 2019 – EPFL – Lecture Notes.

Lecturer: Friedrich Eisenbrand

November 29, 2019

Acknowledgements: These notes are heavily based on the lecture notes of the Graph Theory courses given by István Tomon, Frank de Zeeuw, Andrey Kupavskii and Dániel Korándi of the DCG group.

Lecture 1

Introduction

---

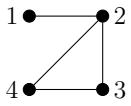
1 Definitions

Definition. *A* graph $G = (V, E)$ *consists of a finite set* V *and a set* E *of two-element subsets of* V*. The elements of* V *are called* vertices *and the elements of* E *are called* edges*.*

For instance, very formally we can introduce a graph like this:

$$V = \{1, 2, 3, 4\}, \quad E = \{\{1, 2\}, \{3, 4\}, \{2, 3\}, \{2, 4\}\}.$$

In practice we more often think of a drawing like this:



Technically, this is what is called a *labelled graph*, but we often omit the labels. When we say something about an unlabelled graph like  , we mean that the statement holds for any labelling of the vertices.

Here are two examples of related objects that we do not consider graphs in this course:

The first is a *multigraph*, which can have multiple edges and loops; the corresponding definition would allow the edge set and the edges to be multisets. The second is a *directed graph*, in which every edge has a direction; in the corresponding definition the edges would be ordered pairs instead of two-element subsets.

Although this course is mostly not about these variants, in some cases it will be more natural to state our results for directed or multigraphs. In any case, we will not treat *infinite graphs* in this course.

Graphs (and their above-mentioned variants) are highly applicable in- and outside mathematics because they provide a simple way of modeling many concepts involving connections between objects. For example, graphs can model social networks (vertices=people & edges=friendships), computer networks (computers & links), molecules (atoms & bonds) and many other things. The aim of this course is to study graphs in the abstract sense, and to introduce the fundamental concepts, tools, tricks and results about them.

Some notation: Given a graph G, we write $V(G)$ for the vertex set, and $E(G)$ for the edge set. For an edge $\{x, y\} \in E(G)$, we usually write xy, and we consider yx to be the same edge. If $xy \in E(G)$, then we say that $x, y \in V(G)$ are *adjacent* or *connected* or that they are *neighbors*. If $x \in e$, then we say that $x \in V(G)$ and $e \in E(G)$ are *incident*.

Definition (Subgraphs)**.** *Two graphs* G, G' *are* isomorphic *if there is a bijection* $\varphi : V(G) \to V(G')$ *such that* $xy \in E(G)$ *if and only if* $\varphi(x)\varphi(y) \in E(G')$*. A graph* H *is a* subgraph *of a graph* G*, denoted* $H \subset G$*, if there is a graph* H' *isomorphic to* H *such that* $V(H') \subset V(G)$ *and* $E(H') \subset E(G)$*.*

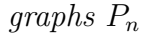
With this definition we can for instance say that  is a subgraph of  . As mentioned above, when we talk about graphs we often omit the labels of the vertices. A more formal way of doing this is to define an *unlabelled graph* to be an isomorphism class of labelled graphs. We will be somewhat informal about this distinction, since it rarely leads to confusion.

Definition (Degree)**.** *Fix a graph $G = (V, E)$. For $v \in V$, we write*

$$N(v) = \{w \in V : vw \in E\}$$

for the set of neighbors of v (which does not include v). Then $d(v) = |N(v)|$ is the degree *of v. We write $\delta(G)$ for the minimum degree of a vertex in G, and $\Delta(G)$ for the maximum degree.*

Definition (Examples)**.** *The following are some of the most common types of graphs.*

- Paths *are the graphs P_n of the form*  *. The graph P_n has $n-1$ edges and n different vertices; we say that P_n has* length $n-1$.
- Cycles *are the graphs C_n of the form*  *. The graph C_n has n edges and n different vertices; the* length *of C_n is defined to be n.*
- Complete graphs *(or* cliques*) are the graphs K_n on n vertices in which all vertices are adjacent. The graph K_n has $\binom{n}{2}$ edges. For instance, K_4 is*  *.*
- *The* complete bipartite graphs *are the graphs $K_{s,t}$ with a partition $V(K_{s,t}) = X \cup Y$ with $|X| = s, |Y| = t$, such that every vertex of X is adjacent to every vertex of Y, and there are no edges inside X or Y. Then K_{st} has st edges. For example, $K_{2,3}$ is*  *.*

The following are the most common properties of graphs that we will consider.

Definition (Bipartite)**.** *A graph G is* bipartite *if there is a partition $V(G) = X \cup Y$ such that every edge of G has one vertex in X and one in Y; we call such a partition a* bipartition*.*

Definition (Connected)**.** *A graph G is* connected *if for all $x, y \in V(G)$ there is a path in G from x to y (more formally, there is a path P_k which is a subgraph of G and whose endpoints are x and y).*
A connected component *of G is a maximal connected subgraph of G (i.e., a connected subgraph that is not contained in any larger connected subgraph). The connected components of G form a partition of $V(G)$.*

2 Basic Facts

In this section we prove some basic facts about graphs. It is a somewhat arbitrary collection of statements, but we introduce them here to get used to the terminology and to see some typical proof techniques.

Proposition 1.1. *In any graph G we have* $\sum_{v \in V(G)} d(v) = 2|E(G)|$.

Proof. We count the number of pairs $(v, e) \in V(G) \times E(G)$ such that $v \in e$, in two different ways. On the one hand, a vertex v is involved in $d(v)$ such pairs, so the total number of such pairs is $\sum_{v \in V(G)} d(v)$. On the other hand, every edge is involved in two such pairs, so the number of pairs must equal $2|E(G)|$. □

What we used here is a very powerful proof technique in combinatorics, called *double counting*. The lemma itself is sometimes called the "handshake lemma" because it says that at a party the number of shaken hands is twice the number of handshakes. It has useful corollaries, such as the fact that the number of odd-degree vertices in a graph must be even.

Next, we will describe a very important characterization of bipartite graphs. But first, we need two more definitions.

Definition (Walk)**.** *A* walk *is a sequence $v_1e_1v_2e_2\dots v_k$ of (not necessarily distinct) vertices v_i and edges e_i such that $e_i = v_iv_{i+1}$. A* closed walk *is a walk with $v_1 = v_k$. The length of this walk is the number of edges, $k-1$.*

It is easy to see that paths are exactly walks with no repeating vertices, and cycles are exactly closed walks with no repeating vertices apart from $v_1 = v_k$.

Definition (Distance)**.** *The* distance *$d(u,v)$ of two vertices $u, v \in V(G)$ is the length of the shortest path (or walk) in G from u to v. (If there is no u-v path in G then $d(u,v) = \infty$.)*

Now we are ready to prove the characterization. Note that "contains a cycle" means that the graph has a subgraph that is isomorphic to some C_n, and similarly for paths. An "odd cycle" is just a cycle whose length is odd.

Theorem 1.2. *A graph is bipartite if and only if it contains no odd cycle.*

Proof. To prove the easy direction of the statement, suppose that G is bipartite with bipartition $V(G) = X \cup Y$, and let $v_1 \cdots v_k v_1$ be a cycle in G with, say, $v_1 \in X$. We must have $v_i \in X$ for all odd i and $v_i \in Y$ for all even i. Since v_k is adjacent to v_1, it must be in Y, so k must be even and the cycle is not odd.

Now for the other direction, suppose G has no odd cycles. We may assume that G is connected. Indeed, otherwise we can apply the same argument to each connected component G_i of G to get a bipartition $X_i \cup Y_i$ of G_i. Choosing $X = \cup_i X_i$ and $Y = \cup_i Y_i$ will then give a bipartition of G.

So if v is a fixed vertex, then every other vertex $u \in V(G)$ has finite distance from v. Let

$$X = \{u : \text{distance of } v \text{ and } u \text{ is } \textit{even}\}$$
$$Y = \{u : \text{distance of } v \text{ and } u \text{ is } \textit{odd}\}.$$

Our aim is to prove that this is a bipartition of G. For this, we need to check that no two vertices in X are adjacent and no two vertices in Y are adjacent.

Suppose for contradiction that some two vertices $u_1, u_2 \in X$ are adjacent, and let e be the edge u_1u_2. By construction, there are paths P_1 from v to u_1 and P_2 from u_2 to v that both have even lengths. But then joining P_1, P_2 and the edge e gives a closed walk P_1eP_2 of odd length, so by Claim 1.3 below, G contains an odd cycle, as well, contradiction the assumption. (Note that P_1eP_2 is not necessarily a cycle because P_1 and P_2 might intersect!)

We can do the same to show that no two vertices $u_1, u_2 \in Y$ are adjacent: here the paths P_1, P_2 will both have odd lengths, so again P_1eP_2 is a closed odd walk. So $X \cup Y$ is indeed a bipartition of G. □

The proof above is a *constructive argument*, where we explicitly constructed the object we were looking for (and then proved that it satisfies the required properties). Of course, we still need to prove the following lemma to complete the proof. We use an *inductive argument*.

Claim 1.3. *Every closed walk of odd length contains an odd cycle.*

Proof. We apply induction on the length k of the walk. Since there is no closed walk of length 1, the statement is vacuously true for $k = 1$. We could use this as the base case, but it is also easy to see that the only closed walk of length 3 is the triangle (K_3), which itself is an odd cycle.

Now suppose the statement is true for every odd length $< k$, and let $W = v_1e_1v_2 \dots v_{k+1}$ be a closed walk of odd length k. Let j be the smallest index such that $v_i = v_j$ for some $i < j$. We have two cases. If $j - i$ is even, then deleting the $j - i$ edges $e_i, \dots, e_{j-1}$ from W yields another closed walk $W' \subseteq W$ of odd length. Applying induction on W' then gives an odd cycle in W' and hence in W.

On the other hand, if $j - i$ is odd (and it cannot be 1, so $j - i \geq 3$), then the $j - i$ edges $e_i, \dots, e_{j-1}$ form an odd cycle. Indeed, they form an odd walk without repeated vertices by the choice of v_j. This is what we were looking for. □

3 Second Lecture: Teaser

The next theorem shows that if a graph has many edges, then it must contain a long path.

Theorem 1.4. *Let $k \geq 2$ be an integer and let G be a graph on n vertices with at least $(k-1)n$ edges. Then G contains a path of length k.*

The proof of this theorem is the combination of two lemmas, which are interesting on their own.

First, we consider the special case when every degree of G is large.

Lemma 1.5. *If the minimum degree of G is k, then G contains a path of length k.*

Proof. Let $v_1 \cdots v_l$ be a maximal path in G, i.e., a path that cannot be extended. Then any neighbor of v_1 must be on the path, since otherwise we could extend it. Since v_1 has at least k neighbors, the set $\{v_2, \dots, v_l\}$ must contain at least k elements. Hence $l \geq k + 1$, so the path has length at least k. □

Note that in general this bound cannot be improved, because the complete graph K_{k+1} has minimum degree k, but its longest path has length k. In problem set 1, we will prove an analogous statement for cycles.

Lecture 2

Basic results. Trees.

---

1 More basic facts

The next theorem shows that if a graph has many edges, then it must contain a long path.

Theorem 2.6. *Let $k \geq 2$ be an integer and let G be a graph on n vertices with at least $(k-1)n$ edges. Then G contains a path of length k.*

Proof. The proof of this theorem is the combination of two lemmas, which are interesting on their own.

First, we consider the special case when every degree of G is large.

Lemma 2.7. *If the minimum degree of G is k, then G contains a path of length k.*

Proof. Let $v_1 \cdots v_l$ be a maximal path in G, i.e., a path that cannot be extended. Then any neighbor of v_1 must be on the path, since otherwise we could extend it. Since v_1 has at least k neighbors, the set $\{v_2, \ldots, v_l\}$ must contain at least k elements. Hence $l \geq k+1$, so the path has length at least k. □

Note that in general this bound cannot be improved, because the complete graph K_{k+1} has minimum degree k, but its longest path has length k. In problem set 1, we will prove an analogous statement for cycles.

The next statement shows that every graph with sufficiently many edges must contain a subgraph with large minimum degree. We give an inductive proof, although it could also be proved using an algorithmic argument.

Lemma 2.8. *Let G be graph on n vertices with at least $(k-1)n$ edges. Then G contains a subgraph H with $\delta(H) \geq k$.*

Proof. We proceed by induction on n. Note that $n \leq 2(k-1)$ is impossible because such a graph has at most $n(n-1)/2 < n(k-1)$ edges. Also, for $n = 2k-1$ the only graph with $n(k-1)$ edges is $K_n = K_{2k-1}$, which has minimum degree $2k-2$, so we can take $H = G$.

Now suppose $n > 2k-1$. If $\delta(G) \geq k$, then we can take $H = G$. Otherwise, there is a vertex v of degree $d(v) \leq k-1$. Let $G' = G - v$ be the subgraph we obtain from G by deleting v and all the edges touching it. Then G' has $n-1$ vertices and at least $n(k-1) - (k-1) = (n-1)(k-1)$ edges, so by induction, there is a subgraph $H \subseteq G' \subseteq G$ such that $\delta(H) \geq k$. □

We are done, as G contains a subgraph H with minimum degree at least k by Lemma 2.8, and H contains a path of length k by Lemma 2.7. But then G contains a path of length k as well. □

The following lemma can be helpful when trying to prove certain statements for general graphs that are easier to prove for bipartite graphs. The lemma says that you don't have to remove more than half the edges of a graph to make it bipartite. The proof is an example of an *algorithmic proof*, where we prove the existence of an object by giving an algorithm that constructs such an object.

Proposition 2.9. *Any graph G contains a bipartite subgraph H with $|E(H)| \geq |E(G)|/2$.*

Proof. We prove the stronger claim that G has a bipartite subgraph H with $V(H) = V(G)$ and $d_H(v) \geq d_G(v)/2$ for all $v \in V(G)$. Starting with an arbitrary partition $V(G) = X \cup Y$ (which need not be a bipartition for G), we apply the following procedure. We refer to X and Y as "parts". For any $v \in V(G)$, we see if it has more edges to X or to Y; if it has more edges that connect it to the part it is in than it has edges to the other part, then we move it to the other part. We repeat this until there are no more vertices v that should be moved.

There are at most $|V(G)|$ consecutive steps in which no vertex is moved, since if none of the vertices can be moved, then we are done. When we move a vertex from one part to the other, we increase the number of edges between X and Y (note that a vertex may move back and forth between X and Y, but still the total number of edges between X and Y increases in every step). It follows that this procedure terminates, since there are only finitely many edges in the graph. When it has terminated, every vertex in X has at least half its edges going to Y, and similarly every vertex in Y has at least half its edges going to X. Thus the graph H with $V(H) = V(G)$ and $E(H) = \{xy \in E(G) : x \in X, y \in Y\}$ has the claimed property that $d_H(v) \geq d_G(v)/2$ for all $v \in V(G)$. □

The following is an easy but important fact. Its proof is extremal.

Proposition 2.10. *Every u-v walk W contains a u-v path.*

Proof. Let $v_1 v_2 \dots v_k$ be a shortest u-v walk in W (more precisely, in the graph defined by the edges of W), so $u = v_1$ and $v = v_k$. We claim that this walk is in fact a path. Indeed, if $v_i = v_j$ for some $i < j$, then $v_1 v_2 \dots v_i v_{j+1} \dots v_k$ is also a u-v walk, and it is shorter (has fewer edges), which is not possible. So the shortest walk has no repeated vertices, i.e., it is a path. □

This fact has the useful corollaries that we can replace paths with walks in some of our definitions:

- The distance $d(u, v)$ is equal to the length of the shortest u-v walk.
- A graph is connected if and only if every pair of vertices u, v is connected by a walk.

The latter connectivity property is sometimes easier to check. Also, it clearly implies that connectivity is an equivalence relation.

Our last basic result gives a connection between the number of edges and the number of connected components in a graph.

Proposition 2.11. *If a graph G has n vertices and k edges, then it has at least $n - k$ components.*

Proof. Let us start with the empty graph and add the edges of G to it one-by-one. At the beginning there are n vertices and no edges, so we have n components. Each added edge touches at most 2 of the components, and joins these components if they are different (an edge within a component does not affect any components). This means that adding an edge decreases the number of components by at most 1. Adding k edges therefore decreases the number of components by at most k, so after adding all k edges of G, we are left with at least $n - k$ of them. □

Corollary 2.12. *Every connected graph on n vertices has at least $n - 1$ edges.*

2 Trees

Definition. *A* tree *is a connected graph without cycles. A* forest *is a graph without cycles. In a tree or a forest, a vertex of degree one is called a* leaf.

Lemma 2.13. *Every tree with at least two vertices has a leaf.*

Proof. Take a longest path $v_0v_1 \dots v_k$ in the tree (so $k \geq 1$, since the tree has at least two vertices). A neighbor of v_0 cannot be outside the path, since then the path could be extended. But if v_0 were adjacent to v_i for some $i > 1$, then $v_0v_1 \cdots v_iv_0$ would be a cycle. So the only neighbor of v_0 is v_1, and v_0 is a leaf. The same argument shows that v_k is also a leaf. □

Theorem 2.14. *Any tree T on n vertices has $n - 1$ edges.*

Proof. We use induction on the number of vertices. If $n = 1$, then we have 0 edges. Otherwise, Proposition 2.13 gives a leaf v of T. Let $T' = T - v$ be the graph obtained by removing v and its only edge. Then T' is connected, since for any $x, y \in V(T')$ there is a path from x to y in T, and this path cannot pass through v, so it is also a path in T'. Since T has no cycles, neither does T', so T' is a tree, on $n - 1$ vertices. By induction T' has $n - 2$ edges, so, using $|E(T')| = |E(T)| - 1$, we see that T has $n - 1$ edges. □

Theorem 2.15. *A graph G is a tree if and only if for all $u, v \in V(G)$ there is a unique path from u to v.*

Proof. First suppose we have a graph G in which any two vertices are connected by a unique path. Then G is certainly connected. Moreover, if G contained a cycle $v_1 \dots v_k v_1$, then v_1v_k and $v_1v_2 \cdots v_k$ would be two distinct paths between v_1 and v_k. Hence G is a tree.

Suppose G is a tree and $u, v \in V(G)$. Since G is connected, there is at least one path from u to v. Suppose there are two distinct paths P, P' from u to v. If these paths only intersect at u and v, we can immediately combine them into a cycle, but in general the paths could intersect in a complicated way, so we have to be careful. The paths P and P' could start out from u being the same; let x be the first vertex that they leave at different edges (so their next vertices are different). Let y be the first vertex of P after x that is also contained in P'. Then there is a cycle in G that goes along P from x to y, and then back along P' from y to x. This is a contradiction, so there is a unique path from u to v in G. □

Lecture 3

BFS. Euler tours.

---

1 Breadth-first search

Definition. *A* spanning tree *of a graph G is a subgraph $T \subseteq G$, which is a tree with $V(T) = V(G)$.*

Our aim is to show that

Theorem 3.16. *Every connected graph has a spanning tree.*

We start with the *greedy algorithm*. For this, we assume that the edges are indexed by $e_1, \ldots, e_m$.

Algorithm 1. *Unweighted Greedy*

Initialize the edge set $F = \varnothing$
for $i = 1, \ldots, m$
  if $(V, F + e_i)$ *does not contain a cycle*
    $F = F + e_i$

Theorem 3.17. *Let F be constructed by the procedure in Algorithm 2. If G is connected, then (V, F) is a spanning tree.*

Proof. Clearly, (V, F) has no cycles. We only have to show that (V, F) is connected. Let $x = v_1, \ldots, v_l = y$ be a path connecting x and in y in G. And suppose that x and y are not connected in (V, F), then let $k \geq 2$ be minimal such that v_1 and v_k are not connected in (V, F). The edge $v_{k-1}v_k$ has not been inserted by the greedy algorithm. This means that there is a path from v_{k-1} to v_k in (V, F). This means that v_k can be reached from v_1 in (V, F), a contradiction. □

In the *minimum weight spanning tree problem* we are given a connected graph and a weight function $w : E \to \mathbb{R}$. The goal is to find a spanning tree $T = (V, F)$ such that $w(F) \leq w(F')$ for each spanning tree (V, F') of G. This is computed by the weighted greedy algorithm.

Algorithm 2. *Weighted Greedy*

Sort the edges in increasing order according to weight:
$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$
Initialize the edge set $F = \varnothing$
for $i = 1, \ldots, m$
  if $(V, F + e_i)$ *does not contain a cycle*
    $F = F + e_i$

Theorem 3.18. *The weighted greedy algorithm finds a minimum weight spanning tree.*

Proof. Theorem 3.18 shows that T is a tree. Let $|V| = n$ and suppose that the edges of T are $e_{\pi_1}, \ldots, e_{\pi_{n-1}}$ and that they are picked in that order. Let T_{OPT} be a minimum weight spanning tree for which the smallest k such that e_{π_k} is not an edge of T_{OPT} is maximal.

Inserting e_{π_k} into T_{OPT} results in a cycle that does not only contain edges of T. Let e be an edge on this cycle that does not belong to T. This edge must be after e_k in the ordering of the edges according to their weight, since the edges $e_{\pi_1}, \ldots, e_{\pi_{k-1}}$ also belong to T_{OPT} and since any other edge before e_{π_k} creates a cycle in $e_{\pi_1}, \ldots, e_{\pi_{k-1}}$.

This means that $w(e) \leq w(e_{\pi_k})$. Thus $T_{OPT} + e_{\pi_k} - e$ is an optimal spanning tree which contradicts the maximality of k. □

We next describe a spanning tree that is called *breadth-first-search* tree. To this end, let $s \in V$ be a starting vertex and let $V_i = \{v \in V : d(s, v) = i\}$ be the vertices that have distance i from s. Let k be the largest distance of a vertex from s that is connected to s. We can build a tree as follows.

Initialize the edge set $F = \varnothing$.
for $i = 1, \ldots, k$
  for each $v \in V_i$:
    choose $u \in V_{i-1}$ with $uv \in E$.
    $F = F + uv$

Let $V' = \cup_{i=0}^{k} V_i$. The graph $T_{BFS} = (V', F)$ is connected, since one can reach s from any vertex by following their edge "downwards". Also $|F| = |V'| - 1$ which implies that T_{BFS} is a tree. It is a *breadth-first-search tree*. The unique path from s to any other vertex in T_{BFS} is a shortest path from s to that vertex in G. It is computed by the following algorithm.

Algorithm 3 (Breadth-First-Search)**.**

Initialize: $F = \varnothing$, *queue* $Q = \{s\}$
Label each vertex $v \in V \setminus \{s\}$ unknown
Label s known
while Q *is not empty*
  $u = head(Q)$
  $dequeue(u)$
  for each $v \in N(u)$
    if v *is unknown*
      label v *as known*
      $F = F + uv$
      enqueue v

The algorithm computes a bfs-tree in a bottom up manner. The following are claims that are easily verified by induction.

Lemma 3.19. *For each* $i = 0, \ldots, k$, *there is a moment in time, when* Q *contains exactly* V_i, *all vertices in* $\cup_{j=0}^{i} V_j$ *are labeled as known and all other vertices are labeled as unknown. Furthermore* $(\cup_{j=0}^{i} V_j, F)$ *is a tree in which the unique path from* s *to any other vertex is a shortest path in* G.

Definition. *The* diameter *of a graph* G *is the largest distance among any pairs of vertices:* $diam(G) = \max_{x,y \in V(G)} d(x, y)$. *If* G *is disconnected, then* $diam(G) = \infty$.

BFS has a number of applications in providing fairly (or very) efficient algorithms for solving certain tasks on graphs. For example:

- *Find a shortest path from xu to v in G:* Run the algorithm with root u to get a tree T. The unique path from u to v in T (which exists by Lemma 2.15) is a shortest path.
- *Find the connected components of G:* Run the algorithm with some root r. The vertices explored by BFS are exactly the component of r. If there is an unexplored vertex r', run BFS again from r' as a root. Repeat until all vertices are visited. This actually gives a spanning forest of G.
- *Compute diam(G):* For each pair of vertices, we can find a shortest path and thus the distance. Do this for all pairs and take the largest distance.
- *Find a shortest cycle in G:* For every edge uv, find a shortest path between u and v in $G - uv$ (if it exists), then combine this path with uv to get a cycle. (This will be a shortest cycle through uv.) Compare all these cycles to find the shortest.
- *Determine if G is bipartite:* Determine the connected components of G. In every component H, select a root r, and partition the vertices into $X = \{x \in V(H) : d(r, x) \text{ is even}\}$ and $Y = \{y \in V(H) : d(r, y) \text{ is odd}\}$. Then H is bipartite if and only if X and Y have no internal edges (see the proof of Theorem 1.2), and G is bipartite if and only if every component is bipartite.

Not all of these algorithms are the most efficient, but they are already much better than brute force approaches that go over all possible answers. There are all kinds of algorithms that do these tasks faster, but in this course, we don't care too much about efficiency, and we focus on the graph-theoretical aspects (in particular, proving that the algorithms work).

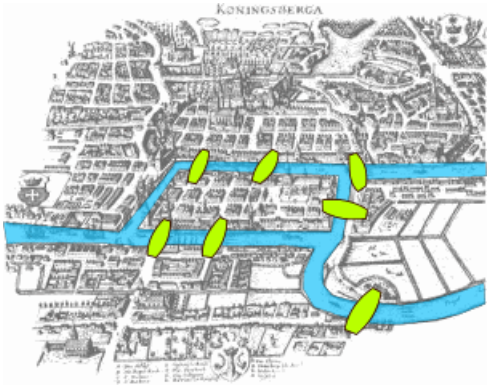
We should emphasize, though, that for finding shortest paths between two vertices, BFS is the best general algorithm. For example, *Dijkstra's algorithm*, a variant of BFS that works for graphs with positive edge weights (representing, e.g, the lengths of streets) is directly used by routing softwares.

2 Euler tours

Definition. *A* trail *is a walk with no repeated edges. A* tour *is a closed trail (i.e., one that starts and ends at the same vertex).*

Definition. *An* Euler (or Eulerian) trail *in a (multi)graph $G = (V, E)$ is a trail in G passing through every edge (exactly once). An* Euler tour *is a tour in G passing through every edge.*

This notion originates from the "seven bridges of Königsberg" problem – the oldest problem in graph theory, originally solved by Euler in 1736 – that asked if it was possible to walk through all the seven bridges of Königsberg in one go without crossing any of them twice.



This question can be turned into a graph problem asking for an Euler trail. Euler solved the problem by noticing that the existence of Euler trails is closely related to the degree parities.

Theorem 3.20. *A connected (multi)graph has an Eulerian tour if and only if each vertex has even degree.*

The proof of this theorem is based on the following simple lemma.

Lemma 3.21. *In a graph where all vertices have even degree, every maximal trail is a closed trail.*

Proof. Let T be a maximal trail. If T is not closed, then T has an odd number of edges incident to the final vertex v. However, as v has even degree, there is an edge touching v that is not contained in T. This edge can be used to extend T to a longer trail, contradicting the maximality of T. □

Proof of Theorem 3.20. To see that the condition is necessary, suppose G has an Eulerian tour C. If a vertex v was visited k times in the tour C, then each visit used 2 edges incident to v (one incoming edge and one outgoing edge). Thus, $d(v) = 2k$, which is even.

To see that the condition is sufficient, let G be a connected graph with even degrees. Let $T = e_1e_2 \dots e_\ell$ (where $e_i = (v_{i-1}, v_i)$) be a longest trail in G. Then it is maximal, of course. According to the Lemma, T is closed, i.e., $v_0 = v_\ell$. G is connected, so if T does not include all the edges of G then there is an edge e outside of T that touches it, i.e., $e = uv_i$ for some vertex v_i in T. Since T is closed, we can start walking through it at any vertex. But if we start at v_i then we can append the edge e at the end: $T' = e_{i+1} \dots e_\ell e_1 e_2 \dots e_i e$ is a trail in G which is longer than T, contradicting the fact that T is a longest trail in G. Thus, T must include all the edges of G and so it is an Eulerian tour. □

Corollary 3.22. *A connected multigraph G has an Euler trail if and only if it has either* 0 *or* 2 *vertices of odd degree.*

Proof. Suppose T is an Euler trail from vertex u to vertex v. If $u = v$ then T is an Eulerian tour and so by Theorem 3.20, it follows that all the vertices in G have even degree. If $u \neq v$ then let us add a new edge $e = uv$ to G. In this new multigraph $G \cup \{e\}$, $T \cup \{e\}$ is an Euler tour. By Theorem 3.20 we see that all the degrees in $G \cup \{e\}$ are even. This means that in the original multigraph G, the vertices u, v are the only ones that have odd degree.

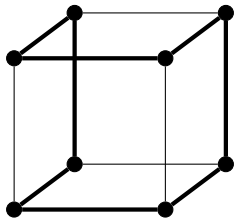
Now we prove the other direction of the corollary. If G has no vertices of odd degree then by Theorem 3.20 it contains an Eulerian tour which is also an Eulerian trail. Suppose now that G has 2 vertices u, v of odd degree. Then add a new edge e to G. Now all vertices of the resulting multigraph $G \cup \{e\}$ have even degree, so, by Theorem 3.20, it has an Eulerian tour C. Removing the edge e from C gives an Eulerian trail of G from u to v. □

Lecture 4

Hamilton cycles.

Euler trails are walks that use each edge exactly once. But what if we want to use each *vertex* exactly once?

Definition. *A* Hamilton (or Hamiltonian) cycle *in a graph G is a cycle that contains all vertices of G. A* Hamilton path *in a graph G is a path that contains all vertices of G. A graph G is* Hamiltonian *if it contains a Hamilton cycle.*



Although the proof of Theorem 3.20 is not really described in an algorithmic way, it can actually be turned into an efficient algorithm. However, no good algorithm is known that would find a Hamilton cycle in a graph: nothing that would be much better a brute-force algorithm that tries every possible way to visit all vertices in the graph. Deciding if G is Hamiltonian is a so-called *NP-hard* (actually, NP-complete) problem. Loosely speaking, this means that an efficient algorithm for this problem could be transformed into an efficient algorithm for many other problems that are also considered algorithmically difficult.

Definition. *The* girth *of a graph G is the length of the shortest cycle contained in G. The* circumference *is the length of the longest cycle contained in G.*

For example, the complete graph K_n for $n \geq 3$ has girth 3 and circumference n.

We have seen that computing the girth of a graph can be done fairly efficiently using BFS. But finding the circumference is even more difficult than deciding if a graph is Hamiltonian (because a graph has a Hamilton cycle if and only if its circumference is n).

Another related problem is to find a shortest Hamilton cycle in a graph with weighted edges; this is called the *travelling salesman problem (TSP)* and is one of the most famous computationally hard problems, with many real-life applications.

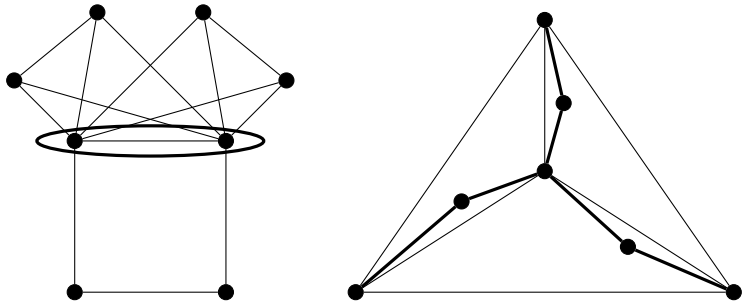
Although we have no general algorithm or recipe for finding Hamilton cycles, we can still prove some theorems that are useful in certain situations. Our first example is a necessary condition for Hamiltonicity.

Given a graph G and a set $S \subset V(G)$ of vertices, we write $G - S$ for the graph obtained by removing the vertices of S from G, along with all the edges touching the vertices in S.

Lemma 4.23. *If G has a Hamilton cycle, then for all $S \subset V(G)$, $G - S$ has at most $|S|$ connected components.*

Proof. The Hamilton cycle must visit all the components of $G - S$ (viewed as subgraphs of G), and to get from one component to another the cycle must pass through a vertex of S. Thus every component is connected to S by two edges of the cycle (and possibly by other edges not in the cycle). Since every vertex is incident to two edges of the cycle, we have that twice the number of components is at most twice the number of vertices of S. □

This lemma can be useful to show that a graph does not have a Hamilton cycle. For example, if in the left-hand graph G below, S consists of the middle two vertices, then $G - S$ has three connected components, so by Lemma 4.23 the graph has no Hamilton cycle.



On the other hand, it is important to emphasize that the condition in Lemma 4.23 is not sufficient. For example, one can check that the right-hand graph H satisfies the condition that for all $S \subset V(H)$, $H - S$ has at most $|S|$ components, yet it has no Hamilton cycle. To see the latter, observe that for each vertex of degree 2, both incident edges would need to be in the cycle; but then the middle vertex would be incident to at least three edges of the cycle, which is impossible.

1 Sufficient conditions for Hamiltonicity

First, we will show that a graph with many edges must have a Hamilton cycle. Unfortunately, there is no chance to get a very good bound on the number of edges sufficient to guarantee a Hamilton cycle, because there are "almost complete" graphs that are not Hamiltonian.

Take for instance the graph G consisting of K_{n-1} and an additional vertex connected to a single vertex of K_{n-1}. This graph has $\binom{n-1}{2} + 1$ edges (so only misses $n-2$ edges), but it is not Hamiltonian, because it has a vertex of degree 1. Our next result shows that any graph with more edges contains a Hamilton cycle.

Theorem 4.24. *If G is an n-vertex graph with $|E(G)| > \binom{n-1}{2} + 1$ edges, then G contains a Hamilton cycle.*

Proof. We apply induction on n. The statement is clearly true for $n = 1, 2, 3$, so we assume $n > 3$.

We claim that G has a vertex v of degree at least $n - 2$. Indeed, otherwise every vertex has degree at most $n - 3$, so

$$|E(G)| = \frac{\sum_v d(v)}{2} \leq \frac{n(n-3)}{2} < \frac{(n-1)(n-2)}{2} = \binom{n-1}{2},$$

contradicting our assumption.

Now let $G' = G - v$, so G' has $n - 1$ vertices. We distinguish the two cases $d(v) = n - 2$ and $d(v) = n - 1$.

Suppose $d(v) = n - 2$. Then

$$|E(G')| = |E(G)| - (n-2) > \binom{n-1}{2} + 1 - (n-2) = \binom{n-2}{2} + 1,$$

so G' has a Hamilton cycle C by induction. Since $d(v) = n-2$ and $n > 3$, v must be adjacent to two consecutive vertices of this cycle, x and y. Then we can remove the edge xy from C and replace it by xv and vy to get a Hamilton cycle in G.

Now suppose $d(v) = n-1$. In this case we only have $|E(G')| > \binom{n-2}{2}$, so we cannot apply induction right away. If G' is complete, then G' has a Hamilton cycle, and we can add v as in the previous case (in fact, as $d(v) = n-1$, v is now adjacent to every vertex of the cycle).

Otherwise, there is an edge xy missing from G'. Let us look at the graph $G' + xy$ that we get by adding xy to G'. Now this graph has more than $\binom{n-2}{2} + 1$ edges, so we can apply induction to find a Hamilton cycle C in it. If C does not contain xy, then we can again add v as in the previous case. If C does contain xy, then replacing xy with the path xvy in C gives a Hamilton cycle in G. □

As mentioned before, the condition in the theorem above is somewhat weak in the sense that many graphs that have a Hamilton cycle do not satisfy the condition. The following sufficient condition does better by looking at the minimum degree instead of the total number of edges. Note that we have previously seen that every graph G has a cycle of length at least $\delta(G)+1$. The following theorem says that something much stronger is true when the minimum degree is at least $|V(G)|/2$.

Theorem 4.25 (Dirac)**.** *Let G be a graph on $n \geq 3$ vertices. If $\delta(G) \geq \frac{n}{2}$, then G contains a Hamilton cycle.*

Proof. First observe that G must be connected, because each component contains at least $\delta(G) + 1 > n/2$ vertices, so G cannot have more than one components.

Take a longest path $P = v_1 v_2 \dots v_k$ in G. By maximality, all neighbors of v_1 and v_k are in the path. Let us say that an edge $v_i v_{i+1}$ is type-1 if $v_{i+1} \in N(v_1)$, and let us say that it is type-2 if $v_i \in N(v_k)$. As $\delta(G) \geq n/2$, we have at least $n/2$ type-1 and $n/2$ type-2 edges in P. But P has at most $n-1$ edges, so some edge $v_j v_{j+1}$ is both type-1 and type-2, i.e., $v_1 v_{j+1}$ and $v_j v_k$ are edges of G. Then $C = P - v_j v_{j+1} + v_1 v_{j+1} + v_j v_k = v_j \dots v_1 v_{j+1} \dots v_k v_i$ is a cycle.

In fact, C is a Hamilton cycle. Indeed, suppose not all vertices are contained in C. Since G is connected, there must be an edge uv_i where $u \notin C$. Then there is a path that goes from u to v_i and then all around the cycle C to a neighbor of v_i. This path contains $k+1$ vertices, contradicting the maximality of P. □

This theorem is again best possible, in the sense that a weaker bound on the minimum degree would not imply a Hamilton cycle. Take for instance the graph G consisting of two copies of K_k sharing a single vertex. This graph has $n = 2k-1$ vertices and minimum degree $\delta(G) = k - 1 = \frac{n-1}{2}$, but no Hamilton cycle, because deleting the shared vertex creates two components, contradicting Lemma 4.23.

It is easy to check that the proof of Dirac's theorem works for the following strengthening, as well:

Theorem 4.26 (Ore)**.** *Let G be a graph on $n \geq 3$ vertices. If $d(u) + d(v) \geq n$ for any non-adjacent vertices u and v, then G contains a Hamilton cycle.*

Using this theorem, one can also obtain a short proof of Theorem 4.24 (see problem set), so one can think of Ore's theorem as a common generalization of Theorems 4.24 and 4.25.

Definition. *A* tournament *is a directed graph that has exactly one (oriented) edge between any two vertices.*

For example, if we take a complete graph and give each edge an orientation then we get a tournament. We use $u \to v$ to denote that there is an edge uv directed from u to v. What is the longest directed path in such a graph?

Theorem 4.27. *Every tournament contains a directed Hamilton path.*

Proof. We do induction on the number of vertices n. For $n = 2$, there is nothing to prove, since one can take the only edge of the tournament as the Hamilton path. Now suppose $n > 2$, and let v be a vertex of T. By induction, $T - v$ has a Hamilton path $P = v_1 v_2 \dots v_n$. If $v \to v_1$ or $v_n \to v$, then $v v_1 v_2 \dots v_n$ or $v_1 v_2 \dots v_n v$ is a Hamilton path in T. So assume $v_1 \to v$ and $v \to v_n$. But then there must be a vertex v_i with $1 \leq i \leq n-1$, such that $v_i \to v$ and $v \to v_{i+1}$, so $v_1 v_2 \dots v_i v v_{i+1} v_{i+2} \dots v_n$ is a Hamilton path in T. □

Of course, not every tournament contains a directed Hamilton cycle. In fact, there is a tournament that does not contain any cycle, whatsoever: the so-called transitive tournament on vertex set $\{1, \dots, n\}$ where each edge is oriented from the smaller endpoint to the larger one.

Lecture 5

Matchings.

Recall that a *matching* M is a set of vertex-disjoint edges. A *perfect matching* M in a graph G is a matching that touches every vertex of G. In other words, it is a 1-regular subgraph of G (recall that a graph is k-regular if all degrees equal k).

1 Stable matchings

Suppose that you have n students that want to do their internships in n companies. Each have sent their applications to all the companies. Each student and each company has their list of preferences, and we want to pair up the students with the companies so that this assignment is *stable* in the following sense: there is no student and company that would both prefer to work with each other rather than their assigned pairs.

More formally, consider a bipartite graph G with parts A, B, where $|A| = |B| = n$, and in which each vertex has a (strict) order of preferences for the vertices of the other part. We say that a perfect matching is *stable*, if there is no pair $a \in A, b \in B$, such that both of them would prefer the other to the vertex they are currently matched to.

Below we present an algorithm of Gale and Shapley, which allows to construct such a stable matching.

The Gale-Shapley Algorithm to find a stable matching M in a complete bipartite graph $G = (A \cup B, E)$ with $|A| = |B|$

(1) Set $M = \emptyset$;

(2) Iterate:

  (a) Take an unmatched vertex $a \in A$ and let $b \in B$ be the vertex that a prefers among the ones a has not tried yet.

  (b) a "proposes" to b: If b is unmatched or b is matched to a', but prefers a over a', then "accept" a and "reject" a': put $M := M - a'b + ab$. Otherwise, "reject": leave M unchanged;

  (c) If there is no more unmatched vertices in A that have someone left on the list, then go to (3);

(3) Return M.

Proposition 5.28. *The matching M that the algorithm outputs is stable.*

Proof. First we show that M is perfect. Indeed, if there is a pair of vertices $a \in A, b \in B$, such that both are not in the matching, then a must have proposed to b at some point. However, if a vertex $b \in B$ is in M at some step of the algorithm, then it stays in M.

Next, we show that the matching is stable. Assume that $ab \notin M$. Upon completion of the algorithm, it is not possible for both a and b to prefer each other over their current match. If a prefers b to its match, then a must have proposed to b before its current match. If b accepted its proposal, but is matched to another vertex at the end, then b prefers the current match of b over a. If b rejected the proposal of a, then b was already matched to a vertex that is better for b. □

2 Maximum matchings

The previous problem was about perfect matchings in a complete bipartite graph. But not every graph has a perfect matching. So how can we find a *largest* or *maximum* matching? (Not to be confused with a *maximal* matching, which just means that it cannot be extended, i.e., no larger matching contains it as a subgraph.) How can we even tell if a graph has a perfect matching? How can we check if a given matching is maximum?

First, note that a maximal matching need not be maximum. Take for instance a path with three edges, and the matching consisting of the middle edge. Similarly, a greedy approach that keeps adding edges without removing any, like we used for spanning trees, would probably not lead to a maximum matching. Instead, an algorithm to find maximum matchings will need some kind of backtracking, where we throw away some edges that we previously selected. The following notion lets us do that in a smart way.

Definition. *Given a matching M in a bipartite graph G, a path is* alternating *if among any two consecutive edges on the path, exactly one is in M. An alternating path with at least one edge is* augmenting *if its first and last vertices are not covered by M.*

Note that an augmenting path always has odd length.

Lemma 5.29. *A matching M is maximum if and only if there is no augmenting path for M.*

Proof. We prove that M is not maximum if and only if there is an augmenting path for M. First suppose that there is an augmenting path $P = v_1v_2 \dots v_{2k}$ for the matching M. So $v_2v_3, \dots, v_{2i}v_{2i+1}, \dots, v_{2k-2}v_{2k-1} \in M$. Then we can get a larger matching by removing these edges from M and replacing them by $v_1v_2, \dots, v_{2i-1}v_{2i}, \dots, v_{2k-1}v_{2k}$. In other words, we replace M by $M' = M \triangle E(P)$.[1] Then M' is a matching since v_1 and v_k were unmatched by M, and we have $|M'| = |M| + 1$, so M is not maximum.

Now suppose M is not maximum, i.e., there is a matching M' with $|M'| > |M|$. Let H be the graph with $V(H) = V(G)$ and $E(H) = M \triangle M'$. Every vertex of H has degree 0, 1 or 2, so each component of H is either a cycle, a path, or an isolated vertices. A cycle in H has the same number of edges from M and M', so $|M'| > |M|$ implies that there is a path P in D with more edges from M' than from M. Then P must be an augmenting path for M. □

This result shows that to improve our matching, we just need to find an augmenting path. If no such path exists, then the matching is maximum. But it is not so easy to check if an augmenting path exists; a naive algorithm would take exponentially many steps. An efficient algorithm for a maximum matching was found by Edmonds in 1965, but it is beyond the scope of this course. However, the problem becomes much simpler in bipartite graphs.

3 Matchings in bipartite graphs

Let $G = (A \cup B, E)$ be a bipartite graph, and M be a matching in it. Note that the end vertices of every augmenting path P are in different parts (because P has odd length). So if there is an augmenting path in G, then it starts in $S = A - V(M)$ and ends in $T = B - V(M)$. Crucially, such a path starting in S will always use an edge *not in* M to jump to B, and an edge *in* M to jump back to A.

So let us define D_M to be the digraph obtained from G by orienting edges not in M from A to B, and orienting edges in M from B to A. The observations above show that

[1] We write $S \triangle T$ for the *symmetric difference* of two sets, i.e., $S \triangle T = (S \setminus T) \cup (T \setminus S)$.

augmenting paths in G correspond to directed S-T paths in D_M. (An S-T path is a path that starts in S and ends in T.)

The point is, finding S-T paths in a directed graph is easy (e.g., using breadth-first search). So together with Lemma 5.29, we arrive at the following algorithm for finding a maximum matching in a bipartite graph.

Augmenting Path Algorithm
to find a maximum matching in a bipartite $G = (A \cup B, E)$

(1) Set $M = \emptyset$;

(2) Iterate :

  (a) Set $S = A - V(M)$, $T = B - V(M)$;

  (b) Find any directed S-T path P in D_M using BFS; if none exist, go to (3);

  (c) Replace M with $M := M \triangle P$;

(3) Return M.

4 Perfect matchings

The next question we study is: when does a bipartite graph have a perfect matching? Apart from the trivial condition $|A| = |B|$, it is easy to see that we need every set $X \subseteq A$ to have at least $|X|$ neighbors. This condition turns out to be enough, as shown by the next theorem. We state it slightly more generally, using the following definitions.

A matching M *matches* or *covers* a vertex set X if every vertex of X is contained in an edge of M. For a set $X \subset V(G)$, we define the *neighborhood* of X in G as $N(X) = \{v \in V(G) \setminus X$ such that $uv \in E(G)$ for some $u \in X\}$.

Theorem 5.30 (Hall, 1935)**.** *Let $G = (A \cup B, E)$ be a bipartite graph. Then G has a matching covering A if and only if for all $X \subseteq A$ we have $|N(X)| \geq |X|$.*

Proof. For one direction, note that if G has a matching M that matches A, then the vertices of any $X \subset A$ are matched by M to $|X|$ distinct neighbors in B, which implies $|N(X)| \geq |X|$.

For the other direction, suppose $|N(X)| \geq |X|$ for every $X \subseteq A$. Let M be a maximum matching, and let $S = A - V(M)$ and $T = B - V(M)$ be the unmatched vertices in A and B. By Lemma 5.29 and the discussion above, there is no S-T path in D_M. So if X_B denotes the set of vertices in B that can be reached from S via a directed path in D_M, then X_B is disjoint from T. In other words, all vertices in X_B are matched by M.

Let $X_A \subseteq A$ be the set of vertices matched to X_B by M (here X_A is clearly disjoint from S). As the matching edges are directed from B to A, the vertices in X_A are also reachable from S in D_M. To finish the proof, we just need to check that $N(X_A \cup S) \subseteq X_B$, because this and our assumption with $X = X_A \cup S$ would imply

$$|X_B| \geq |N(X_A \cup S)| \geq |X_A \cup S| = |X_A| + |S| = |X_B| + |S|,$$

and hence $S = \emptyset$, which is what we want to prove.

To see $N(X_A \cup S) \subseteq X_B$, note that all vertices in $X_A \cup S$ are reachable from S in D_M. But then if there was a vertex $b \in B - X_B$ adjacent to some $a \in X_A \cup S$, then ab is not a matching edge, so it is directed from a to b in D_M. Consequently, $b \notin X_B$ is reachable from S, which is a contradiction. □

If $|A| = |B|$, then of course every matching covering A covers B, as well. So Hall's theorem implies that a bipartite graph $G = (A \cup B, E)$ has a perfect matching if and only if $|A| = |B|$ and $|N(X)| \geq |X|$ for every $X \subseteq A$.

The property "$|N(X)| \geq |X|$ for every $X \subseteq A$" is often referred to as *Hall's condition*. Testing this property algorithmically is slow, but it turns out to be a convenient thing to check in many theoretical applications.

Corollary 5.31. *Let $k \geq 1$ be a integer. Every k-regular bipartite graph has a perfect matching.*

Proof. Let $G = (A \cup B, E)$ be a k-regular bipartite graph. Since every edge touches exactly one vertex of A, and every vertex of A touches exactly k edges, the number of edges in G is exactly $k|A|$. By a similar argument, the number of edges is exactly $k|B|$, so $k \geq 1$ implies $|A| = |B|$.

As discussed above, Hall's condition would now guarantee a perfect matching. To check it, let $X \subseteq A$ be a vertex set. We will double-count the edges touching X. On the one hand, there are exactly $k|X|$ such edges. On the other hand, every such edge has its other endpoint in $N(X)$, so the number of such edges is at most $k|N(X)|$. Hence $k|X| \leq k|N(X)|$, which again implies $|X| \leq |N(X)|$. □

Lecture 6

Kőnig's theorem. Flows.

---

1 Kőnig's Theorem

Definition. *Given a graph G, a* vertex cover *for G is a set $C \subset V(G)$ such that every edge of G is incident with a vertex in C.*

The maximum size of a matching in G is denoted by $\nu(G)$, and the minimum size of a vertex cover in G is denoted by $\tau(G)$. (Note that the "size" of a matching is the number of edges in it, while the "size" of a vertex cover is the number of vertices in it.)

Theorem 6.32 (Kőnig, 1931)**.** *Let $G = (A \cup B, E)$ be a bipartite graph. Then $\nu(G) = \tau(G)$.*

Proof. It is clear that $\nu(G) \leq \tau(G)$, because a vertex cover has to cover every edge of the matching with one of the endpoints of the edge. So to cover a matching of size $\nu(G)$, we need at least this many vertices.

Now we show $\nu(G) \geq \tau(G)$. Let M be a maximum matching of G, and as before, let $S = A - V(M)$ and $T = B - V(M)$ be the set of unmatched vertices. As observed last time, if M is maximum, there is no S-T path in D_M.

Let X_B be the set of vertices in B that can be reached from S via a (directed) path in D_M, and let Y_A be the set of vertices in A that cannot be reached from S in D_M. Clearly, X_B is disjoint from T and Y_A is disjoint from A. Note also, that either both ends of an edge in M can be reached from S, or neither of them can be. In the former case, the edge has an end vertex in X_B, in the latter case it has an endpoint in Y_A. Consequently, $|X_B \cup Y_A| = |M|$

We will now show that $X_B \cup Y_A$ is a vertex cover. Suppose not, then there is an edge uv with $u \in A$ and $v \in B$ not covered by this set (i.e., $u \notin Y_A$ and $v \notin X_B$). But uv is not a matching edge, so in D_M, it is directed from u to v. But then $u \in A - Y_A$ can be reached from S in D_M (by definition), so v should also be reachable via the edge uv. This contradicts $v \notin X_B$.

So $X_B \cup Y_A$ is a vertex cover of size $\nu(G)$, implying $\nu(G) \geq \tau(G)$. □

2 Flows

Definition. *A* network *is a directed graph $G = (V, E)$ with two special vertices, the* source *$s \in V$ and the* sink *$t \in V$, together with a non-negative capacity function $c : E \to \mathbb{R}$.*

Definition. *A* flow *in a network $G = (V, E)$ is a function $f : V^2 \to \mathbb{R}$ such that*

1. *(flow conservation) for every $v \in V - \{s, t\}$,*

$$\sum_{w \in V} f(v, w) = \sum_{w \in V} f(w, v)$$

2. *(capacity constraint) $0 \leq f(u, v) \leq c(u, v)$ for every $\overrightarrow{uv} \in E$*

3. *$f(u, v) = 0$ if $\overrightarrow{uv} \notin E$.*

The *value* $|f|$ of a flow is defined as $\sum_{w\in V} f(s,w) - f(w,s)$. Our main question of study is: what is the maximum value of a flow in a network G?

A *cut* in a network G is a partition (S,T) of the vertices (so $S\cap T=\emptyset$, $S\cup T=V$) such that $s\in S$ and $t\in T$. For two vertex sets $X,Y\subseteq V$ we define

$$f(X,Y) := \sum_{x\in X, y\in Y} f(x,y) - f(y,x), \qquad c(X,Y) := \sum_{x\in X, y\in Y} c(x,y).$$

We call $c(S,T)$ the *capacity of the cut* (S,T). The following proposition is intuitively clear.

Proposition 6.33. *For every cut (S,T) we have $f(S,T)=|f|$.*

Proof. By definition, $|f| = \sum_{w\in V} f(s,w) - f(w,s)$. Also, $\sum_{w\in V} f(v,w) = \sum_{w\in V} f(w,v)$ for all $v\in S-s$, so

$$\begin{aligned} |f| &= \sum_{w\in V} f(s,w) - f(w,s) + \sum_{v\in S-s}\sum_{w\in V} f(v,w) - f(w,v) \\ &= \sum_{u,v\in S} f(u,v) - f(u,v) + \sum_{u\in S, v\in T} f(u,v) - f(v,u) = f(S,T) \end{aligned}$$

Here the second equality holds as we just rearranged the terms. □

3 Ford-Fulkerson algorithm

So how large can a flow be in a network? Proposition 6.33 implies that for any cut (S,T),

$$|f| = f(S,T) = \sum_{u\in S, v\in T} f(u,v) - f(v,u) \le \sum_{u\in S, v\in T} f(u,v) \le \sum_{u\in S, v\in T} c(u,v) = c(S,T).$$

In particular, the maximum value of a flow is bounded by the minimum capacity of a cut.

Theorem 6.34 (Ford–Fulkerson, 1956)**.** *In every network, the maximum value of a flow equals the minimum capacity of a cut.*

This theorem is also called the *max-flow min-cut* theorem. The proof of the lower bound is algorithmic, and uses the following definition.

Given a flow f, let D_f be the directed graph on $V(G)$ such that $\overrightarrow{uv}\in E(D_f)$ if $f(u,v)<c(u,v)$, or $f(v,u)>0$. Say that $\overrightarrow{uv}\in E(D_f)$ is *original* if $f(u,v)<c(u,v)$, and say that $\overrightarrow{uv}$ backwards if $f(v,u)>0$. The algorithm repeatedly finds a directed s-t path in D_f, and pushes some extra flow through it to increase the value of f.

Ford-Fulkerson Algorithm to find a flow f and a cut (S,T) such that $|f|=c(S,T)$.

(1) Set $f=0$ for all edges;

(2) While there is a directed s-t path $v_0v_1\dots v_l$ in D_f (so $v_0=s, v_l=t$):

  (a) Let ε be the minimum of $c(v_{i-1},v_i)-f(v_{i-1},v_i)$ for original edges and $f(v_i,v_{i-1})$ for backwards edges for $i=1,...,l$.

  (b) For $i=1,\dots,l$ set $f(v_{i-1},v_i) := f(v_{i-1},v_i)+\varepsilon$ if $\overrightarrow{v_{i-1}v_i}$ is original, and set $f(v_i,v_{i-1}) := f(v_i,v_{i-1})-\varepsilon$ if $\overrightarrow{v_{i-1}v_i}$ is backwards.

(3) Define S as the set of all vertices that are reachable by a directed path from s in D_f, and let $T=V\setminus S$.

(4) Return f and $(S,V-S)$.

We verify that the algorithm works correctly if c is integer valued. First, note that f satisfies the three conditions in the definition of a flow at each step. Second, since in the network all capacities are integral, we have that $\epsilon > 0$ is an integer at each step, and so $\epsilon \geq 1$. Therefore, since the flow must be finite (bounded by the capacity of any cut), the algorithm terminates in a finite number of steps. Finally, by the definition of D_f and the set S, we have $f(u,v) = c(u,v)$ for every edge $\overrightarrow{uv}$ with $u \in S, v \in T$, and $f(v,u) = 0$ for every edge $\overrightarrow{vu}$ with $u \in S, v \in T$. Hence

$$f(S,T) = \sum_{u \in S, v \in T} f(u,v) - f(v,u) = \sum_{u \in S, v \in T} c(u,v) = c(S,T).$$

Since ε is always an integer, we even get the following.

Fact 6.35. *If c is integral, then there is an integer maximum flow.*

Note that for the algorithm to stop, it was important that c takes integer values. The same argument works for rational c, as well (e.g., by multiplying all values with a number to make c integral), but the algorithm might not stop if c has non-rational values (see problem set). However, Theorem 6.34 holds for any non-negative real capacity function. This can be proved by approximating it with a rational function and taking the limit. Alternatively, Edmonds and Karp showed that the algorithm stops in a bounded number of steps no matter what the capacity function is, provided that it chooses the *shortest* s-t path of D_f in step 2.

Let us give a quick application of the Ford-Fulkerson theorem.

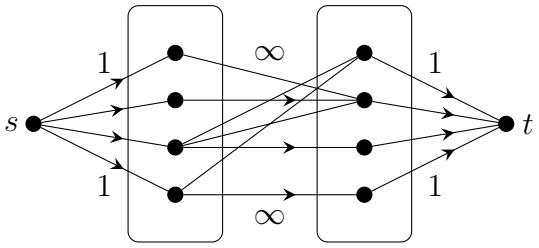
Application: *Proof of Theorem 6.32 via Ford-Fulkerson.*
Given $G = (A \cup B, E)$, let us create a network by adding a source s that sends an edge of capacity 1 to all vertices of A, and adding a sink t that receives an edge of capacity 1 from all vertices of B. We also orient the edges of G from A to B give them infinite capacities. (Think of ∞ as some large number bigger than $|A|$.)

Let f be an integer maximum flow. We claim that the G-edges with positive flow form a matching. Indeed, each such edge has $f(u,v) \geq 1$. If two such edges shared a vertex $u \in A$, then the inflow at u is at most 1 and the outflow is at least 2, which is not possible. Similarly, two such edges cannot meet in B, either. Hence $\nu(G) \geq |f|$.

Now let (S,T) be a minimum cut. Then $(S \cap B) \cup (T \cap A)$ is a vertex cover, because any uncovered edge (u,v) with $u \in S \cap A$ and $v \in T \cap B$ would contribute ∞ capacity to $c(S,T)$ (but $c(S,T) \leq C(s, V-s) \leq |A| < \infty$). Here every s-$(T \cap A)$ edge and every $(S \cap B)$-t edge contributes 1 to $c(S,T)$, so we get $\tau(G) \leq |S \cap B| + |T \cap A| \leq c(S,T)$. Hence

$$\tau(G) \leq c(S,T) = |f| \leq \nu(G),$$

which, together with the trivial $\nu(G) \leq \tau(G)$ finishes the proof. □



Lecture 7

Tutte polynomial.

For now, please consult https://piazza.com/class/k0o1x6oxrsa7cr?cid=17

Lecture 8

Coloring.

---

2 Vertex coloring

Recall the we call a graph (properly) *k-colorable* if its vertices can be colored with k colors in such a way that no two adjacent vertices get the same color, and that the chromatic number $\chi(G)$ is the minimum k such that G is k-colorable.

So far we have looked at the chromatic number of planar graphs, but it also makes sense to study the concept for general graphs. Actually, many real-life problems may be interpreted as graph coloring problems. Here is one example from scheduling:

Example. *The students at a certain university have annual examinations in all the courses they take. Naturally, examinations in different courses cannot be held concurrently if the courses have students in common. How can all the examinations be organized in as few parallel sessions as possible? To find a schedule, consider the graph G whose vertex set is the set of all courses, two courses being joined by an edge if they give rise to a conflict. Clearly, independent sets of G correspond to conflict-free groups of courses. Thus, the required minimum number of parallel sessions is the chromatic number of G.*

We first collect some easy facts about the chromatic number.

Definition. *The* complement *of a graph G is the graph $\overline{G}$ with vertex set $V(\overline{G}) = V(G)$ and edge set $E(\overline{G}) = \{xy : x, y \in V(G), xy \notin E(G)\}$.*
The clique number $\omega(G)$ *is the size of the largest complete subgraph (clique) of G. So $\omega(G) = \alpha(\overline{G})$.*

Fact 8.36. *1. $\chi(K_s) = s$*

2. G is bipartite if and only if $\chi(G) \leq 2$

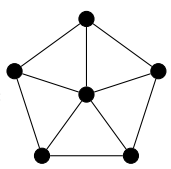
3. If H is a subgraph of G, then $\chi(H) \leq \chi(G)$

4. $\chi(G) \geq \omega(G)$ for every graph G.

Proof. 1. and 2. follow from the definition. For 3., we just need to notice that every proper coloring of G provides a coloring of H. 4. is then a combination of 1. and 3. □

So $\omega(G)$ is a trivial lower bound on the chromatic number, but it is not necessarily tight:

Example. *The following graph G satisfies $\chi(G) = 4$ and $\omega(G) = 3$.*

$G =$ 

In general, the chromatic number cannot be bounded by any function of the clique number. We show two different constructions of graphs with no triangle and arbitrarily large chromatic number.

Theorem 8.37. *For every positive integer k, there exists a graph G_k with $\omega(G) = 2$ and $\chi(G) \geq k$*

Proof. Construction 1. We prove this by induction on k. For $k = 2$, we can take G_2 be a single edge.

Now suppose that $k \geq 3$. Let $H_1, ..., H_{k-1}$ be $k-1$ disjoint copies of G_{k-1}. For every $v_1 \in V(H_1), \ldots, v_{k-1} \in V(H_{k-1})$, add a vertex $X(v_1, ..., v_{k-1})$ that is only connected to $v_1, ..., v_{k-1}$, and let G_k be the resulting graph.

Clearly, $\omega(G_k) = 2$. Indeed, H_i does not contain a triangle, so if there exists a triangle in G_k, then one of its vertices is $X(v_1, ..., v_{k-1})$ for some selection $v_i \in H_i$. But v_i and v_j are not neighbors for $i \neq j$, so $X(v_1, ..., v_{k-1})$ cannot be contained in a triangle either.

Now we show that $\chi(G_k) \geq k$. Suppose that there exists a proper coloring $c : V(G_k) \to \{1, ..., k-1\}$. As $\chi(H_i) \geq k-1$, every color must appear in H_i. But then let $v_i \in V(H_i)$ such that $c(v_i) = i$. Then every color was used in the neighborhood of $X(v_1, ..., v_{k-1})$, contradiction.

Construction 2. Let $n \geq 2^k$. The shift graph S_n is the graph with vertex set

$$\{(i, j) : 1 \leq i < j \leq n\},$$

where (i, j) and (k, l) are joined by an edge if $j = k$. Clearly, $\omega(S_n) = 2$.

Let $c : V(S_n) \to X$ be a proper coloring of S_n. For $i \in \{1, ..., n\}$, let

$$X_i = \{c((i, j)) : i \leq j \leq n\},$$

that is, X_i is the set of colors used for the vertices $(i, i+1), ..., (i, n)$. The main observation is that if $i \neq j$, then $X_i \neq X_j$. Otherwise, if $X_i = X_j$, then consider the color $c = c((i, j))$. By definition, $c \in X_i$, but then $c \in X_j$, which means that there exists $j < k \leq n$ such that $c((j, k)) = c$. But (i, j) and (j, k) are neighbors of the same color, contradiction.

As there are $2^{|X|}$ different subsets of X, and $X_1, ..., X_n$ are pairwise different, we must have $2^{|X|} \geq n$, which gives $|X| \geq \log_2 n \geq k$. □

A k-coloring can be thought of splitting the vertices into k independent sets. This readily implies the following lower bound on $\chi(G)$.

Lemma 8.38. *For every graph G, we have $\chi(G) \geq \frac{|V(G)|}{\alpha(G)}$.*

Proof. Given a coloring with $\chi(G)$ colors, the color classes (label them $V_1, \ldots, V_{\chi(G)}$) are independent sets, and thus have size at most $\alpha(G)$. Hence we have

$$|V(G)| = \sum_{i=1}^{\chi(G)} |V_i| \leq \sum_{i=1}^{\chi(G)} \alpha(G) = \chi(G)\alpha(G).$$

□

Claim 8.39. *For any two graphs G_1 and G_2 on the same vertex set V, $\chi(G_1 \cup G_2) \leq \chi(G_1)\chi(G_2)$.*

Proof. For both $i = 1, 2$, let $k_i = \chi(G_i)$ and take a k_i-coloring $c_i : V \to [k_i]$ of G_i. (Here $[n]$ denotes the set $\{1, \ldots, n\}$.) We will color the vertices in V with elements of the set $[k_1] \times [k_2]$. Indeed, $c(v) = (c_1(v), c_2(v))$ gives a proper k_1k_2-coloring of $G = G_1 \cup G_2$, because if two vertices u, v are adjacent in G, then they are also adjacent in some G_i, hence the i'th coordinate of their colors will differ. So $\chi(G) \leq k_1k_2$. □

Further simple bounds on the chromatic number can be found in the problem set.

Let $G = (V, E)$ be a graph. We say that G is *d-degenerate* if every subgraph of G has a vertex of degree less than or equal to d. The following proposition connects the degeneracy of G and the chromatic number of G. The six color theorem was in fact a special case of this lemma.

Lemma 8.40. *If G is d-degenerate, then $\chi(G) \leq d + 1$.*

Proof. We do induction on the number of vertices. The statement is clearly true for all G with at most $d + 1$ vertices. For larger graphs, pick a vertex v of degree $\leq d$ in G. The graph $G - v$ is d-degenerate (because every subgraph of $G - v$ is a subgraph of G), so it can be $(d + 1)$-colored. Then color v into the color that does not appear among the colors of its neighbors. This gives a proper coloring of G. □

Corollary 8.41. *$\chi(G) \leq \Delta(G) + 1$ for every graph G.*

Proof. G is clearly $\Delta(G)$-degenerate. □

This corollary is tight for cliques and odd cycles. Interestingly, those are basically the only examples, as shown by the following theorem.

Theorem 8.42 (Brooks, 1941)**.** *Let G be a connected graph that is not isomorphic to K_s or C_{2k+1} for any s or k. Then $\chi(G) \leq \Delta(G)$.*

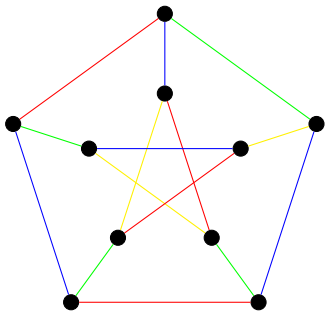
We will not prove this theorem.

3 Edge coloring

Definition. *A* proper edge coloring *of a graph G is a map $c : E(G) \to X$ for some set of colors X, such that $c(e) \neq c(e')$ whenever e, e' are distinct edges that share a vertex. The* edge-chromatic number *$\chi'(G)$ of G is the minimum size of such a set X, i.e., it is the minimum number k such that the G can be (properly) k-edge-colored.*

Example.

- $\chi'(K_3) = 3$
- $\chi'(C_4) = 2$
- $\chi'(K_4) = 3$
- *The picture on the right is an edge coloring of the Petersen graph with four colors. It is not difficult to see that its edge chromatic number is equal to 4.*



Definition. *A* matching *M is a set of vertex-disjoint edges, i.e., a set of edges such that no two of them share an endvertex.*

Each color class is a set is a matching, so an edge coloring is a partition of the edges into matchings.

Lemma 8.43. *For any graph G with at least one edge we have $\Delta(G) \leq \chi'(G) \leq 2\Delta(G) - 1$.*

Proof. There is vertex of degree $\Delta(G)$, and an edge coloring must give different colors to each of the $\Delta(G)$ edges at that vertex. This implies $\chi'(G) \geq \Delta(G)$.

The upper bound follows by a proof similar to the proof of Lemma 8.40: We can do induction on the number of edges. So delete an edge e, and take a $(2\Delta(G)-1)$-edge-coloring of $G-e$. Since an edge shares a vertex with at most $2(\Delta(G)-1)$ edges, there must be a free color left for e. □

However, this simple upper bound can be significantly improved. This is shown by the following theorem, which tells us that any graph G has edge-chromatic number either $\Delta(G)$ or $\Delta(G)+1$. Both are possible, since even cycles have $\chi_e(G) = \Delta(G)$ and odd cycles have $\chi_e(G) = \Delta(G)+1$. However, the algorithm in the proof still does not always give the exact number, and in fact it is NP-hard to determine which of the two values is the edge-chromatic number of a given graph.

Theorem 8.44 (Vizing, 1964)**.** *For every graph G we have $\chi'(G) = \Delta$ or $\Delta(G)+1$.*

Proof. We only need to show $\chi'(G) \leq \Delta(G)+1$. To prove this, we use induction on the number of edges. The statement clearly holds for a graph without edges.

Given an edge $xy \in E(G)$, we describe an algorithm that, given an edge coloring of $G-xy$ with at most $\Delta(G)+1$ colors, produces an edge coloring of G with this many colors. But to find a color for xy, the algorithm may have to change the colors of other edges.

So take a coloring of $G-xy$. As there are more than $\Delta(G)$ available colors, every vertex has at least one color missing from the edges touching it. We say that a vertex v is c-free, if no edge incident to v has color c.

Now let us build a fan of edges around x as follows. Take the edge xy, and let $y_0 = y$. We know that y_0 is c_1-free for some color c_1. Now let xy_1 be an edge of color c_1, if such an edge exists. Then y_1 is c_2-free for some color c_2.

Continue similarly: if there is an edge at x of color c_i that we have not looked at, let xy_i be this edge, and let c_{i+1} be a color missing at y_i. We repeat this process until we can; let xy_k be the last edge we added. There are two possible reasons for getting stuck: either there is no edge at x of color c_{k+1}, or this edge already appeared at some previous xy_i.

Case 1. x is c_{k+1}-free.

In this case we can just "rotate" the colors around x: Define the new color of xy_i to be c_{i+1} for every $i = 0, \ldots, k$ (keeping the colors of all other edges). This does not create any issue with any y_i, because it had been c_{i+1}-free. There is no problem at x either, because the only new color introduced is c_{k+1}. Hence this is a proper coloring of G (including the edge $xy = xy_0$).

Case 2. x is not c_{k+1}-free.

As noted above, the maximality of k implies that some edge xy_i is c_{k+1}-colored, so by definition $c_{k+1} = c_i$. Let us just call this color c, and let d be a color such that x is d-free.

Now let P be a maximal path starting at x that only uses the colors c and d. Actually, since every vertex touches at most one c-colored and one d-colored edge, and x is d-free, there is a unique such maximal path $P = xy_i \ldots z$ for some vertex z. Let us swap the colors c and d along the path. The same observation shows that this is still a proper edge-coloring of $G-e$. We again distinguish two cases.

Case 2/A. $z = y_{i-1}$.

Note that y_{i-1} was c-free, so the last edge of P must have had color d. After swapping, y_{i-1} is no longer c-free, but it is now d-free. However, the edge xy_i has color d, as well, so the edges xy_i satisfy the required properties with $c_i = d$ now. As x is now c-free, we arrive

at a situation covered by Case 1, which gives a proper coloring of G (by rotating the colors, i.e., coloring each xy_j with color c_{j+1}).

Case 2/B. $z \neq y_{i-1}$.

In this case, y_{i-1} remains c-free, and x becomes c-free, as well. This means that we can rotate the colors on the first i edges only to get a proper edge coloring: recolor xy_j with the color c_{j+1} for every $j = 0, \ldots, i-1$, and leave the color of other edges the same as what we had after swapping c and d along P. By the same arguments as before, every edge is colored, but no two touching edges get the same color. □

Lecture 9

Connectivity.

---

1 Menger's theorem

Next we will look at what the Ford-Fulkerson theorem says about networks where every edge has capacity 1. The results in this section hold for directed and undirected graphs, as well. We use the following notation: if $G = (V, E)$ is a graph and $W \subseteq V, F \subseteq E$, then $G - W$ is the graph with vertex set $V \setminus W$ and edge set $E \setminus \{e \in E : e \cap W \neq \emptyset\}$, and the graph $G \setminus F$ is the graph with vertex set V and edge set $E \setminus F$.

Definition. *Let $G = (V, E)$ be a (directed) graph, and $s, t \in V$.*

We say that two paths from s to t (s-t paths) in G are edge-disjoint*, if they don't share any edges. We say that they are* internally vertex-disjoint*, if they don't share any vertices other than s and t.*

A subset $F \subseteq E$ is an s-t edge separator*, if $G - F$ contains no s-t path. A subset $W \subseteq V$ is an s-t* vertex separator*, if $G - W$ contains no s-t path.*

The following theorem is one of the cornerstones of graph theory.

Theorem 9.45 (Menger)**.** *In a (directed) graph $G = (V, E)$ with $s, t \in V$:*

1. *Maximum number of edge-disjoint s-t paths = minimum size of an $s-t$ edge separator.*
2. *If $(s, t) \notin E$, then:*
   Max number of internally vertex-disjoint s-t paths = min size of an $s-t$ vertex separator.

Proof. "$\leq$" is clear in both statements: To "destroy" all the s-t paths, we need to delete a distinct edge/vertex from each of them. So any separator has size at least the number of paths.

Now we show "$\geq$" for *directed* G.

1. Take the network on G where all edges have capacity 1. We will prove

   $$\max \#\ s\text{-}t \text{ edge-disjoint paths} \geq \text{max flow} = \text{min cut} \geq \min\ s\text{-}t \text{ edge separator}$$

   For the first inequality, take a maximum *integer* flow f. If $|f| > 0$, then there is an s-t path P using flow edges. Removing P from f decreases the value of f by 1. More precisely, we "push back" a capacity-1 flow on P to decrease $|f|$ by 1. Crucially, as all capacities are 1, the new flow does not use any edge of P. So if we repeat this step $|f|$ times, we get $|f|$ edge-disjoint s-t paths, just what we wanted.

   The equality in the middle is just the max-flow min-cut theorem (Theorem 6.34). For the last inequality, note that the edges appearing in a min cut form an s-t edge separator. As the edge capacities are all 1, the capacity of this cut is exactly the number of edges in the edge separator.

2. The key to ensure edge-disjointness in the first statement was to limit the edge capacities to 1. To limit the capacities of vertices, we need a trick. Let us define the network $\tilde{G}$ based on G as follows:

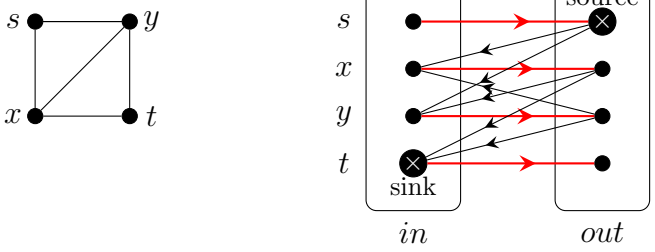
- for every vertex $v \in V$, we add two vertices v_{in} and v_{out} to $\tilde{G}$, connected by a directed edge (v_{in}, v_{out}) of capacity 1,
- for every directed edge $(u, v) \in E$, we add (u_out, v_in) of capacity ∞ to $\tilde{G}$.

Again, we prove

$$\max \#\text{ disjoint } s\text{-}t \text{ paths} \geq \max\ s_{out}\text{-}t_{in} \text{ flow in } \tilde{G} = \min \text{ cut} \geq \min\ s\text{-}t \text{ separator}.$$

For the first inequality, take a maximum integer s_{out}-t_{in} flow f in $\tilde{G}$. If $|f| > 0$, then there is an s-t path P using flow edges. As $(s, t) \notin E$, P must use a "vertex edge" (v_{in}, v_{out}), and so the capacity of the path P is 1. This means that pushing back capacity 1 on P decreases $|f|$ by one, and ensures that the "vertex edges" appearing in P are not used by the remaining flow. So repeating this we get $|f|$ paths in $\tilde{G}$, and the corresponding paths in G are internally vertex disjoint.

The equality in the middle is again the max-flow min-cut theorem. For the last inequality, note that the min cut has finite capacity (because $(s, t) \notin E$), so all edges in the cut are "vertex edges" corresponding to a vertex separator. They all have capacity 1, so the capacity of this cut is exactly the number of vertices in the separator.



To prove the undirected variants of Menger's theorem, one can just replace every undirected edge with two directed edges (one in each direction), and apply directed Menger to it. The details are left as an exercise. □

2 Connectivity

From now on, we again talk about undirected graphs.

Definition. *$G = (V, E)$ is* k-connected *if $|V| > k$ and $G - X$ is connected for any set X of at most $k - 1$ vertices. The greatest k such that G is k-connected is the connectivity $\kappa(G)$. G is* k-edge-connected *if $G \setminus F$ is connected for every set F of at most $k - 1$ edges. The greatest k such that G is k-edge-connected is the edge-connectivity $\kappa'(G)$ of G.*

Let us make some observations.

- G connected $\iff$ G is 1-connected $\iff$ G is 1-edge-connected.
- If G is k-(edge-)connected, then it is $(k - 1)$-(edge-)connected.
- $\kappa(K_s) = s - 1$, $\kappa(C_k) = \kappa'(C_k) = 2$

Let G be a connected graph. A *bridge* is an edge e such that $G \setminus e$ is disconnected. A *cut vertex* is a vertex v such that $G - v$ is disconnected.

- G is 2-connected $\iff$ G has no cut vertex.
  G is 2-edge-connected $\iff$ G has no bridge.

Theorem 9.46 (Global version of Menger's theorem)**.** *Let $G = (V, E)$ be a graph.*

1. *G is k-edge-connected $\iff$ G contains k edge-disjoint paths between any two vertices*
2. *G is k-connected $\iff$ G contains k internally vertex-disjoint paths between any two vertices*

This is a strong characterization of k-connected graphs that easily implies the following non-trivial facts.

Proposition 9.47. $\kappa(G) \leq \kappa'(G) \leq \delta(G)$.

Proof. $\kappa'(G) \leq \delta(G)$ is trivial, because deleting all edges touching a vertex disconnects G.

To show $\kappa(G) \leq \kappa'(G)$, note that by Theorem 9.46, any two vertices of G are connected by $\kappa(G)$ internally vertex-disjoint paths. But then these paths are edge-disjoint, so any two vertices are connected by $\kappa(G)$ edge-disjoint paths, hence G is $\kappa(G)$-edge-connected by the other direction of Theorem 9.46. □

Proposition 9.48. *G is 2-connected $\iff$ any 2 vertices of G are on a cycle.*

Proof. By Theorem 9.46, G is 2-connected if and only if any two vertices are connected by 2 internally vertex-disjoint paths. But this happens if and only if the two vertices are on a cycle (the union of the two paths). □

Proof of Theorem 9.46. After the local version, this is not hard to prove.

1. $\Leftarrow$: If s and t are connected by k edge-disjoint paths, any s-t edge-separator needs to contain at least one edge from each path. As this is true for any pair of vertices, there is no set F of at most $k-1$ edges such that deleting F disconnects G, thereby separating some two vertices.

   $\Rightarrow$: If G is k-edge-connected, then there is smallest s-t edge separator has size at least k for every $s, t \in V$. Then by Theorem 9.45, s and t are connected by k edge-disjoint paths for any $s, t \in V$.

2. $\Leftarrow$: The above argument works the same with vertices instead of edges.

   $\Rightarrow$: Again the same argument works, except for adjacent pairs $s, t \in V$. The problem is that we cannot apply Menger's theorem if $st \in E$. We fix this issue by deleting it, so let $G' = G \setminus st$.

   Claim. *G' is $(k-1)$-connected.*

   Proof of Claim. Suppose not, i.e., there is a set X of at most $k-2$ vertices such that $G' - X$ is disconnected. Note that $G - X$ is not disconnected (G is k-connected), and $G' - X = (G - X) \setminus st$, so s and t must be in different components of $G' - X$. Now as G is k-connected, it has at least $k+1$ vertices, so there is a vertex w not in X and different from s, t. We may assume that w and t are in different components of $G' - X$ (if not, then w and s are). But then $X' = X \cup s$ is a set of $k-1$ vertices such that $G - X = G' - X$ (because deleting s deletes the edge st), and so $G - X$ is disconnected. This contradicts G being k-connected. □

Using this claim, we can apply Theorem 9.45 to G' and get $k-1$ internally vertex-disjoint s-t paths in G'. Together with the edge st, we get k such paths in G, as needed. All we have left is to prove this claim.

□

The following corollary of the global Menger's theorem, called the *fan lemma*, is a nice and highly applicable tool for k-connected graphs.

Lemma 9.49 (fan lemma)**.** *Let G be a k-connected graph. For every $x \in V(G)$ and $U \subseteq V(G)$ with $|U| \geq k$, there are k paths from x to U that are disjoint aside from x, and each has exactly one vertex in U.*

Proof. Add a vertex u that is adjacent to all the vertices in U. It is an easy exercise to check that the resulting graph is still k-connected (note that this requires $|U| \geq k$). By Theorem 9.46, there are k internally vertex-disjoint paths from x to u. Removing u from these paths gives paths from x to U that are disjoint aside from x. If such a path touches more than one vertex in U, then it contains a subpath with exactly one vertex from U, which we can take instead. □

Lecture 10

The probabilistic method.

Probabilistic tools turn out to be extremely useful for certain combinatorial problems. At first, applications might feel like magic: problems that have nothing to do with probability often have simple solutions if we introduce some randomness. Deeper theoretical results (like the so-called Szemerédi regularity lemma) show some inherent connection between graphs and random structures.

However, (fortunately or unfortunately) this is not the course to go this deep, so we will have to stick with the magical results.

1 Review of basic notions

We will use standard notions from probability theory, although in a quite restricted setting: we will always work with a probability space that is defined on a *finite* base set Ω with a probability mass function $p : \Omega \to [0, 1]$ satisfying $\sum_{\omega\in\Omega} p(\omega) = 1$. The *events* in this probability space are all subsets $E \subseteq \Omega$, and the *probability* that E holds is $\mathbb{P}[E] = \sum_{\omega\in A} p(\omega)$.

A *random variable* is just a function $X : \Omega \to \mathbb{R}$, and the *expected value* or *expectation* of a random variable X is

$$\mathbb{E}[X] = \sum_{x\in\mathbb{R}} x \cdot \mathbb{P}[X = x] = \sum_{\omega\in\Omega} p(\omega) \cdot X(\omega).$$

A very important fact (that can be easily seen from the right-hand side of this equality) is the *linearity of expectation*, i.e., that for any two random variables X and Y and any real c, we have $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ and $\mathbb{E}[cX] = c\mathbb{E}[X]$.

Two events E_1 and E_2 are said to be *independent* if $\mathbb{P}[E_1 \cap E_2] = \mathbb{P}[E_1] \cdot \mathbb{P}[E_2]$. For more than two events, $E_1, \ldots, E_k$ are independent if for any $I \subseteq \{1, \ldots, k\}$, we have

$$\mathbb{P}\left[\bigcap_{i\in I} E_i\right] = \prod_{i\in I} \mathbb{P}[E_i].$$

Similarly, the random variables $X_1, \ldots, X_k$ are independent if for any $x_1, \ldots, x_k$ and any $I \subseteq \{1, \ldots, k\}$, we have

$$\mathbb{P}\left[\bigcap_{i\in I} X_i = x_i\right] = \prod_{i\in I} \mathbb{P}[X_i = x_i].$$

2 Simple applications

As a first example, let us give a new proof of the following fact. In Lecture 2, we gave an algorithmic proof, this time we provide a probabilistic argument.

Proposition 10.50. *Any graph G contains a bipartite subgraph H with $|E(H)| \geq |E(G)|/2$.*

Proof. Let us take a random partition of the vertices into parts A and B, where each vertex is independently assigned to A or B with probability $1/2$. So Ω is the set of all partitions. For every edge $e \in E(G)$, let X_e be the indicator random variable for the event that e "crosses"

between A and B. Then $X = \sum_{e \in E(G)} X_e$ is a random variable that counts the number of edges crossing between A and B. Thus the expected number of edges crossing is

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} \mathbb{E}[X_e].$$

Here for an edge $e = uv$, we have

$$\mathbb{E}[X_e] = \mathbb{P}[e \text{ crossing}] = \mathbb{P}[u \in A, v \in B] + \mathbb{P}[u \in B, v \in A] = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2},$$

so $\mathbb{E}[X] = \sum_{e \in E} 1/2 = |E(G)|/2$. But then there must be a partition $\omega \in \Omega$ such that $X(\omega) \geq |E(G)|/2$ (otherwise the expectation would be strictly smaller). □

Note that this is only an existence proof. We do not get any information on how to find such a subgraph.

For our next application, we give a new proof of the quantitative version of Turán's theorem. As a reminder, Turán's theorem implies that $\mathrm{ex}(n, K_r) \leq (1 - \frac{1}{r-1})\frac{n^2}{2}$. Taking graph complementation, this implies that if G is a graph with less than

$$\binom{n}{2} - \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2} = \frac{n^2}{2(r-1)} - \frac{n}{2}$$

edges, then G has an independent set of size r. In other words, if G is a graph with n vertices and e edges, then $\alpha(G) \geq \frac{n^2}{2e+n}$. We shall prove this equivalent formulation of Turán's theorem, and we will deduce this from an even stronger theorem, which takes the degree sequence of the graph in consideration.

Theorem 10.51. *(Caro) Let G be a graph. Then G contains an independent set of size at least*

$$\sum_{v \in V(G)} \frac{1}{1 + d(v)}.$$

Proof. Let $n = |V(G)|$ and let $v_1, ..., v_n$ be a random ordering of the vertex set, that is, an ordering chosen from the set of all possible ordering with probability $1/n!$. Let S be the set of vertices v_i such that v_i has no neighbor in $\{v_1, ..., v_{i-1}\}$. Then S is an independent set. Let us calculate the expected value of the size of S.

For a vertex v, let X_v be the indicator random variable that v is in S. That is, $X_v = 1$ if $v \in S$, $X_v = 0$, otherwise. Then $|S| = \sum_{v \in V(G)} X_v$. Also, the probability of the event $v \in S$ is exactly $\frac{1}{1+d(v)}$, as $v \in S$ if v is the first vertex in the ordering among $\{v\} \cup N(v)$. Hence, $\mathbb{E}(X_v) = \frac{1}{1+d(v)}$. But then by the linearity of expectation, we have

$$\mathbb{E}(|S|) = \sum_{v \in V(G)} \mathbb{E}(X_v) = \sum_{v \in V(G)} \frac{1}{1 + d(v)}.$$

But then there exists an ordering of the vertices such that $|S| \geq \sum_{v \in V(G)} \frac{1}{1+d(v)}$ and we are done. □

Corollary 10.52. *Let G be a graph with n vertices and e edges. Then*

$$\alpha(G) \geq \frac{n^2}{2e + n}.$$

Proof. By the inequality between the arithmetic and harmonic mean, we have

$$\frac{\sum_{v\in V(G)} \frac{1}{1+d(v)}}{n} \geq \frac{n}{\sum_{v\in V(G)} 1+d(v)} = \frac{n}{2e+n}.$$

But then $\alpha(G) \geq \sum_{v\in V(G)} \frac{1}{1+d(v)} \geq \frac{n^2}{2e+n}$. □

3 Constructions for the Kővári-Sós-Turán theorem

By the Kővári-Sós-Turán theorem, for every positive integer $s \geq 2$, we have

$$\mathrm{ex}(n, K_{s,s}) \leq cn^{2-1/s},$$

where $c > 0$ is a constant depending only on s. However, if $s \geq 4$, it is not known whether $\mathrm{ex}(n, K_{s,s}) \geq c'n^{2-1/s}$ also hold with some constant $c' > 0$, and this is one of the central open questions in extremal graph theory. In this section, we use the probabilistic method to find graphs with slightly less than $n^{2-1/s}$ edges that do not contain $K_{s,s}$.

Theorem 10.53. *Let $s \geq 2$ be a positive integer. Then*

$$\mathrm{ex}(n, K_{s,s}) > \frac{1}{16}n^{2-2/(s+1)}.$$

Proof. Let $p = \frac{1}{2}n^{-2/(s+1)}$ and let G be the random graph on an n element vertex set in which each pair of vertices is connected by an edge independently with probability p. The graph G might contain $K_{s,s}$, but we can make it $K_{s,s}$-free by deleting an edge from each copy of $K_{s,s}$. Let X be the number of copies of $K_{s,s}$ in G, more precisely, the number of pairs (A, B) such that $A, B \subset V(G)$, $|A| = |B| = s$, $A \cap B = \emptyset$ and $ab \in E(G)$ for every $a \in A$, $b \in B$.

Let us calculate the expectation of X. For any pair (A, B) such that $A, B \subset V(G)$, $|A| = |B| = s$, $A \cap B = \emptyset$, let $X_{A,B}$ be the indicator random variable that $ab \in E(G)$ for every $a \in A, b \in B$. Then $X = \sum X_{A,B}$. The probability that $ab \in E(G)$ for every $a \in A, b \in B$ is p^{s^2}, as this is the probability that s^2 given edges are present in G. Hence, $\mathbb{E}(X_{A,B}) = p^{s^2}$. The number of pairs (A, B) such that $A, B \subset V(G)$, $|A| = |B| = s$, $A \cap B = \emptyset$ is $\binom{n}{s}\binom{n-s}{s} < n^{2s}$. By the linearity of expectation, we get

$$\mathbb{E}(X) = \sum \mathbb{E}(X_{A,B}) \leq n^{2s}p^{s^2} = \frac{1}{2^{s^2}}n^{2-2/(s+1)}.$$

Let Y be the number of edges of G. Then $\mathbb{E}(Y) = \binom{n}{2}p > \frac{1}{4}n^2p = \frac{1}{8}n^{2-2/(s+1)}$.

Consider the random variable $Z = Y - X$. We have

$$\mathbb{E}(Z) = \mathbb{E}(Y) - \mathbb{E}(X) = \frac{1}{8}n^{2-2/(s+1)} - \frac{1}{2^{s^2}}n^{2-2/(s+1)} \geq \frac{1}{16}n^{2-2/(s+1)},$$

so there exists a graph G satisfying $Z(G) \geq \frac{1}{16}n^{2-2/(s+1)}$. Remove an arbitrary edge from each copy of $K_{s,s}$ in G and let H be the resulting graph. Then H is $K_{s,s}$-free, and ss we removed at most $X(G)$ edges, we have $E(H) \geq Y(G) - X(G) = Z(G) \geq \frac{1}{16}n^{2-2/(s+1)}$. □