6.1200J/18.062J *Mathematics for Computer Science* — Tuesday 19th March, 2024
Massachusetts Institute of Technology, Spring 2024
Z. Abel, B. Chapman, E. Demaine
revised Tuesday 19th March, 2024

Lecture 11: Graphs and Coloring

1 Graphs

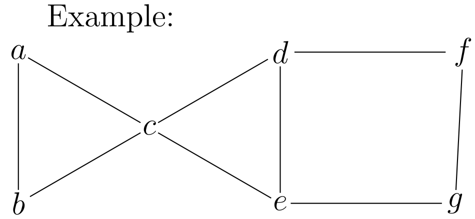
- Incredibly useful structures in computer science
- Applications:
  - Scheduling
  - Optimization
  - Communications
  - Design & Analysis of Algorithms
  - Dating apps (Nobel Prize!)
  - Search engine

1.1 Simple Graphs

Definition 1. *A* (simple) graph G *is a pair* (V, E)*, where* V *is a non-empty set of* vertices*, and* E *is a set of* 2*-element subsets of* V *called* edges*.*

Note: vertices are sometimes called *nodes*.

Example:



$$
\begin{aligned}
V &= \{a, b, c, d, e, f, g\} \\
E &= \{\{a,b\}, \{a,c\}, \{b,c\}, \{c,d\}, \{c,e\}, \{d,e\}, \{d,f\}, \{e,g\}, \{f,g\}\}
\end{aligned}
$$

Notation: $\{u, v\}$ denotes an edge between vertices u and v. We may also write this edge as $u - v$, uv, or (since edges are *sets*) $\{v, u\}$.

Warning: definitions differ slightly across sources!

- Some definitions allow E to be a multiset (multiple edges between the same pairs of vertices) or contain self-loops (edges with only one vertex).
- Our definition precludes such non-simple graphs.
- Some definitions allow a graph to have no vertices.
- We disallow $V = \emptyset$ because it is not terribly interesting and must be treated as a special case in some theorems we want to prove.
- We *do* allow graphs without edges.
  - Example: $G = (\{a, b, c\}, \emptyset)$
  - Non-Example: $G = (\emptyset, \emptyset)$
  - Non-Example: $G = (\{a, b, c\}, \{\{a\}\})$

1.2 Adjacency

Definition 2. *Two vertices u and v are* adjacent *iff they are connected by an edge, i.e. iff $\{u, v\} \in E$.*

Example: In the above graph, a is adjacent to c but not f.

Definition 3. *An edge $\{u, v\}$ is* incident *to u and v.*

Definition 4. *An edge $\{u, v\}$ has* endpoints *u and v.*

Example: In the above graph, $\{a, c\}$ is incident to a; a is an endpoint of $\{a, c\}$.

1.3 Graphs as a Useful Abstraction

Some real-world examples of graphs:

- Friendship (e.g., on Facebook): edge between a and b iff they are friends.
  - Not Twitter followers (since not symmetric); these could be modelled as a *directed* graph, which we'll discuss after break.
  - But twitter mutuals makes sense as an undirected graph!
  - "Six degrees of separation": claims any two people can be connected by a path of length ≤ 6 in this graph (or one like it). I'm not going to speculate on whether or not it's true, but interesting that it has a direct correspondence with some graph theoretic ideas!
- Scheduling conflicts: edge between two classes iff some student is taking both, so the final exams can't overlap. We'll see more about this later today!

- The brain (a network of neurons)
- The internet: most network communication links are bidirectional
  - By contrast, hyperlinks are *not* symmetric in general, would make more sense as a directed graph (we'll see these after break)

Graphs are a useful abstraction, because they strike a balance between simplicity and complexity:

- Graphs are conceptually simple to describe: just a bunch of objects (vertices) and pairwise relationships between them (edges). This means many real-world scenarios can be faithfully modelled using graphs.
- Graphs have a rich mathematical theory, with many powerful theorems, tools, and concepts that can then be applied to the graphs and thus to those real-world scenarios.

1.4 Vertex degree

Definition 5. *The* degree *of a vertex v is the number of edges incident to v.*

In friendship graph, it's the number of friends that person has.

In example graph above, the degrees are $(2, 2, 4, 3, 3, 2, 2)$. Sometimes called the "degree sequence" of the graph.

Does there exist a graph with degree sequence $(2, 2, 1)$? (Try it! Doesn't work.)

What about $(2, 2, 2, 2, 2, 1)$? Well, how many edges would it have?

Lemma 1 (Handshake Lemma)**.** *For a graph $G = (V, E)$, we have*

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|.$$

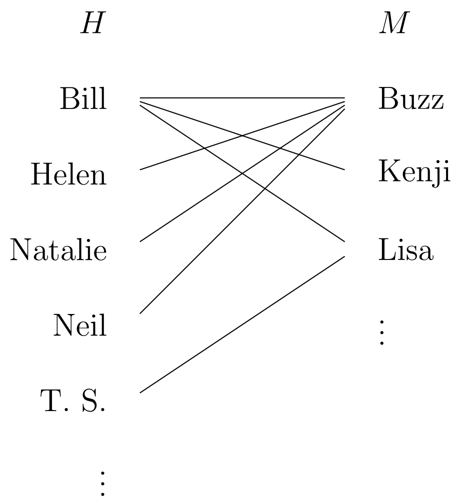
Proof. On left, for each vertex, count the number of edges. Since every edge has two endpoints, this counts every edge twice. □

So $(2, 2, 2, 2, 2, 1)$ is impossible, since it would have $11/2 = 5.5$ edges.

Another application: what's the maximum number of edges in an n-node graph? Each vertex has degree at most $n - 1$, so $2 \cdot |E| \leq n \cdot (n - 1)$, i.e., $|E| \leq (n - 1)(n)/2$. This is achieved by the n-vertex ***complete graph***, K_n, since every node has degree exactly $n - 1$.

1.5 Bipartite Graphs

For the typical MIT student, how many Harvard friends do they have? How many MIT friends does the typical Harvard student have? Which is bigger? Hmm, might need to ask around...



Let's define a graph:

- H = set of current Harvard undergrads, M = set of current MIT undergrads (these sets are disjoint!), $V = H \cup M$
- E = set of pairs $\{h, m\}$ where $h \in H$, $m \in M$, and h and m are friends

Definition 6. *A graph $G = (V, E)$ is* bipartite *if V can be partitioned into disjoint sets, often called L and R, such that every edge has one endpoint in each of L and R.*

Note that the two sets do not have to have the same size! In our example, $|H| \approx 7200$, and $|W| \approx 4600$. How many edges does this example have, i.e., how many Harvard-MIT friendships?

A bipartite version of handshake lemma says that

$$\sum_{h \in H} \deg(h) = |E| = \sum_{m \in M} \deg(m),$$

since each edge has exactly one endpoint in H and exactly one endpoint in M.

So if A_H is the average degree of nodes in H, and A_M is the average degree of nodes in M, then

$$A_H = \frac{1}{|H|} \sum_{h \in H} \deg(h) = |E|/|H|$$

and similarly $A_M = |E|/|M|$, so

$$A_M/A_H = (|E|/|M|)/(|E|/|H|) = |H|/|M| \approx 7200/4600 \approx 1.6$$

So the average MIT student has 1.6 times more Harvard friends than the typical Harvard student has MIT friends. Don't need to know $|E|$ or the distribution of these friendships, all that matters is the ratio of the number of nodes in the left and right sides of the bipartite graph.

In general, in a bipartite graph $G = (V, E)$ with V partitioned into L and R (and at least one edge),

$$\frac{\text{average degree of nodes in } L}{\text{average degree of nodes in } R} = \frac{|R|}{|L|}.$$

1.5.1 Studies About Romantic Partners

There's a famous 1994 UChicago study (*The Social Organization of Sexuality: sexual practices in the U.S.*). It followed 2500 people over several years, and had a 700 page writeup collecting many conclusions and statistics across many subsets of the population. One particular claim from the study focuses specifically on opposite-gender pairings among cisgender men and women. They asked the men how many female partners they've had, and likewise asked the women how many male partners they've had. They concluded that men have ≈ 1.74 times as many female partners as women have male partners. A similar 2004 study from ABC News (lauded as one of the most scientific studies ever done on the topic) claimed this number to be ≈ 3.33. And yet another study from the National Center for Health (2007) concluded it was ≈ 1.75. Other relationships, genders, and identities were also considered in other parts of these studies, but I'm focusing on these particular results for one reason: they're clearly incorrect, because of simple graph theory.

Because this survey question only asks about relations between one man and one woman, we're in exactly the same bipartite scenario as before, with the set M of men on the left, the set W of women on the right, and edges representing relationships. Collectively, men and women have the same *total number* of opposite-gender partners, because such a relationship includes one of each.

So if A_M and A_W are the average degree for men and women respectively, then $A_M/A_W = |W|/|M|$, which is approximately 157mil/152mil ≈ 1.03 according to 2010 census data. Doesn't seem like a very useful subject for a behavioral study – it has nothing to do with behavior, just population counts. Amusingly, an author of the 2007 study reported that she knew the results had to be wrong, but it was her duty to report on the data they had collected.

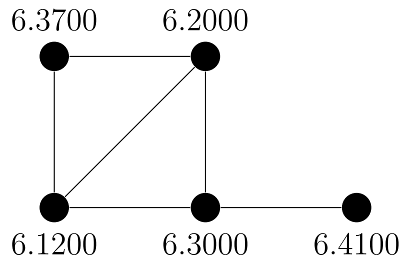
2 Coloring

- Last section: edges denote affinity between two vertices
- This time let's model the opposite: edges denote *conflict* between two vertices

2.1 Exam Scheduling

- Every class needs an exam time
- Two classes conflict if there are many students enrolled in both
- Conflicting classes shouldn't have exams at the same time

Example scheduling graph G:



- As defined so far, this is an easy problem to solve!

5pm	6.4100
7pm	6.3700
9pm	6.3000
11pm	6.2000
1am	6.1200

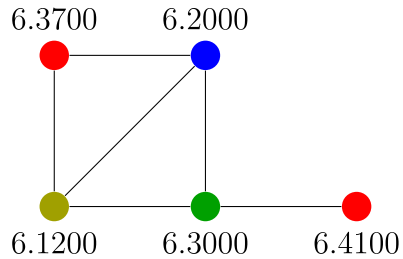
- Some 6.1200 students might complain...
- Much better if we can squeeze the exams into fewer time slots (but without creating conflicts)
- Well-known problem in graph theory!

Graph Coloring Problem: Given a graph $G = (V, E)$ and k colors, assign a color to each vertex so that no two adjacent vertices share a color.

Definition 7. *A* proper k coloring f *of* $G = (V, E)$ *is a function* $f : V \to C$ *such that* $|C| \le k$*, and for every edge* $\{u, v\} \in E$*,* $f(u) \neq f(v)$*. In other words, every vertex in* V *get assigned one of the colors in* C*, where no edge has endpoints assigned the same color.*

Definition 8. *The* chromatic number $\chi(G)$ *of* G *is the minimum* k *for which* G *has a proper* k*-coloring.*

Example 4-coloring of G:



This gives an improved schedule:

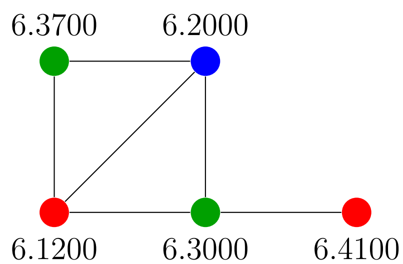
5pm 6.4100, 6.3700

7pm 6.3000

9pm 6.2000

11pm 6.1200

- Still not so great for 6.1200...
- Can we do better?
- 3 time slots?



Now we don't need 11pm:

5pm 6.4100, 6.1200

7pm 6.3000, 6.3700

9pm 6.2000

- What about 2 slots?
- Impossible: 6.3700, 6.2000, 6.1200 all need different time slots!
- G has a proper 3-coloring, but no proper 2-coloring.
- $\chi(G) = 3$.

To emphasize: to prove $\chi(G) = 3$, we had to prove both an upper bound and a lower bound:

- $\chi(G) \leq 3$ because G can be 3-colored. (We proved this by finding a 3-coloring.)
- $\chi(G) > 2$ because G cannot be 2-colored. (We proved this by contradiction.)

In general, $\chi(G)$ is defined as the *smallest* number of colors needed, so you need to show that $\chi(G)$ colors are enough, *and* that fewer colors are insufficient.

2.2 Applications

- Map coloring
  - Four-color theorem is difficult to prove
  - Very simple proof that six colors suffice
- Register allocation
- Radio tower broadcast frequencies
- Scheduling
- Akamai

2.3 NP-completeness

- As it turns out, computing the chromatic number of large graphs is *very* hard.
- Oddly enough, given a candidate coloring, it is easy to *verify* whether or not it is a proper k-coloring.
- Best known algorithm to *find* a k-coloring is essentially brute-force search.
- In fact, even determining whether $\chi(G) = 3$ is hard (3-Coloring problem).
- 3-Coloring and Graph Coloring are NP-complete.
  - Many well-studied problems are NP-complete
  - Easy to verify a candidate solution
  - No known algorithm to find an optimal solution
  - *Equivalent* to all other NP-complete problems in the sense that an algorithm for one solves all of them
  - $1M question: Does such an algorithm exist?

2.4 Approximation

- Graph Coloring is NP-complete, but we still want a solution!
- Can we find a near-optimal solution quickly?

Basic algorithm:

1. Order the vertices $v_1, v_2, \ldots, v_n$
2. Order the colors $c_1, c_2, \ldots$
3. For each vertex in order, assign it the smallest legal color

- Different orders give different colorings, and even different numbers of colors!
- Both the 3-coloring and the 4 coloring above were generated with this algorithm.
- Very hard problem to figure out good orders
- Basic algorithm uses the *greedy* paradigm:
  - Pick vertices
  - Assign colors
  - Don't look back!
  - Greedy algorithms are usually simple
  - Performance can often be analyzed

Theorem 2. *For all $d \geq 0$ and for all graphs G, if every vertex in G has degree at most d, then for all vertex orders, the Basic algorithm uses at most $d+1$ colors for G.*

Example:

- Scheduling graph has max degree 3
- Basic algorithm uses at most 4 colors (sometimes fewer)

2.5 Induction on Graphs

- As with most theorems in this class, we first try to prove Theorem 1 by induction.
- What is IH?
  - Only integer variable is d, so perhaps try $P(d) :=$ "For all graphs G, if every vertex in G has degree at most d, then for all vertex orders, the Basic algorithm uses at most $d+1$ colors for G."

  - How to prove $P(d) \Rightarrow P(d+1)$? With great difficulty...
  - Try something stronger? Even more disaster...

- Idea: induction on *size* of G!
  - First try induction on $|V|$
  - Next try induction on $|E|$
  - Then look for alternatives

Proof of Theorem 1. By induction on $|V|$:

- Inductive Hypothesis: $P(n) :=$ "For all d and all n-vertex graphs G, if every vertex in G has degree at most d, then for all vertex orders, the Basic algorithm uses at most $d+1$ colors for G."
- Base case ($n = 1$): Then G has 0 edges, so $d = 0$. The Basic algorithm uses 1 color for G.
- Inductive step: Assume $P(n)$ for purpose of induction. We wish to prove $P(n+1)$. To this end, let $G = (V, E)$ be a graph with vertices $v_1, v_2, \ldots, v_{n+1}$ and max degree d. Notice that when coloring G, the Basic algorithm colors $v_1, v_2, \ldots, v_n$ in order, *without considering* v_{n+1}. It therefore produces the same coloring on those vertices as if they were the entire graph. Per the inductive hypothesis, the Basic algorithm colors $v_1, v_2, \ldots, v_n$ with at most $d+1$ colors. Now v_{n+1} has at most d neighbors, which collectively must have at most d colors. Therefore at least one of the first $d+1$ colors is available to use for v_{n+1}.
- By induction, $\forall n \in \mathbb{N}. P(n)$.

$\square$

Be warned: when inducting on graphs, it is doubly important to use your "proof outlining" skills to identify the proper proof structure (especially in the inductive step) before diving into the proof. A very common mistake is to set up a proof of $P(n)$ IMPLIES $P(n+1)$ intuitively instead of carefully breaking it down, and this intuition is very often incorrect – you'll see a concrete example of this in recitation, called *buildup error*. Students often feel that induction on graphs is "different" or "backwards", but it's in fact using the same induction principle in the same way as always – it's the intuition that often gets it backwards.

The difference comes from the fact that $P(n)$ often starts with "**for all** graphs with n vertices" (or sometimes edges instead of vertices); that "for all" means that when proving $P(n+1)$, you need to prove something about **all** graphs with $n+1$ vertices, which means your proof should start with "suppose G is any graph with $n+1$ vertices". (The incorrect intuition often starts with a graph with n vertices and then "builds up" to $n+1$ vertices, hence the name "buildup error".)

2.6 How good/bad is this greedy algorithm?

- Can we improve this upper bound? I.e. do all graphs with max degree d have a proper d-coloring?
- No: let K_n be the n-vertex graph with all possible edges
  - Called the *complete graph* on n vertices, or n-clique
  - K_n has max degree $n-1$, but $\chi(K_n) = n$
- Sometimes the upper bound is much *higher* than the actual number of colors used
- Let S_k be the graph with a single vertex of degree k and k vertices of degree 1
  - Called the *star graph* on $k+1$ vertices or the *complete bipartite graph* $K_{1,k}$
  - S_k has max degree k, but $\chi(S_k) = 2$, and in fact every vertex order causes the Basic algorithm to use two colors
- But there are graphs on which the Basic algorithm can perform terribly...
- Let $H_{k,k}$ be the bipartite graph with vertices $1, 2, 3, \ldots, 2n$, and an edge between vertices i and j iff $i + j$ is odd but not equal to $2n + 1$
  - Called the *crown graph* on $2n$ vertices
  - $\chi(H_{k,k}) = 2$
  - With vertex order $1, 2n, 2, 2n-1, 3, 2n-2, \ldots$, Basic algorithm uses n colors