Problem Set 6

Due: March 30

Reading: Chapter 9.5–9.9, Partial Orders; Chapter 11–11.6, Simple Graphs.
Skip Chapter 10, Communication Nets, which will not be covered this term.

Problem 1.
Let R_1, R_2 be binary relations on the same set, A. A relational property is preserved under product, if $R_1 \times R_2$ has the property whenever both R_1 and R_2 have the property.

(a) Verify that each of the following properties are preserved under product.

1. reflexivity,
2. antisymmetry,
3. transitivity.

(b) Verify that if *either* of R_1 or R_2 is irreflexive, then so is $R_1 \times R_2$.

Note that it now follows immediately that if if R_1 and R_2 are partial orders and at least one of them is strict, then $R_1 \times R_2$ is a strict partial order.

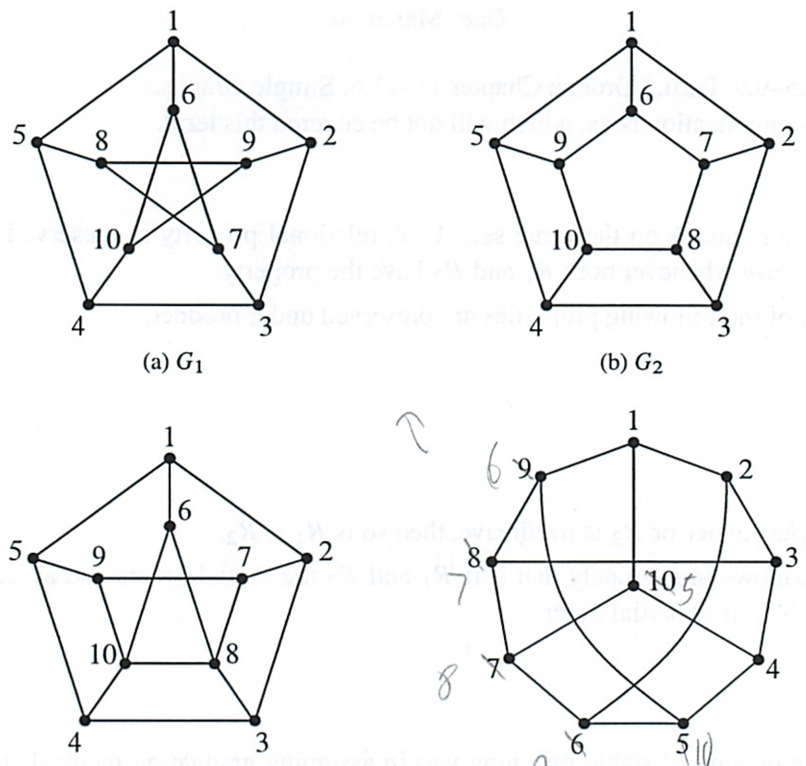
Problem 2.
The most famous application of stable matching was in assigning graduating medical students to hospital residencies. Each hospital has a preference ranking of students and each student has a preference order of hospitals, but unlike the setup in the notes where there are an equal number of boys and girls and monogamous marriages, hospitals generally have differing numbers of available residencies, and the total number of residencies may not equal the number of graduating students. Modify the definition of stable matching so it applies in this situation, and explain how to modify the Mating Ritual so it yields stable assignments of students to residencies.

Briefly indicate what, if any, modifications of the preserved invariant used to verify the original Mating are needed to verify this one for hospitals and students.

Problem 3.
Scholars through the ages have identified *twenty* fundamental human virtues: honesty, generosity, loyalty, prudence, completing the weekly course reading-response, etc. At the beginning of the term, every student in Math for Computer Science possessed exactly *eight* of these virtues. Furthermore, every student was unique; that is, no two students possessed exactly the same set of virtues. The Math for Computer Science course staff must select *one* additional virtue to impart to each student by the end of the term. Prove that there is a way to select an additional virtue for each student so that every student is unique at the end of the term as well.

Suggestion: Use Hall's theorem. Try various interpretations for the vertices on the left and right sides of your bipartite graph.

Problem 4.
Determine which among the four graphs pictured in the Figures are isomorphic. If two of these graphs are isomorphic, describe an isomorphism between them. If they are not, give a property that is preserved under isomorphism such that one graph has the property, but the other does not. For at least one of the properties you choose, *prove* that it is indeed preserved under isomorphism (you only need prove one of them).



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 1 Which graphs are isomorphic?

Problem 5. (a) For any vertex, v, in a graph, let $N(v)$ be the set of *neighbors* of v, namely, the vertices adjacent to v:

$$N(v) ::= \{u \mid u—v \text{ is an edge of the graph}\}.$$

Suppose f is an isomorphism from graph G to graph H. Prove that $f(N(v)) = N(f(v))$.

Your proof should follow by simple reasoning using the definitions of isomorphism and neighbors—no pictures or handwaving.

Hint: Prove by a chain of iff's that

$$h \in N(f(v)) \quad \text{iff} \quad h \in f(N(v))$$

for every $h \in V_H$. Use the fact that $h = f(u)$ for some $u \in V_G$.

(b) Conclude that if G and H are isomorphic graphs, then for each $k \in \mathbb{N}$, they have the same number of degree k vertices.

Doing P-Set 6

1. Two relations on set

relational property preserved under product

if have both

So what is this

↑ "concatinate"

9.9 product orders

$a R_1 b_1$ and $a_2 R_2 b_2$

what is this exactly

$a_1 \xrightarrow{R_1} b_1$ and

$a_2 \xrightarrow{R_2} b_2$

WP: Cartesaian product

$C_1 \times C_2$ is this same thing {suits} × {Ace, king, 10, 9, ...}

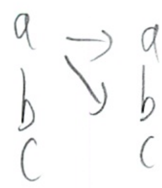
is the 52 cards

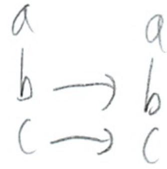
But this is a relation

- ~~sets,~~ domain, codomain, graph
- basically an arrow

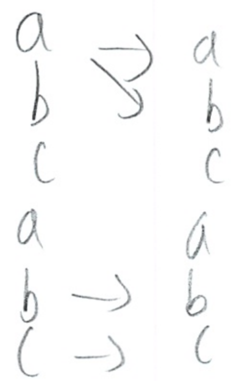
(2)

So like





Would be



or

Wish he did an example
One in book very unclear
So one relation younger
other shorter
So combined is both younger and shorter?
Both must have property
But how to show this problem is asking
Reflexitivity
- if aRa for all $a \in A$
But how to really explain?

(3)

2. How to modify

Was thinking about this

How much detail do they want?

(4)

3. 20 fund. human virtues

Each student 8 of these at start

" " unique set of virtue

Can an (1) addl. virtue be added so still unique

no

My first thought is we are adding - so of course more possibilities

What do we have now

1 1

2 2

3 3

⋮ ⋮

20 20

20^8 possibilities

- oh repeats can't count

$20 \cdot 19 \cdot 18 \cdot 17 \cdot 16 \cdot 15 \cdot 14 \cdot 13$

Not prob so don't need $\binom{20}{8}$

Or is this it?

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

5

$$\binom{20}{8} = 125970$$
$$= \frac{20!}{8! \times 12!}$$

$20 \cdot \ldots \cdot 13 = 5$ billion

- much larger

What is right?

$\binom{20}{8}$ is ~~?? prob that 8 chosen?~~

No the choose notation likely right

My guess - why wrong

1 round

20 possible

2 round

$20 \cdot 19$ possible

AB
AC
AD
$\Big]$ 19

...

BA
BC
BD
$\Big]$ 19

$= 380$

20 times

$\binom{20}{2} = 190$ So 1 half of above

(6)

Oh duh since

$$AB = BA$$

So ( ) is correct

But they suggest Hall's theorm

- which is set of women liked by men if at least one man likes

So every subset of ~~men~~ women must be ~~as~~ size $\geq$ ~~every~~ every subset of men

So what are matches?

I can solve w/ ( ) - but that is not what they want

- we are not supposed to have learned yet

How would you solve like they want?

? student → virtue

- but nothing we learned is about # of likes from each

Or student → virtue set 125,450

then student → new virtue set 167,960

? Or need something more ~~Read~~ - not the added item.

Don't foget Hall

7

4. isomorphic

- Same arrows
- diff labling + layout

- same graph isomophic
- describe
- don't get if not
- and 2 of 4 or what? guess just why not
- prove one property

Count verticies

G_1	G_2	G_3	G_4
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3
3	3	4	3
3	3	3	3
3	3	4	3

Oh but same # of verticies is not definitive
Must map vertices

(8)

G_1 G_2

1 → 1

. . . .

5 → 5

6 → 6

~~6~~

7 - goes to 6, 3, ~~7~~ 8

: so no ~~iso~~ isomorphism :

G_4

How to check exactly:

---

def: edge preserving bijection

isomorphism is the bij

two graphs isomorphic if isomorphism b/w them

transitive (duh)

Graph "preserved under isomorphism"

Look at preserved properties

└ what is that?

not in book

WP: property preserved under all isomorphisms

⑨

So its no use in finding isomophisms

Graph isomorphism property is hard

Can use matlab to check?

Too much work

I found one while doing write up

Hope writeup is enough

(10)

5. $N(v)$ is neighbors $N(v) ::= \{u \mid u-v \text{ is edge}\}$

f is isomophism

Prove $f(N(v)) = N(f(v))$

isomophism of neighbors | neighbors of ismophism

- Using def of iso + neighbors

(I get it - but how to write exactly?)

Use chain of iffs

$h \in N(f(v))$ iff $h \in f(N(v))$

for all $h \in V_H$

Use $h = f(u)$ for some $u \in V_G$

How to actually write?

This is what I don't get at all in this class

Ask Matt

He took awhile to figure out - but solved it in 3 lines

Still really don't get

- Study!

How get lines 1 + 2? Can you make it any more basic?

Matt's Help

3/29

Subset of ele ~~that~~ in G would be $N(v)$

H is in $N(v)$ if neighbor
└ def

Then iff $f(H)$ is in $f(N(v))$
H is el of $N(v)$

"i can just assume"

~~the~~ Each h is = to $f(v)$

Start $H = f(u)$
~~u~~ u is N of some v
~~know that~~
then sire $u - v$
and $H = f(u)$
v is in $f(v)$
def of isomorphism
H is in $N(f(u))$ iff $f^{-1}(G)$ is in $N(v)$

(bii?)

~~if u, v are N then~~

$f(u) \in N(f(v))$ IFF $u \in N(v)$ Def. of Isomorphism

$u \in N(v)$ IFF $f(u) \in f(N(v))$

Let $h = f(u)$ for some $u \in V_G$

$h \in N(f(v))$ IFF $h \in f(N(v))$

So ~~$f(N(v))$~~

(12)

Now b

- just conclude?

I am going to expand with some stuff

Still looking back - why do you need the h = part?

I will not be able to remember this!

I think I did pretty good on this

Student's Solutions to Problem Set 6

Your name: Michael Plasneior

Due date: March 30

Submission date: 3/30

Circle your TA/LA: Ali Nick Oscar Oshani

Table 12

Collaboration statement: Circle one of the two choices and provide all pertinent info.

1. I worked alone and only with course materials.
2. I collaborated on this assignment with:

   got help from:[1] Matt Fawk

   and referred to:[2] wikipedia, Cartesian product - not right topic
   Binomial coefficent
   Graph property
   Graph isomorphism

DO NOT WRITE BELOW THIS LINE

Problem	Score
1	
2	
3	
4	
5	
Total	

[1]People other than course staff.

[2]Give citations to texts and material other than the Spring '11 course materials.

1\. a. Reflexive

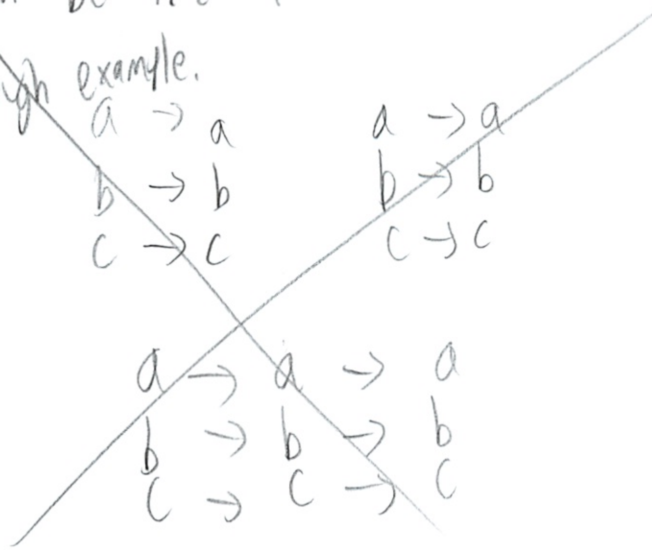
- aRa for all $a \in A$

10

This means that every item has a self arrow, at least

If this is true for both R_1 and R_2 then it will be true for $R_1 \times R_2$

~~Rough example.~~



If there is a self arrow for each item in both of the relations there will be a self arrow in the combined item

$$(a_1, a_2)(R_1 \times R_2)(b_1, b_2) \text{ iff } [a_1 R_1 b_1 \text{ and } a_2 R_2 b_2]$$

This is because $R_1 \times R_2$ means the edge must be in both relations - it must satisfy both conditions - ie younger + shorter (example from book)

(2)

antisymmetry -

aRb IMPLIES NOT (b R a) for all $a \neq b \in A$

This bacially means arrows are only allowed in one direction.

$\rightarrow$ or $\leftarrow$ NOT $\leftrightarrow$

If one relation is antisymmetric and the other relation is antisymetric then the whole thing will be antisymmetric.

Actually could say only one relation needs to be antisymmetric to make the $R_1 \times R_2$ antisymmetric - right?

- because the one relation "breaks" the ability to go backwards

[There is at most one edge between two points but can be self loops]

6.

(3)

transitivity —

$\forall x, y, z \in A. \ (xRy \text{ and } yRz)$ IMPLIES xRz

To me this is the definition of a product order, —
Well wait — no the example is something different.
Only have one domain and codomain — and must
satisfy both conditions — younger AND shorter
Never-theless. this condition still applies inside.

If there is a positive lenght path from u to v
then there simply can be an edge from u to v,
If both R_1 and R_2 have it, then $R_1 \times R_2$ will
have it because $R_1 \times R_2$ is the arrows that satisfy <u>both</u>
conditions.

(4.)

b. If either R_1 or R_2 is irreflexive then so is $R_1 \times R_2$

R is irreflexive when NOT $[\exists x \in A \quad x R x]$

Basically it means there can not be no self loops,

$R_1 \times R_2$ means that both conditions need to be true
- ie younger and shorter

If one of the relations does not have self-loops
- ie is irreflexive then $R_1 \times R_2$ will not have it

Michael Plasmeier
Oshani
Table 12

10

#2 We can modify the Mating Ritual so that we can assign students to hospital residencies.

However since # students ≠ # spots then we do not guarentee that every student (if # students > # spots) is placed or every spot (if # students < # spots) is filled.

One way to think of it is each spot is a seperate "balcony" and that all spots in a hosital have the same pref list. However this has some problems. Which spot/"balcony" should students stand under at a given hospital? — this does work.

A better approach would be that hospitals are one balcony but they keep their top N students where N is the number of spots at their certain hospital. If a student is not prefered and can't get a spot, then they go to their next choice hospital. Also this plan allows hospitals to have a diff. # of spots.

(2)

Remember stable matching means no hospital perfers a Student more AND student perfers that hospital more

This is still true.

Also true is Lemm 11.6.4: For every hospital h and student s if h is crossed of s's list then h has N students it perfers over s

Students get their optimal matching
Hospitals " " pessimal

Michael Plasmeier

Oshani P-Set 6

Table 12

#3 There are $\binom{20}{8}$ possible combos of the 8 virtues. This means that there are 125,970 possible combos - far larger than the size of 6.042.

When you add a virtue there are $\binom{20}{9}$ combos. This is 167,960 possible combos.

can you guarantee this for every subset of students?

Since 167,960 ≥ 125,970 it means that a unique solution can still be found, since there are now more possibilities. An extra virtue can always be imported.

each student can only end up with certain set of 9 virtues

Alterantivly we can represent this as a biparte graph

Students → Virtue Set of 8

~100 125,970

exactly 1 arrow out 0 or 1 arrow in

5

Then the bigger virtue set

Students → Virtue Set of 9

167,960

can still exactly 1 arrow out

still 0 or 1 arrow in since more possibilities

②

By Halls Theorm/ Matching Principle

The # of virtue sets must be $\geq$ # of students for every possible subset of students

Michael Plasmeier
Oshani
Table 12
P-Set 6

4

#4. First off we know that G_3 is not isomophic because all verticies in G_1, G_2, G_4 have a degree of 3. Some verticies in G_3 have degree 4.

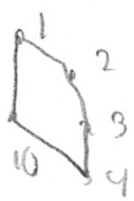
The other 3 remain canidates for isomorphism. However I was not able to find a match between some of them.

Can you be more precise?

G_1 and G_2 - If you would unravel the star into a pentagon (which would normally be possible) it would break the ring.

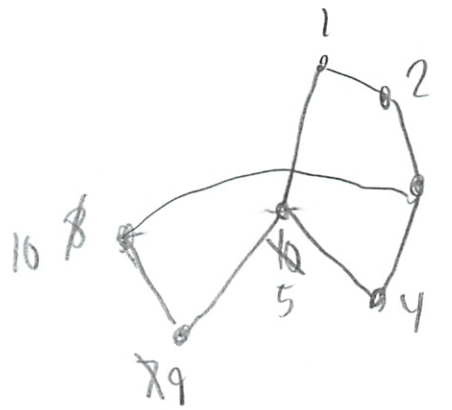
G_2 and G_4 - I could not find a way to map this. In G_4 what would be the ring. There needs to be a path that includes 5 vertices where the third path is to one of the other points included.

I was able to make such a path in G_4



②

However then point 10, which I labled as 5 did not connect well. I did mark 7 as 9 and 8 as 10 but then I was forced to mark 3 as 8. 3 should be part of the original ring, so I can not lable it so there is not an isomorphism

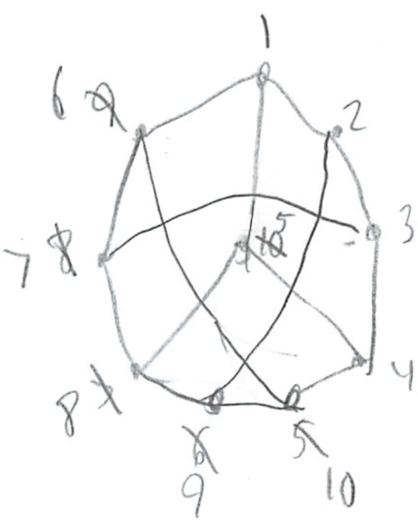


3 ← would have to be 8
contridiction

G₁ and G₄ - I was able to find an isomorphism between this. Again I labled 1, 2, 3, 4, 10 as 1, 2, 3, 4, 5. I then sought to lable the rest. In G₁ 5 is connected to 1, 4, 8. I already had 1, 4 so the third line from 5 in G₄ must be labled 8. (it was 7) 8 then connects to 7 and 9 in G₁. I saw that 8 in G₄ also connected to 3, just like 7 did in G₁ so I labled 8 as 7.

③.

This let me fill in 9 as 6 in G_4 - wich links to 2 - like in G_1. I then contined orand G_1 and saw that 9 was linked to 10. This must be 5 on G_4. 10 on G_1 links to 6. This must be 9 on G_4. All points have been relabled and confirmed sucessfully so an isomophism exists



proof of property?

In general, you don't need to give a story, it obscures your solution.

Michael Plasmeier
Oshani
Table 12
P-set 6

1/7 #5 Proof $f(N(v)) = N(f(v))$

same statement since using iff

$f(u) \in N(f(v))$ iff $u \in N(v)$ Def of Isomorphism

$u \in N(v)$ iff $f(u) \in f(N(v))$

no actual proof

Let $h = f(u)$ for some $u \in V_G$

$h \in N(f(v))$ iff $h \in f(N(v))$ for all $h \in V_H$

3/3 b. So if G, H are isomorphic, then for each $k \in \mathbb{N}$ they have the same # of degree k verticies

- property of isomorphism
- We showed above that for every $h \in V_H$ that $h = f(v)$ for some $v \in V_G$. This means that all the vertices need to isomorphic - and have some match in degree

Solutions to Problem Set 6

Reading: Chapter 9.5–9.9, Partial Orders; Chapter **??–??**, Simple Graphs.
Skip Chapter 10, Communication Nets, which will not be covered this term.

Problem 1.
Let R_1, R_2 be binary relations on the same set, A. A relational property is preserved under product, if $R_1 \times R_2$ has the property whenever both R_1 and R_2 have the property.

(a) Verify that each of the following properties are preserved under product.

1. reflexivity,
2. antisymmetry,
3. transitivity.

Solution. These facts follows directly from the definitions. We'll write out just the case of antisymmetry. So suppose R_1, R_2 are antisymmetric.

Proof. To prove $R_1 \times R_2$ is antisymmetric, suppose

$$(r_1, r_2)\ [R_1 \times R_2]\ (s_1, s_2) \quad \text{and also} \tag{1}$$
$$(s_1, s_2)\ [R_1 \times R_2]\ (r_1, r_2). \tag{2}$$

We need to show that $(r_1, s_1) = (r_2, s_2)$.

By (1) and the definition of $R_1 \times R_2$, we know that $r_i\ R_i\ s_i$ for $i = 1, 2$. Similarly, by (1) $s_i\ R_i\ r_i$. Since R_i is antisymmetric, it follows that $r_i = s_i$ for $i = 1, 2$. That is, $(r_1, s_1) = (r_2, s_2)$. ■

(b) Verify that if *either* of R_1 or R_2 is irreflexive, then so is $R_1 \times R_2$.

Solution. We may as well assume R_1 is irreflexive. This means that $\text{NOT}(r_1\ R_1\ r_1)$ for every $r_1 \in \text{domain}(R_1)$. So by definition of relational product,

$$\text{NOT}[(r_1, r_2)\ [R_1 \times R_2]\ (r_1, s_2)]$$

for all $r_1 \in \text{domain}(R_1)$ and $r_2, s_2 \in \text{domain}(R_2)$. In particular

$$\text{NOT}[(r_1, r_2)\ [R_1 \times R_2]\ (r_1, r_2)],$$

which implies that $R_1 \times R_2$ is irreflexive. ■

Note that it now follows immediately that if if R_1 and R_2 are partial orders and at least one of them is strict, then $R_1 \times R_2$ is a strict partial order.

Problem 2.
The most famous application of stable matching was in assigning graduating medical students to hospital residencies. Each hospital has a preference ranking of students and each student has a preference order of hospitals, but unlike the setup in the notes where there are an equal number of boys and girls and monogamous marriages, hospitals generally have differing numbers of available residencies, and the total number of residencies may not equal the number of graduating students. Modify the definition of stable matching so it applies in this situation, and explain how to modify the Mating Ritual so it yields stable assignments of students to residencies.

Briefly indicate what, if any, modifications of the preserved invariant used to verify the original Mating are needed to verify this one for hospitals and students.

Solution. The Mating Ritual can be applied to this situation by letting the students be the boys and each of the *residencies* (not the hospitals) be the girls.

A matching is an assignment of students to residencies (an injection, A : students $\rightarrow$ residencies) such that every student has a residency (A is total), or every residency has an assigned student (A is a surjection). A stable assignment is one with no *rogue couples*, where a rogue couple is a hospital student pair (H, S) such that S is not assigned to one of the residencies at H, which she prefers over her current assignment, and

- H has some students assigned to some of its residencies and prefers S to at least one of its assigned students, or
- H has none of its residencies assigned,

■

Problem 3.
Scholars through the ages have identified *twenty* fundamental human virtues: honesty, generosity, loyalty, prudence, completing the weekly course reading-response, etc. At the beginning of the term, every student in Math for Computer Science possessed exactly *eight* of these virtues. Furthermore, every student was unique; that is, no two students possessed exactly the same set of virtues. The Math for Computer Science course staff must select *one* additional virtue to impart to each student by the end of the term. Prove that there is a way to select an additional virtue for each student so that every student is unique at the end of the term as well.

Suggestion: Use Hall's theorem. Try various interpretations for the vertices on the left and right sides of your bipartite graph.

Solution. Construct a bipartite graph G as follows. The vertices on on the left are all students and the virtues on the right are all subset of nine virtues. There is an edge between a student and a set of 9 virtues if the student already has 8 of those virtues.

Each vertex on the left has degree 12, since each student can learn one of 12 additional virtues. The vertices on the right have degree at most 9, since each set of 9 virtues has only 9 subsets of size 8. So this bipartite graph is degree-constrained, and therefore, by Lemma **??**, there is a matching for the students. Thus, if each student is taught the additional virtue in the set of 9 virtues with whom he or she is matched, then every student is unique at the end of the term. ■

Problem 4.
Determine which among the four graphs pictured in the Figures are isomorphic. If two of these graphs are isomorphic, describe an isomorphism between them. If they are not, give a property that is preserved under

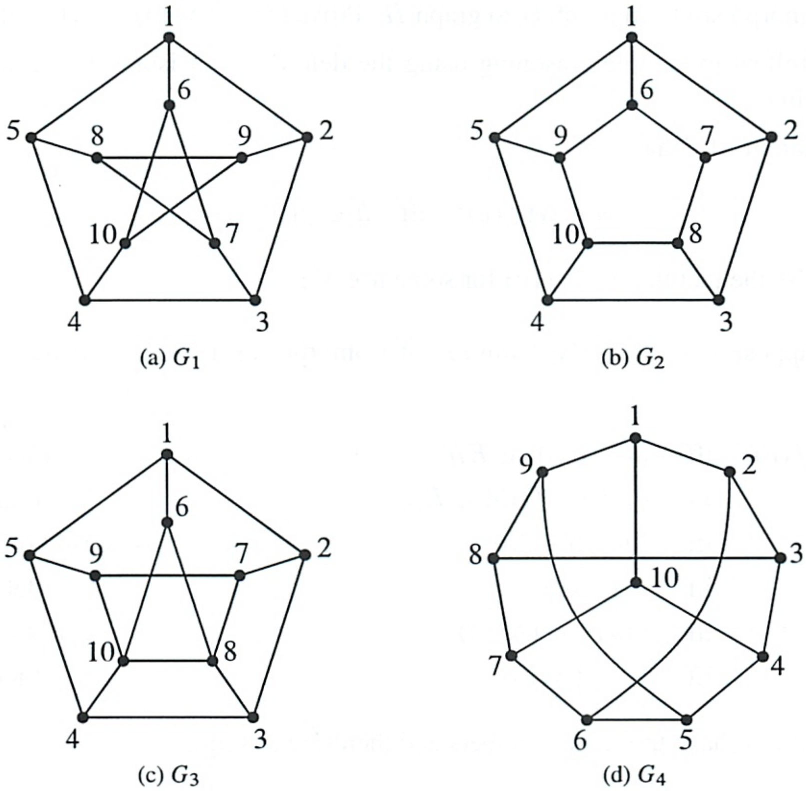

(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 1 Which graphs are isomorphic?

isomorphism such that one graph has the property, but the other does not. For at least one of the properties you choose, *prove* that it is indeed preserved under isomorphism (you only need prove one of them).

Solution. G_1 and G_3 are isomorphic. In particular, the function $f : V_1 \to V_3$ is an isomomorphism, where

$f(1) = 1$	$f(2) = 2$	$f(3) = 3$	$f(4) = 8$	$f(5) = 9$
$f(6) = 10$	$f(7) = 4$	$f(8) = 5$	$f(9) = 6$	$f(10) = 7$

G_1 and G_4 are not isomorphic to G_2: G_2 has a vertex of degree four and neither G_1 nor G_4 has one.

G_1 and G_4 are not isomorphic: G_4 has a cycle of length four and G_1 does not.

There are many examples of properties preserved under graph isomorphism. For example, we will prove that the degree of each vertex is preserved under isomorphism.

Let G and H be isomorphic graphs. Since they are isomorphic, there is an edge-preserving bijection between the vertices of G and H:

$$f(u) \in V(H) \longleftrightarrow f(u) \in V(G)$$

We let the set of vertices adjacent to u be $N(u)$. Because f is an edge-preserving bijection, there is an edge from $f(u)$ to a vertex $f(k)$ iff $k \in N(u)$. Thus $|N(f(u))| = |N(u)|$ and the degree of each vertex is preserved under isomorphism. ■

Problem 5. (a) For any vertex, v, in a graph, let $N(v)$ be the set of *neighbors* of v, namely, the vertices adjacent to v:

$$N(v) ::= \{u \mid \langle u\text{—}v\rangle \text{ is an edge of the graph}\}.$$

Suppose f is an isomorphism from graph G to graph H. Prove that $f(N(v)) = N(f(v))$.

Your proof should follow by simple reasoning using the definitions of isomorphism and neighbors—no pictures or handwaving.

Hint: Prove by a chain of iff's that

$$h \in N(f(v)) \quad \text{iff} \quad h \in f(N(v))$$

for every $h \in V_H$. Use the fact that $h = f(u)$ for some $u \in V_G$.

Solution. *Proof.* Suppose $h \in V_H$. By definition of isomorphism, there is a unique $u \in V_G$ such that $f(u) = h$. Then

$$\begin{aligned}
h \in N(f(v)) \quad & \text{iff} \quad \langle h—f(v)\rangle \in E_H && \text{(def of } N) \\
& \text{iff} \quad \langle f(u)—f(v)\rangle \in E_H && \text{(def of } u) \\
& \text{iff} \quad \langle u—v\rangle \in E_V && \text{(since } f \text{ is an isomorphism)} \\
& \text{iff} \quad u \in N(v) && \text{(def of } N) \\
& \text{iff} \quad f(u) \in f(N(v)) && \text{(def of } f\text{-image)} \\
& \text{iff} \quad h \in f(N(v)) && \text{(def of } u)
\end{aligned}$$

So $N(f(v))$ and $f(N(v))$ have the same members and therefore are equal.

■

(b) Conclude that if G and H are isomorphic graphs, then for each $k \in \mathbb{N}$, they have the same number of degree k vertices.

Solution. By definition, $\deg(v) = |N(v)|$. Since an isomorphism is a bijection, any set of vertices and its image under an isomorphism will be the same size (by the Mapping Rule from Week 2 Notes), so part (a) implies that an isomorphism, f, maps degree k vertices to degree k vertices. This means that the image under f of the set of degree k vertices of G is precisely the set of degree k vertices of H. So by the Mapping Rule again, there are the same number of degree k vertices in G and H. ■

3/30

Mathematics for Computer Science

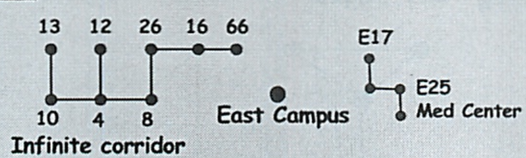
MIT 6.042J/18.062J

Graph Connectivity Trees & Coloring

Connected Components

Every graph consists of separate connected pieces (subgraphs) called connected components

Connected Components



3 connected components

the more connected components, the more "broken up" the graph is.

actually wrong now

Connected Components

The connected component of vertex v ::=

$\{w \mid v \text{ and } w \text{ are connected}\}$

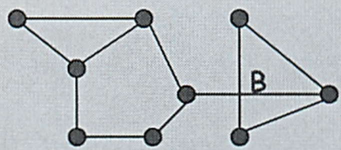
Connected Components

So a graph is connected iff it has only 1 connected component

Cut Edges

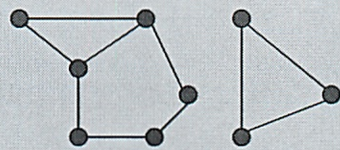
An edge is a cut edge if removing it from the graph disconnects two vertices.

Cut Edges



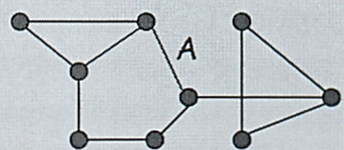
B is a cut edge

Cut Edges



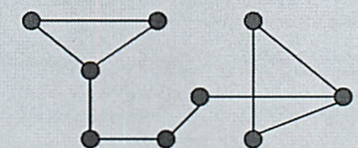
deleting B gives two components

Cut Edges



A is *not* a cut edge

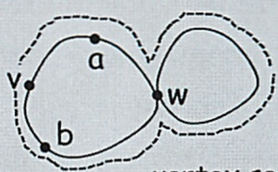
Cut Edges



still connected with edge A deleted

Closed Walks

A closed walk is a walk that begins and ends with the same vertex

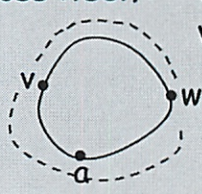


vertex sequence:
v···b···w···w···a···v

Cycles

A cycle is a closed walk of length > 2 that doesn't cross itself:

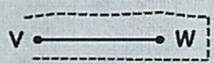
vertex sequence:
v···a···w···v



also:
w···a···v···w

Cycles

length $>$ 2 implies that going back & forth over an edge is not a cycle

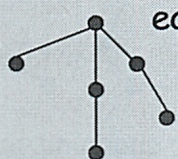


Cut Edges and Cycles

Lemma: An edge is a not a cut edge iff it is on a cycle.

Trees

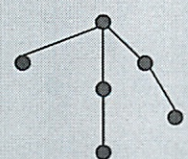
A tree is a connected graph with no cycles.



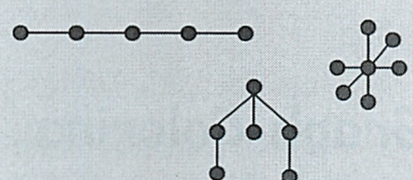
equivalently:

Trees

A tree is a connected graph with every edge a cut edge.



More Trees



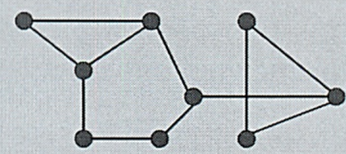
Other Tree Definitions

- graph with a unique path between any 2 vertices
- connected graph with n vertices and n-1 edges
- an edge-maximal acyclic graph

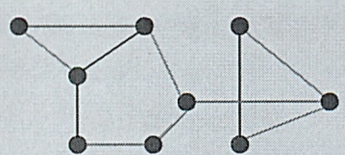
Spanning Trees

A *spanning tree* of a graph G is any subgraph T that is a tree and contains all the vertices of G.

Spanning Trees

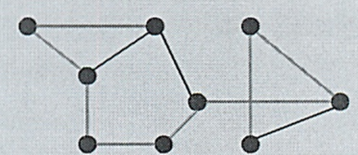


Spanning Trees



a spanning tree

Spanning Trees



another spanning tree
(can have many)

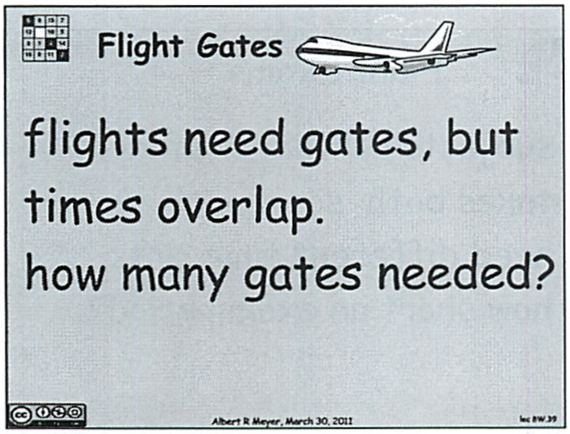
Spanning Trees

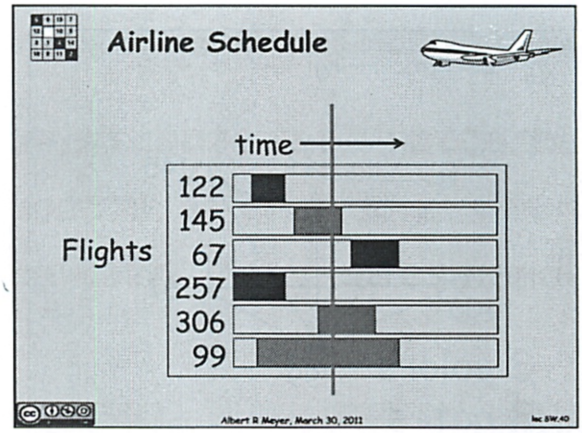
Lemma: G connected implies G has a spanning tree

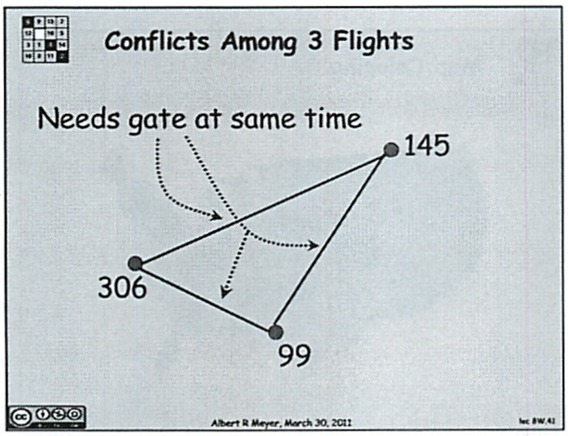
Pf: Among connected subgraphs with all the vertices of G: those with the fewest edges are spanning trees. (Why?)

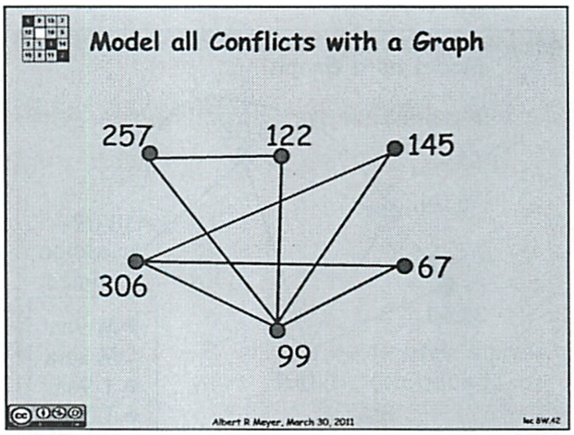
Mathematics for Computer Science
MIT 6.042J/18.062J

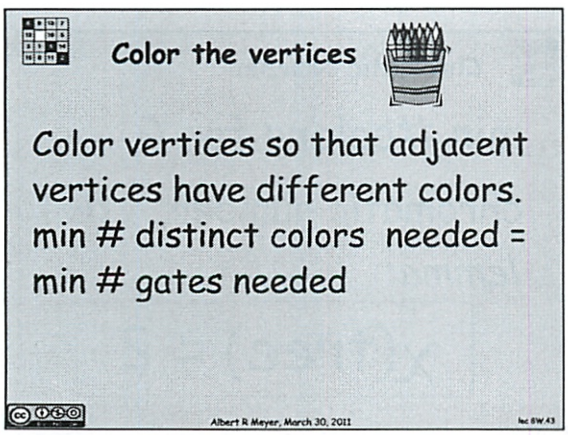
Graph Coloring

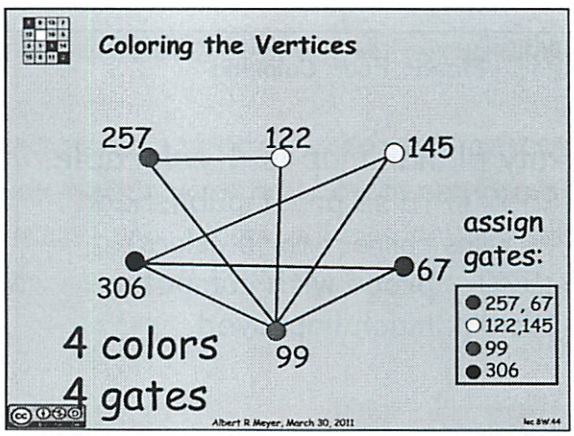

Flight Gates
flights need gates, but times overlap.
how many gates needed?
Albert R Meyer, March 30, 2011
lec 8W.39


Airline Schedule
time
Flights
122
145
67
257
306
99
Albert R Meyer, March 30, 2011
lec 8W.40

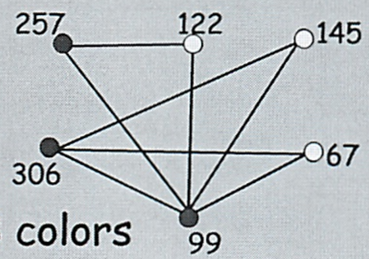

Conflicts Among 3 Flights
Needs gate at same time
145
306
99
Albert R Meyer, March 30, 2011
lec 8W.41


Model all Conflicts with a Graph
257
122
145
306
67
99
Albert R Meyer, March 30, 2011
lec 8W.42


Color the vertices
Color vertices so that adjacent vertices have different colors.
min # distinct colors needed = min # gates needed
Albert R Meyer, March 30, 2011
lec 8W.43


Coloring the Vertices
257
122
145
306
67
99
assign gates:
257, 67
122,145
99
306
4 colors
4 gates
Albert R Meyer, March 30, 2011
lec 8W.44

Better coloring

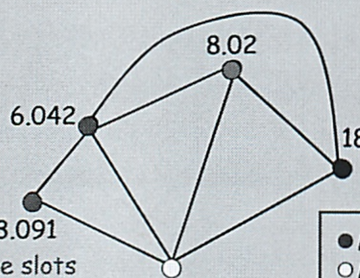


3 colors
3 gates

Final Exams

subjects conflict if student takes both, so
need different time slots.
how short an exam period?

Model as a Graph

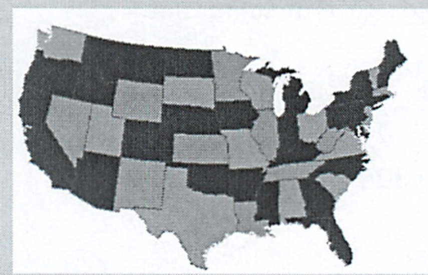


assign times:

- M 9am
- M 1pm
- T 9am
- T 1pm

4 time slots
(best possible)

Map Coloring



Planar Four Coloring

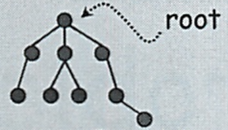
any planar map is 4-colorable.
1850's: false proof published
(was correct for 5 colors).
1970's: proof with computer
1990's: much improved

Chromatic Number

min #colors for G is
chromatic number, $\chi(G)$
lemma:

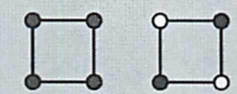
$$\boxed{\chi(\text{tree}) = 2}$$

Trees are 2-colorable

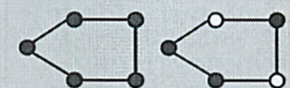


Pick any vertex as "root."
if (unique) path from root is
even length: ●
odd length: ●

Simple Cycles



$\chi(C_{even}) = 2$

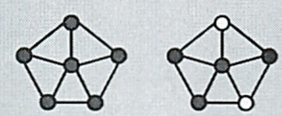


$\chi(C_{odd}) = 3$

Complete Graph K_5

$\chi(K_n) = n$

The Wheel W_n



W_5

$\chi(W_{odd}) = 4$
$\chi(W_{even}) = 3$

Bounded Degree

all degrees $\leq$ k, implies

$\chi(G) \leq k+1$

very simple algorithm...

"Greedy" Coloring

...color vertices in any order.
next vertex gets a color
different from its neighbors.

$\leq$ k neighbors, so
k+1 colors always work

coloring arbitrary graphs

2-colorable? --easy to check
3-colorable? --hard to check
(even if planar)

find $\chi(G)$? --theoretically no harder than 3-color, but harder in practice

Team Problems

Problems 1—4

6.042 Graph Conectivity + Coloring

3/30

In general graph not connected

Form groups that are connected "connected compontents"

Not all components connected to each other

Def v

$\{ w \mid v$ and w are connected $\}$

Connected if only 1 connected coomponent

Cut edge - if remove 2 vertices are no longer conected

now ≥ 2 connected components

Closed walk - walk begins + ends at same pt

- can ~~growth~~ go through pts multiple times
  - not a cycle
- need to specify begining, end, stops

Cycle - special closed walk

- lenght ≥ 2
- does not cross itself
- don't think of begining, end, or direction
- "hunk" of graph

②

Lemma: An edge is not a cut edge iff it is on a ~~simple~~ cycle

 ← can still get anywhere

- kinda follows from def

Trees - connected graph w/ no cycles
- break any edge, it falls apart
- unique ~~pa~~ path b/w any 2 points
- graph w/ 1 vertex 0 edges is a tree
- Connected graph w/ n vertices + n-1 edges
  - is proved in notes

Spanning Tree - minimal set of edges that allow everything to be connected
- ~~AAA~~ - purple subgraph on slides
- can have multiple
- Cool algebra to calc how many spanning trees are

3

Lemma G connected → G has spaning rees
- the one w/ the fewest edges is the spanning tree

Graph Coloring

- scheduling
- resolving conflicts
- how many gates are needed?

- draw edge b/w flights - on ground at same time
  - at some moment

Color the vertices so adj vertices have diff colors
- each gate diff color
- then that min # of colors is ~~an~~ min # gates needed

May not color right - his inital try had 4 colors
Did again for 3
Problem to find min # colors

Final Exam scheduling

How short an exam period can you get away w/?

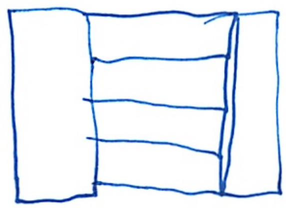
(4)

Do graph coloring

Also map coloring
- if have border - diff colors
- Corners don't count

Planar Map can always be done in 4 colors

(? Wait



Oh wait is right)

Needs 600 cases for computer to check

Min # colors for G is

Chromatic # $\chi(G)$
?chi

Trees are 2-Colorable
- One color per level
- or more abstractly distance from root
- even + odd

Cycles even lenght
$\chi = 2$

Cycles odd $\chi = 3$

(5)

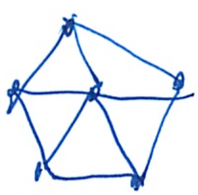
Messiest k_5 <u>Complete graph</u>

- Since every vertex adj to each one
- So 5 colors - each one is different

<u>Comple k_n</u>

$\chi(k_n) = n$

<u>Wheel W_n</u>



circle w/ axle in middle

4 colors

- 3 for odd cycle
- axel 4th color

$\chi(W_{odd}) = 4$

$\chi(W_{even}) = 3$

<u>Greedy Assignment</u>

Assign something ~~th~~ that does not conflict w/ neighbor

$\leq k$

(too fast!)

(6)

2 colorable check

- easy

3 colorable check

- very hard, Millenium prize
- even if planer
  - know 4 is enough

Can translate graphs into not, or gates if find SAT solution

$\chi(G)$

- theoretically as hard as 3-color
- pragmatically

In-Class Problems Week 8, Wed.

Problem 1.

False Claim. *If every vertex in a graph has positive degree, then the graph is connected.*

(a) Prove that this Claim is indeed false by providing a counterexample.

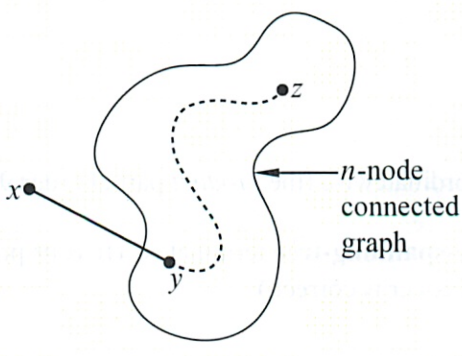
(b) Since the Claim is false, there must be an logical mistake in the following bogus proof. Pinpoint the *first* logical mistake (unjustified step) in the proof.

Bogus proof. We prove the Claim above by induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has positive degree, then the graph is connected.

Base cases: ($n \leq 2$). In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.

$P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 2$. Consider an n-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph:



All that remains is to check that there is a path from x to every other vertex z. Since x has positive degree, there is an edge from x to some other vertex, y. Thus, we can obtain a path from x to z by going from x to y and then following the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$, which proves the Claim.

■

Problem 2.

Procedure **create-spanning-tree**

Given a simple graph G, keep applying the following operations to the graph until no operation applies:

1. If an edge $\langle u\text{—}v\rangle$ of G is on a cycle, then delete $\langle u\text{—}v\rangle$.
2. If vertices u and v of G are not connected, then add the edge $\langle u\text{—}v\rangle$.

Assume the vertices of G are the integers $1, 2, \ldots, n$ for some $n \geq 2$. Procedure **create-spanning-tree** can be modeled as a state machine whose states are all possible simple graphs with vertices $1, 2, \ldots, n$. The start state is G, and the final states are the graphs on which no operation is possible.

(a) Let G be the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{\langle 1\text{—}2\rangle, \langle 3\text{—}4\rangle\}$$

What are the possible final states reachable from start state G? Draw them.

(b) Prove that any final state of must be a tree on the vertices.

(c) For any state, G', let e be the number of edges in G', c be the number of connected components it has, and s be the number of cycles. For each of the derived variables below, indicate the *strongest* of the properties that it is guaranteed to satisfy, no matter what the starting graph G is and be prepared to briefly explain your answer.

The choices for properties are: *constant*, *strictly increasing*, *strictly decreasing*, *weakly increasing*, *weakly decreasing*, *none of these*. The derived variables are

(i) e

(ii) c

(iii) s

(iv) $e - s$

(v) $c + e$

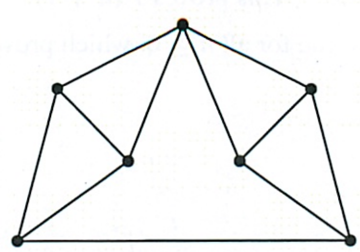
(vi) $3c + 2e$

(vii) $c + s$

(viii) (c, e), partially ordered coordinatewise (the *product* partial order 9.9.1).

(d) Prove that procedure **create-spanning-tree** terminates. (If your proof depends on one of the answers to part (c), you must prove that answer is correct.)

Problem 3.
Let G be the graph below[1]. Carefully explain why $\chi(G) = 4$.



[1]From *Discrete Mathematics*, Lovász, Pelikan, and Vesztergombi. Springer, 2003. Exercise 13.3.1

Problem 4.
A portion of a computer program consists of a sequence of calculations where the results are stored in variables, like this:

$$\begin{array}{rrcl}
 & \text{Inputs:} & & a, b \\
\text{Step 1.} & c & = & a + b \\
2. & d & = & a * c \\
3. & e & = & c + 3 \\
4. & f & = & c - e \\
5. & g & = & a + f \\
6. & h & = & f + 1 \\
 & \text{Outputs:} & & d, g, h
\end{array}$$

A computer can perform such calculations most quickly if the value of each variable is stored in a *register*, a chunk of very fast memory inside the microprocessor. Programming language compilers face the problem of assigning each variable in a program to a register. Computers usually have few registers, however, so they must be used wisely and reused often. This is called the *register allocation* problem.

In the example above, variables a and b must be assigned different registers, because they hold distinct input values. Furthermore, c and d must be assigned different registers; if they used the same one, then the value of c would be overwritten in the second step and we'd get the wrong answer in the third step. On the other hand, variables b and d may use the same register; after the first step, we no longer need b and can overwrite the register that holds its value. Also, f and h may use the same register; once $f + 1$ is evaluated in the last step, the register holding the value of f can be overwritten.(Assume that the computer carries out each step in the order listed and that each step is completed before the next is begun.)

(a) Recast the register allocation problem as a question about graph coloring. What do the vertices correspond to? Under what conditions should there be an edge between two vertices? Construct the graph corresponding to the example above.

(b) Color your graph using as few colors as you can. Call the computer's registers $R1$, $R2$, etc. Describe the assignment of variables to registers implied by your coloring. How many registers do you need?

(c) Suppose that a variable is assigned a value more than once, as in the code snippet below:

$$\begin{aligned}
&\ldots \\
t &= r + s \\
u &= t * 3 \\
t &= m - k \\
v &= t + u \\
&\ldots
\end{aligned}$$

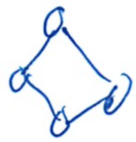
How might you cope with this complication?

InClass 8 Wed

Connected - ~~Path from every vertex to every other vertex~~

Total - ~~vert~~ edge from every vertex to every vertex

Connected - every vertex has at least one edge



1.a Prove false by counter example

i, Isn't that true

Is true - as in that must be true for connected

But exception

> 0 degree $\not\to$ Connected

Connected $\to$ > 0 degree

b) Must be logical mistake in proof

Well if keep adding a ~~line~~ point and line to last added point then it would work

But counterexample does not do this

Meyer: I want to know exactly which step it went wrong

(2)

- does not matter <u>which</u> edge you connect to in proof - this is the issue

Meyer: The QED

- no line in here that is wrong
- proving wrong thing
- are graphs w 70 degree that can't be built that way
- build up error
- Induction - think about N+1
    - break up into smaller pieces you understand
    - must be sure built every possible graph

What I put!

Lectre

---

2. Create a spanning Tree proc given simple graph G
    1. If edge (u-v) is on cycle, delete
    2. If verticies u, v not conected add <u-v>

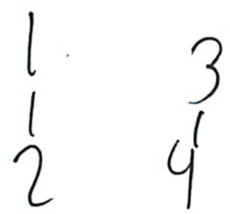
③

Assume vertices 1, 2, ... n for some $n \geq 2$

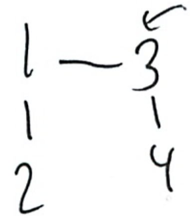
Can model as SM

- all possible graphs that can be constructed

a) Let G be ~~{1, 2, 3, 4}~~

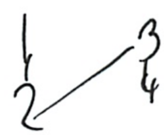


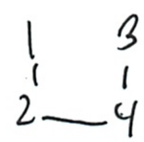
What are possible states

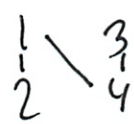


← not connected means in general connected ~~e~~ guessing or directly (ie ~~____~~ edge b/w them)

Also







b) Prove that final state must be a tree

i def of tree

Check def of tree and use

A final state reached when proc terminates

- when no cycles in graph + all vertices connected

Theirfore all final states are connected graphs w/o cycles

→ So all final states are trees

(4)

That def feels to me as cheating

c) Which property guarenteed to satisfy

- no matter starting graph

So basically - what happens to variables

e none

c weakly ↓

s " ↓

e - s " ↑

c + e " ↓

3c + 2e strictly ↓

c + s " ↓

(c, e) " "

Strictly - always ↓

Weakly - ↓ or stays same

$\underbrace{\text{stays same}}$ weakly adds this possibility

d) Prove terminates

- that one of these quantaties comes to what the def. said

c = 1 s = 0

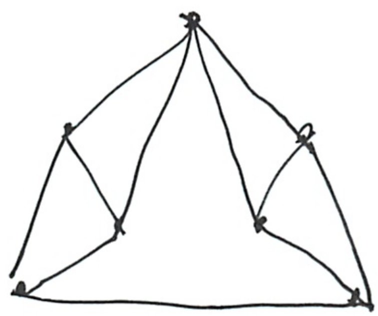
c ≥ 1 s ≥ 0 keeps ↓

\# connected components always ↓ to 1 - so must terminate

c + s strongly decreasing - one or other must go down

(5)

3. Why $\chi(G) = 4$
graph given



easy to show 4 colors

but how to prove 3 colors

¿ Like I did on P-set - show example ?

No Just proves that one did not work

Not that there could be something that works

Solutions to In-Class Problems Week 8, Wed.

Problem 1.

False Claim. *If every vertex in a graph has positive degree, then the graph is connected.*

(a) Prove that this Claim is indeed false by providing a counterexample.

Solution. There are many counterexamples; here is one:

■

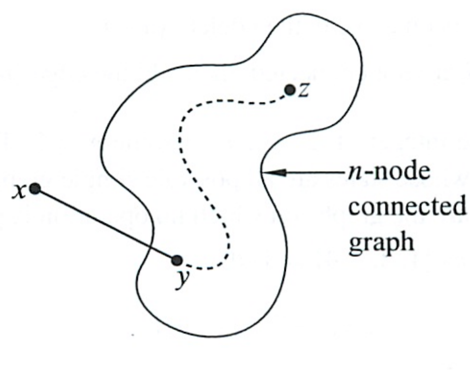
(b) Since the Claim is false, there must be an logical mistake in the following bogus proof. Pinpoint the *first* logical mistake (unjustified step) in the proof.

Bogus proof. We prove the Claim above by induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has positive degree, then the graph is connected.

Base cases: $(n \leq 2)$. In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.

$P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 2$. Consider an n-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph:



All that remains is to check that there is a path from x to every other vertex z. Since x has positive degree, there is an edge from x to some other vertex, y. Thus, we can obtain a path from x to z by going from x to y and then following the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$, which proves the Claim.

■

Solution. This one is tricky: the proof is actually a good proof of something else. The first error in the proof is only in the final statement of the inductive step: "This proves $P(n+1)$".

The issue is that to prove $P(n+1)$, *every* $(n+1)$-vertex positive-degree graph must be shown to be connected. But the proof doesn't show this. Instead, it shows that every $(n+1)$-vertex positive-degree graph *that can be built up by adding a vertex of positive degree to an n-vertex connected graph*, is connected.

The problem is that *not every* $(n+1)$-vertex positive-degree graph can be built up in this way. The counterexample above illustrates this: there is no way to build that 4-vertex positive-degree graph from a 3-vertex positive-degree graph.

More generally, this is an example of "buildup error". This error arises from a faulty assumption that every size $n+1$ graph with some property can be "built up" in some particular way from a size n graph with the same property. (This assumption is correct for some properties, but incorrect for others—such as the one in the argument above.)

One way to avoid an accidental build-up error is to use a "shrink down, grow back" process in the inductive step: start with a size $n+1$ graph, remove a vertex (or edge), apply the inductive hypothesis $P(n)$ to the smaller graph, and then add back the vertex (or edge) and argue that $P(n+1)$ holds. Let's see what would have happened if we'd tried to prove the claim above by this method:

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an $(n+1)$-vertex graph G in which every vertex has degree at least 1. Remove an arbitrary vertex v, leaving an n-vertex graph G' in which every vertex has degree... uh-oh!

The reduced graph G' might contain a vertex of degree 0, making the inductive hypothesis $P(n)$ inapplicable! We are stuck—and properly so, since the claim is false! ■

Problem 2.

Procedure **create-spanning-tree**

Given a simple graph G, keep applying the following operations to the graph until no operation applies:

1. If an edge $\langle u—v \rangle$ of G is on a cycle, then delete $\langle u—v \rangle$.
2. If vertices u and v of G are not connected, then add the edge $\langle u—v \rangle$.

Assume the vertices of G are the integers $1, 2, \ldots, n$ for some $n \geq 2$. Procedure **create-spanning-tree** can be modeled as a state machine whose states are all possible simple graphs with vertices $1, 2, \ldots, n$. The start state is G, and the final states are the graphs on which no operation is possible.

(a) Let G be the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{\langle 1—2 \rangle, \langle 3—4 \rangle\}$$

What are the possible final states reachable from start state G? Draw them.

Solution. It's not possible to delete any edge. The procedure can only add an edge connecting exactly one of vertices 1 or 2 to exactly one of vertices 3 or 4, and then terminate. So there are four possible final states. ■

(b) Prove that any final state of must be a tree on the vertices.

Solution. We use the characterization of a tree as an acyclic connected graph.

A final state must be connected, because otherwise there would be two unconnected vertices, and then a transition adding the edge between them would be possible, contradicting finality of the state.

A final state can't have a cycle, because deleting any edge on the cycle would be a possible transition. ■

(c) For any state, G', let e be the number of edges in G', c be the number of connected components it has, and s be the number of cycles. For each of the derived variables below, indicate the *strongest* of the properties that it is guaranteed to satisfy, no matter what the starting graph G is and be prepared to briefly explain your answer.

The choices for properties are: *constant*, *strictly increasing*, *strictly decreasing*, *weakly increasing*, *weakly decreasing*, *none of these*. The derived variables are

(i) e

Solution. none of these ■

(ii) c

Solution. weakly decreasing ■

(iii) s

Solution. weakly decreasing ■

(iv) $e - s$

Solution. weakly increasing ■

(v) $c + e$

Solution. weakly decreasing ■

(vi) $3c + 2e$

Solution. strictly decreasing ■

(vii) $c + s$

Solution. strictly decreasing ■

(viii) (c, e), partially ordered coordinatewise (the *product* partial order 9.9.1).

Solution. none of these ■

(d) Prove that procedure **create-spanning-tree** terminates. (If your proof depends on one of the answers to part (c), you must prove that answer is correct.)

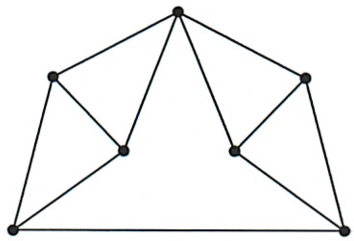
Solution. If a value (a *derived variable*) associated with a process state is nonnegative integer-valued and decreases at each step, then the process terminates after at most as many steps as the initial value of the quantity. So we need only identify such a derived variable. There are two in the list above, namely (vi) and (vii).

To show that the variable (vi) strictly decreases, note that the rule for deleting an edge ensures that the connectedness relation does not change, so neither does the number of connected components c. Meanwhile the number of edges e decreases by one when an edge is deleted. Therefore the variable $3c + 2e$ decreases by 2. The rule for adding an edge ensures that the number of connected components c decreases by one and the number of edges e increases by one. Therefore the variable $3c + 2e$ decreases by 1.

To show that the variable (vii) strictly decreases, note that the rule for deleting an edge ensures that the number of connected components c does not change and the number of cycles s decreases by n, where $n \geq 1$. Therefore the variable $c + s$ decreases by n. The rule for adding an edge ensures that the number of connected components c decreases by one and the number of cycles s does not change. Therefore the variable $c + s$ decreases by one.

■

Problem 3.
Let G be the graph below[1]. Carefully explain why $\chi(G) = 4$.



Solution. Four colors are sufficient, so $\chi(G) \leq 4$.

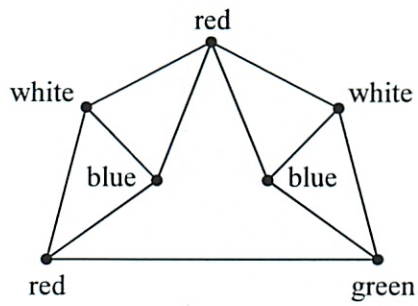


Figure 1 A 4-coloring of the Graph

Now assume $\chi(G) = 3$. We may assume the top vertex is colored red. The top two triangles require 3 colors each, and since they share the top red vertex, they must have the other two colors, white and blue, at their bases, as in Figure 1. Now the bottom two vertices are both adjacent to vertices colored white and blue, and cannot have the same color since they are adjacent, so there is no alternative but to color one with a third color and the other with a fourth color, contradicting the assumption that 3 colors are enough. Hence, $\chi(G) > 3$. This together with the coloring of Figure 1 implies that $\chi(G) = 4$. ■

[1] From *Discrete Mathematics*, Lovász, Pelikan, and Vesztergombi. Springer, 2003. Exercise 13.3.1

Problem 4.
A portion of a computer program consists of a sequence of calculations where the results are stored in variables, like this:

$$\begin{array}{rrcl} & \text{Inputs:} & & a, b \\ \text{Step 1.} & c & = & a+b \\ 2. & d & = & a * c \\ 3. & e & = & c+3 \\ 4. & f & = & c-e \\ 5. & g & = & a+f \\ 6. & h & = & f+1 \\ & \text{Outputs:} & & d, g, h \end{array}$$

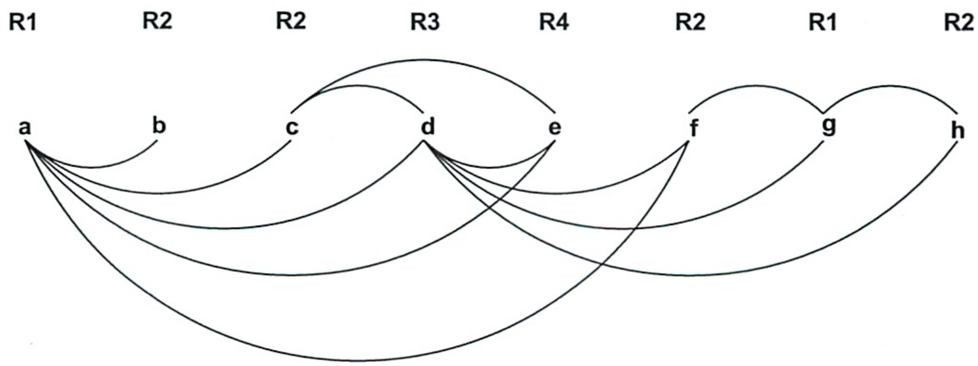
A computer can perform such calculations most quickly if the value of each variable is stored in a *register*, a chunk of very fast memory inside the microprocessor. Programming language compilers face the problem of assigning each variable in a program to a register. Computers usually have few registers, however, so they must be used wisely and reused often. This is called the *register allocation* problem.

In the example above, variables a and b must be assigned different registers, because they hold distinct input values. Furthermore, c and d must be assigned different registers; if they used the same one, then the value of c would be overwritten in the second step and we'd get the wrong answer in the third step. On the other hand, variables b and d may use the same register; after the first step, we no longer need b and can overwrite the register that holds its value. Also, f and h may use the same register; once $f+1$ is evaluated in the last step, the register holding the value of f can be overwritten.(Assume that the computer carries out each step in the order listed and that each step is completed before the next is begun.)

(a) Recast the register allocation problem as a question about graph coloring. What do the vertices correspond to? Under what conditions should there be an edge between two vertices? Construct the graph corresponding to the example above.

Solution. There is one vertex for each variable. An edge between two vertices indicates that the values of the variables must be stored in different registers.

We can classify each appearance of a variable in the program as either an *assignment* or a *use*. In particular, an appearance is an assignment if the variable is on the left side of an equation or on the "Inputs" line. An appearance of a variable is a use if the variable is on the right side of an equation or on the "Outputs" line. The *lifetime of a variable* is the segment of code extending from the initial assignment of the variable until the last use.[2] There is an edge between two variables if their lifetimes overlap. This rule generates the following graph:



■

[2]This definition is for the case that each variable is assigned at most once (see part (c)).

(b) Color your graph using as few colors as you can. Call the computer's registers $R1$, $R2$, etc. Describe the assignment of variables to registers implied by your coloring. How many registers do you need?

Solution. Four registers are needed.

One possible assignment of variables to registers is indicated in the figure above. In general, coloring a graph using the minimum number of colors is quite difficult; no efficient procedure is known. However, the register allocation problem always leads to an *interval graph*, and optimal colorings for interval graphs are always easy to find. This makes it easy for compilers to allocate a minimum number of registers. ■

(c) Suppose that a variable is assigned a value more than once, as in the code snippet below:

$$
\begin{aligned}
&\ldots \\
t &= r + s \\
u &= t * 3 \\
t &= m - k \\
v &= t + u \\
&\ldots
\end{aligned}
$$

How might you cope with this complication?

Solution. Each time a variable is reassigned, we could regard it as a completely new variable. Then we would regard the example as equivalent to the following:

$$
\begin{aligned}
&\ldots \\
t &= r + s \\
u &= t * 3 \\
t' &= m - k \\
v &= t' + u \\
&\ldots
\end{aligned}
$$

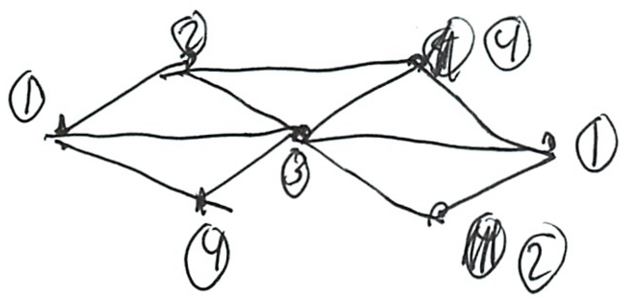
We can now proceed with graph construction and coloring as before. ■

Tutor 7 Cont

TP7.7 Coloring

Chromatic #

- need to do manually - no better way



can do ① - ② - ①

just not ① - ①

4 ⓧ

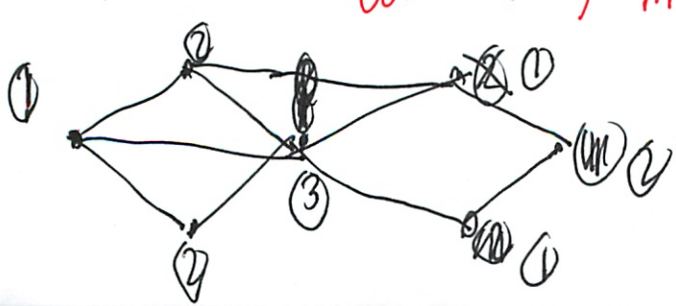
5 ⓧ

3 ✓ - triangles need 3

~~What do~~

What does it mean for colors to be sufficient?

Use two for outer rim, third for center



← could do that

(2)

TP 7.8 Which are Trees?

- ? or forests of trees?

1

2

~~3~~

4

1 2 4 X

2 4 Guess forests don't count

TP 7.9 Leaves

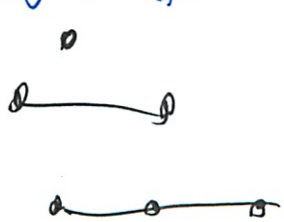
1. What is possible # of vertices where everything is leaf

leaf = degree 1

1

2

~~3~~



2 X

1,2 ✓ oh wanted all possible

(3)

b. Smallest possible # leaves in trees w/ 99 vertices

c) Largest is easy 98

- does not need to be binary

b)

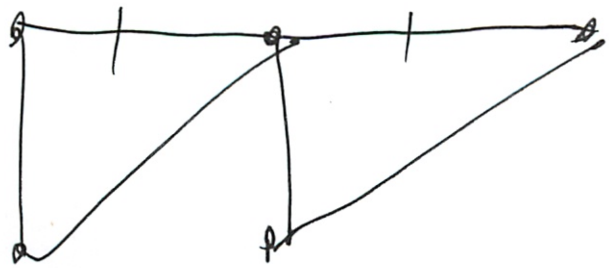
2

Tp 7.10 Graph Colloring

$\chi(\text{Tree}) = 2$

TP. 7.11 Spanning Trees

Find a spanning tree



4

TP 7.12 Graph Algorithm

Graph G

- with vertices V

  edges E

Mark edges if no path marked edges b/w

(sounds like Spanning tree)

1. Pres. Inv. and also hold for start state

   1. No
   2. ✓
   3. ~~3 ✓~~
   4. Not always

   2,3 ⊗

   2 ✓

2. Derived variables - how do they change?

   # unmarked edges - strictly ↓ ✓

   marked edges " ↑ ✓

   # unmarked edges + marked edges = constant ✓

5

marked − # unmarked

↑ − ↓

+1 − −1

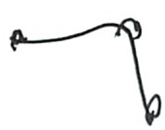
+2 strictly ↑ ✓

connected components − only marked edges

i weakly ↓ ✗

well first

then



could also

weakly ↑ ✗

but then they connect at some pt?

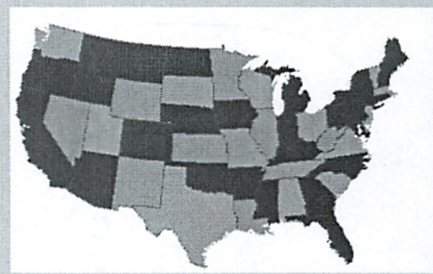
strictly ↓ ✓ * a vertice sitting by itself is a connected component *

4/1

Mathematics for Computer Science
MIT 6.042J/18.062J

Planar Graphs

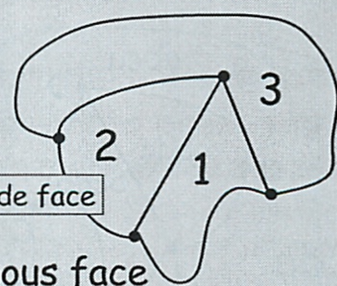
Planar Graphs



Planar Graphs

A graph is planar if there is a way to draw it in the plane without edges crossing.

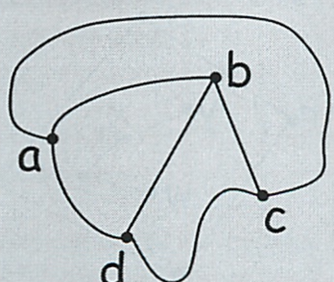
4 Continuous Faces here



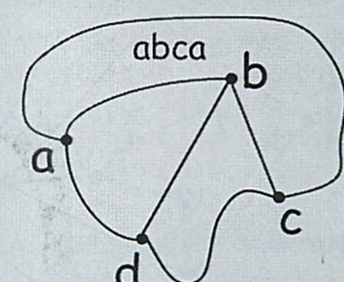
the outside face

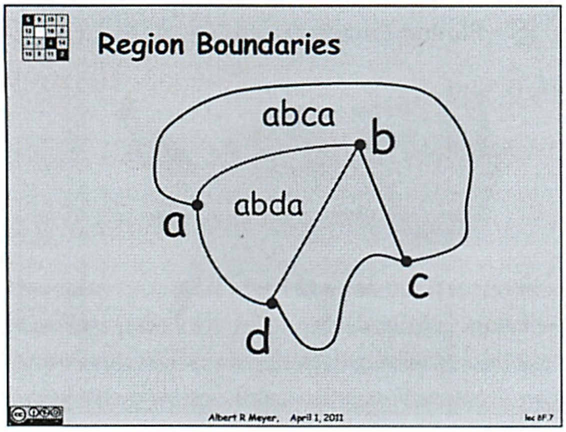
continuous face ::=connected region

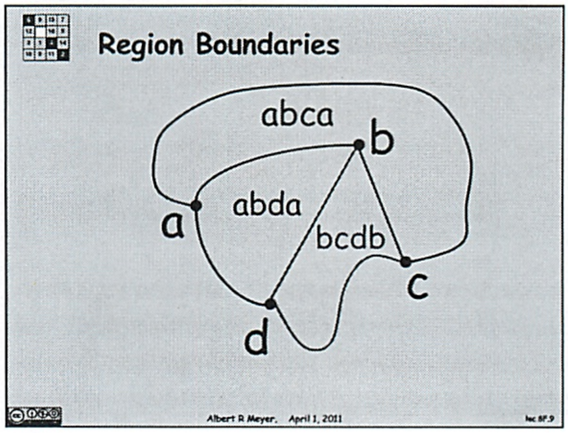
Region Boundaries

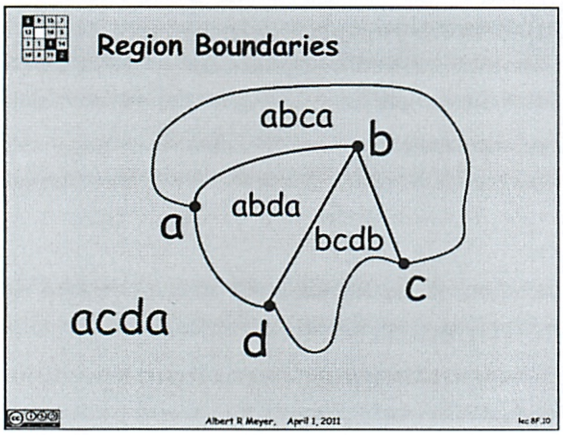


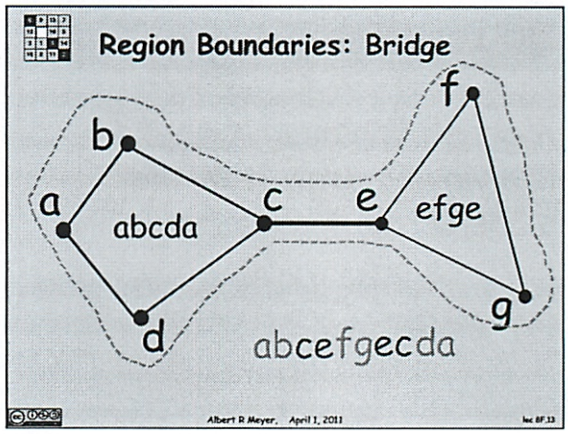
Region Boundaries

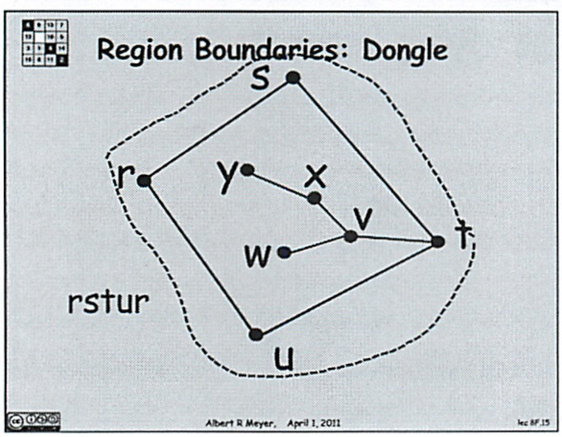


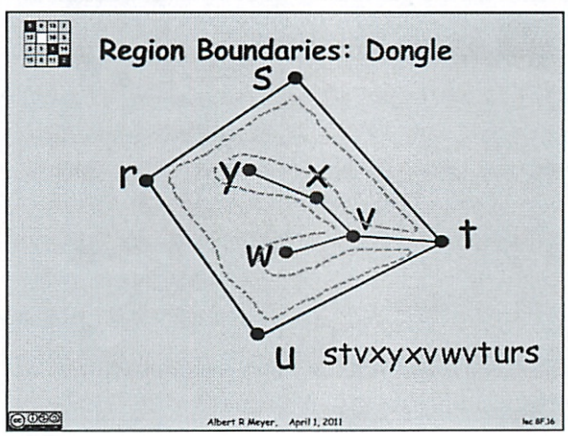

Region Boundaries
abca
b
a
abda
c
d
Albert R Meyer, April 1, 2011


Region Boundaries
abca
b
a
abda
bcdb
c
d
Albert R Meyer, April 1, 2011


Region Boundaries
abca
b
a
abda
bcdb
c
acda
d
Albert R Meyer, April 1, 2011


Region Boundaries: Bridge
f
b
a
abcda
c
e
efge
g
d
abcefgecda
Albert R Meyer, April 1, 2011

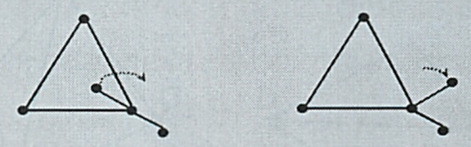

Region Boundaries: Dongle
s
r
y
x
v
t
w
rstur
u
Albert R Meyer, April 1, 2011


Region Boundaries: Dongle
s
r
y
x
v
t
w
u
stvxyxvwvturs
Albert R Meyer, April 1, 2011

Planar Embedding

A planar embedding is a connected graph *along with* its face boundary walks (same graph may have different embeddings)

Same graph, different embeddings



2 length 5 faces

length 3 face
length 7 face

Recursive Def: Planar Embeddings

Base: a graph consisting of
- single vertex, v,
- with a single face: *length 0 walk* from v to v,

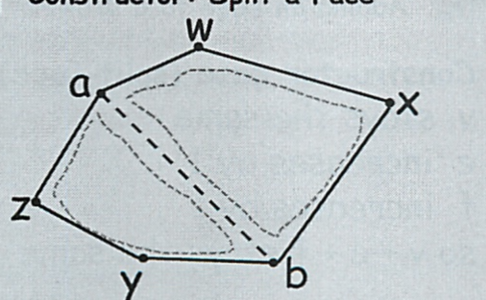
is a PE.

v●   v
graph   face

Adding an edge to an embedding

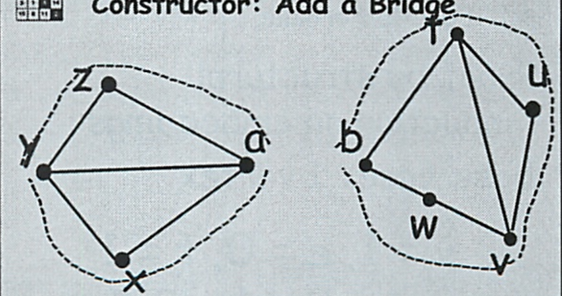
Two constructor cases:
1) Add edge across a face (splits face in two)
2) Add bridge between connected components (merges 2 outer faces)

Constructor: Split a Face



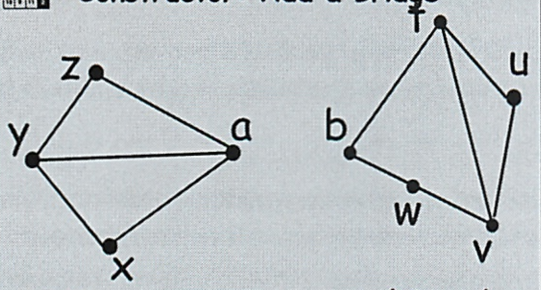
awxbyza → awxba, abyza

Constructor: Add a Bridge



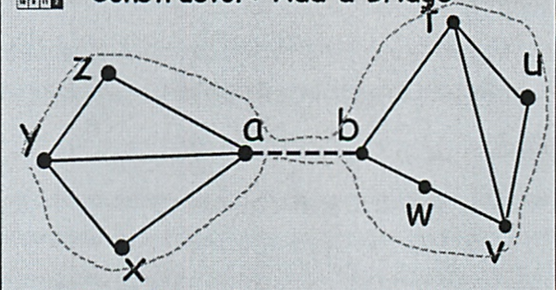
axyza   btuvwb

Constructor: Add a Bridge



axyza, btuvwb

Constructor: Add a Bridge



axyza, btuvwb $\to$ axyzabtuvwba

Team Problem

Problem 1

Euler's Formula

If a planar embedding has v vertices, e edges, and f faces, then

$$v - e + f = 2$$

Euler's Formula

Proof by structural induction on embeddings:

base case: 1 vertex

$$v = 1, e = 0, f = 1$$

$$1 - 0 + 1 = 2$$ O.K.

Adding an edge to a drawing

Constructor case (split face):

v stays the same

e increases by 1

f increases by 1

so v - e + f stays the same

O.K.

Adding an edge to a drawing

Constructor case (add bridge):

$$\begin{array}{rl} v = & v_1 + v_2 \\ -e = & -(e_1 + e_2 + 1) \\ f = & f_1 + f_2 - 1 \\ \hline 2 = & 2 + 2 - 2 \end{array}$$

(two outer faces merge into one)

O.K.

Planar Properties

- an edge appears twice on faces
- face length ≥ 3 (for $v \geq 3$)

$$3(e-v+2) = 3f \leq 2e$$

combining with Euler

$$\boxed{e \leq 3v-6}$$

Planar Properties

$$e \leq 3v-6$$

Cor: K_5 is not planar

pf: $v = 5$, $e = 10$

$$10 \not\leq 3\cdot 5 - 6$$

Planar Properties

$$e \leq 3v-6$$

Cor: Every planar graph has a vertex of degree ≤ 5

pf: suppose all degrees ≥ 6

Then

$6v \leq \sum$ degrees $= 2e \leq 6v-12$

contradiction!

Planar Properties

Cor: Every planar graph has a vertex of degree ≤ 5

Therefore,

every planar graph is 6-colorable

Euler's Formula

Cor: There are at most 5 regular polyhedra (proof in Notes)

Team Problems

Problems 2 & 3

Albert R Meyer, April 1, 2011 lec 8F.35

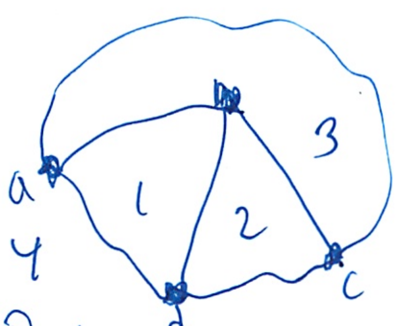
Planar Graphs

- "beautiful"
- but won't build on it later
- map = planar graph
  - vertices + edges
  - but drawn in plane w/o edges crossing each other
  - usually ~~…~~ thought of as state is vertex
  - edge b/w if they have a (+) lenght border

* can draw ~~…~~ w/o edges cross

divide up into smaller regions

- contineous faces



outside face - to ∞

(2)

But want to think about it discretly

- Seq of vertices along region

3 is abca   2 bcdb

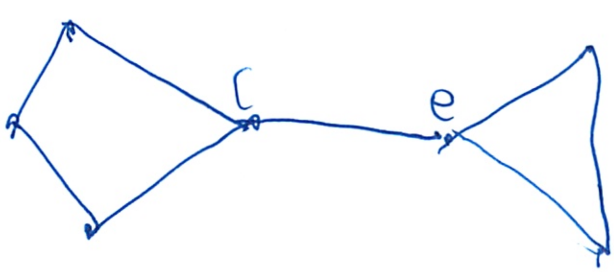
1 " abda   4 acda

does not matter where start, what dir

~~region~~

when nice - region boundries are all cycles

Sometimes <u>bridges</u>

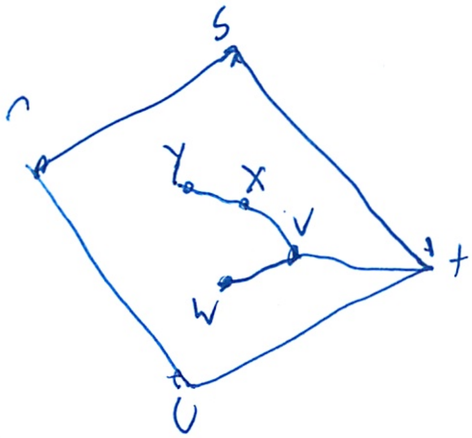


but now outer boundries are messy

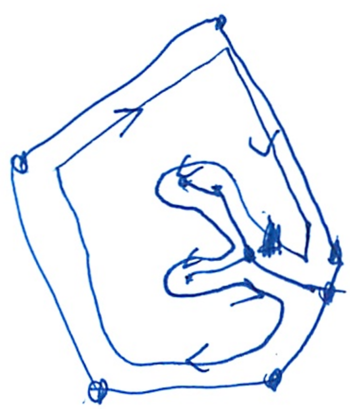
are closed walks - need to cross vertices and edges

(3)

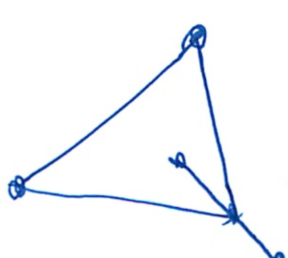
dangles



Have to backtrack
- every edge on dangle visited twice



Planar embedding - connected graph along w/ its face boundary walks

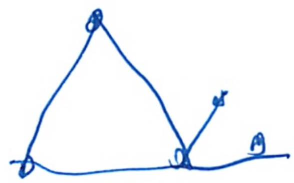


↙ 2 dangles

2 lenght 5 faces

(4)

But could puller inner dongle outside - isomorphism



lenght 3 face
7 face

but diff embeddings

Could insist tripple connected to get rid of bridges + dongles

Define ~~Plane~~ Planar Embeddings recursivly

Base

single face : lenght 0 walk v → v

Constructor

1. Add edge accross face (splits face in two)

2. Add a bridge b/w. 2 connected components

5

- which is combining 2 faces into one big face

Problem 1

Embedding - a bunch of closed walks

Now make use of def

Euler's Formula

$$\underset{\text{vertices}}{V} - \underset{\text{edges}}{e} + \underset{\text{faces}}{f} = 2$$

- is an invarient
- satisfied law of planar embeddings
  - Only embeddings have faces

Base

$V = 1$
$e = 0$
$f = 1$

$1 - 0 + 1 = 2$ ✓

6

Constructor 1 (split)

V stays same

e ↑ 1

f ↑ 1

Only one face changes → split - get one more since added edge

So $0 + 1 - 1 = 0$ so same ✓

Constructor 2 (bridge)

$V = V_1 + V_2$

V_1 ↑ first graph

V_2 ↑ 2nd graph

$e = e_1 + e_2 + 1$

$f = f_1 + f_2 - 1$

add it up

$2 + 2 - 2 = 2$ ✓

□

(7)

2 preserved planer graph propertices

- an edge appears twice on faces
- from def
- ~~for $v \geq 3$ - doesn't work for degenerate graphs~~

So total face lenght $= 2e$

- face lenghth ≥ 3 (for $v \geq 3$ - does not work degenerate)

So $3f \leq 2e$

- connect w/ Euler's theorm

and $\boxed{e \leq 3v - 6}$

Cor: K_5 is not planar

- can't drawing it is not a proof!
- but can use the invarient

$v = 5 \quad e = 10$

$10 \leq 3(5) - 6$ X

Contridiction! so ~~false~~ true

(8)

Corollary Every planar graph has a vertex of degrees ≤ 5

Pf Suppose all degrees ≥ 6

Then $6v \leq \sum$ degrees

$= 2e \leq 6v - 12$

Therefore every graph is 6-colorable

Cor There are at most 5 regular polyhedra

- in textbook
- Eulers + degree constraint

Applied result in CS