Chapter 2

2.0: Addition and Multiplication Rules

- A **set** is a collection of objects such that there are no duplicates; every object is unique
  - Ex. {0, 1, 2} ← set   {0, 1, 1} ← not a set
- A **disjoint set** are two sets which do not share any objects
  - Ex. {0, 1, 2}, {3, 4, 5} ← disjoint, because no object from either set appears in the other
  - {0, 1, 2}, {0, 1, 3} ← not disjoint because both sets contain 0 & 1
- <u>The Addition Rule (Associated with OR)</u>
  - Suppose we have *m* sets. There are r_1 different objects in the first set, r_2 different objects in the second set, …, and r_m different objects in the *m*th sets, then the number of ways to select a single object from one of the *m* sets is $r_1+r_2+...+r_m$
  - Ex. There are 5 math books, 6 history books, and 4 science books. How many ways are there to select a single book?
    - We can select 1 math book OR 1 history book OR 1 science book. Since we're using OR, we can add up our options, meaning we have 5 + 6 + 4 = 15 options.
  - Associate OR with the plus (+) sign.
- <u>The Multiplication Rule (Associated with AND)</u>
  - Same *m* sets from the addition rule above. The number of ways to select *m* objects, with one object from each of the *m* sets, is $r_1 * r_2 * ... * r_m$.
  - Ex. Same as above, except how many ways are there to select one book from each topic?
    - We can select 1 math book AND 1 history book AND 1 science book. How man Since we're using AND, we can multiply our options, meaning we have 5 * 6 * 4 = 120 options
  - Associate AND with the multiplication (*) sign
- <u>Put Them Together</u>
  - How many ways are there to select one book from only two topics?
    - Consider our options. We can choose 1 math book AND 1 history book OR 1 math book AND 1 science book OR 1 history book AND 1 science book. Combining our rules together, we get the following: 5 * 6 + 5 * 4 + 6 * 4 = 30 + 20 + 24 = 74 options

2.1: Strings

- A **string** is an ordering of objects taken from X, with the possibility than an object can be chosen more than once (*with replacement*)
    - The **length** of the string is the number of objects in the ordering
    - Ex. X = {b, a, c}, "cab" is a string because the elements were taken from the set X.
- A string is characterized by replacement and order that matters.

	Replacement	Order Matters
String	YES	YES
Permutation	NO	YES
Combination	NO	NO

- Suppose I pull objects out of X…
    - The order in which I take them out matters.
        - Ex. Take our previous set X = {a, b, c}. In strings, "ab" =/= "ba"
    - There also exists replacement
        - Ex. The string "abacaaabb" is allowed, i.e. we can reuse the objects from the set
- Formula: Given a set X containing m objects, the number of strings of length n is…

$$S(m, n) = m^n = size\ of\ set^{requested\ length}$$

    - This makes sense. Since replacement is allowed, and the order matters, at each position in the string, we will always have the size of the set # of options to choose from. Ex. If I have a set of size 5, at each position of a string, I have 5 options. Using the multiplication rule, we can say that "I choose a AND I choose b AND I choose …"

2.2: Permutations

- A **permutation** is a string with the constraint that it is built **without replacement**. That is, an object can't be chosen for the permutation more than once

	Replacement	Order Matters
String	YES	YES
Permutation	NO	YES
Combination	NO	NO

- Suppose I pull objects out of X…
    - The order in which I take them out matters.
        - Ex. ab =/= ba
    - Replacement does NOT exist
        - Ex. abaa is **not** a permutation
- Formula: Given a set X containing m objects, the number of permutations of length n is

$$P(m, n) = \frac{m!}{(m-n)!}$$

- Ex. Given the set X = {a, b, c}, how many permutations of length 3 can we make?
    - Let's start with our options approach and then we can do the formula. For our first spot, we have 3 options to choose from. In our second spot, since replacement is a no-go, we have 2 options to choose from. In our last spot, we then only have 1 option left. Therefore, we have 3 * 2 * 1 options, or 3! options. Now, if we use the formula, we get $\frac{3!}{(3-3)!} = \frac{3!}{0!} = \frac{3!}{1} = 3!$
- Example questions. Let X = {J, A, C, K, E, T, S}.
    - How many permutations of length four are there? $\frac{7!}{(7-4)!}$
    - How many permutations of length six? $\frac{7!}{(7-6)!}$
    - How many permutations of length six are there if the first letter in the permutation is a vowel? $2 * \frac{6!}{(6-5)!}$
        - We have 2 options for our first letter. Since there's no replacement, we now only have 6 options left from our set. For the remaining permutation, we say we have a set of length 6 (since we use a vowel at the beginning), and there are only 5 spots remaining.
    - How many permutations of length six are there if the first and fourth letters must be a vowel? $2 * \frac{5!}{(5-2)!} * 1 * \frac{3!}{(3-2)!}$

2.3: Combinations

- A **combination** is an unordered collection of objects from a set X. In other words, a combination is a permutation, except we forgot about the order of the objects.

	Replacement	Order Matters
String	YES	YES
Permutation	NO	YES
Combination	NO	NO

- Suppose I pull objects out of X…
  - The order in which I take them out DOES NOT matter.
    - Ex. ab = ba
  - Replacement does NOT exist
    - Ex. abaa is **not** a combination
- <u>Formula</u>: If X is a set of size n, then the number of size k combinations we can make from X is…

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{P(n,k)}{P(k,k)}$$

- Example..
  - Suppose $X = \{a, b, c\}$. How many combinations of size 2 are there? $\binom{3}{2} = \frac{3!}{2!(3-2)!} = \frac{3*2*1}{2*1*1} = 3$ which is correct because our combinations are {a, b}, {a, c}, {b, c}
  - Same set X. How many combinations of size 3 are there? $\binom{3}{3} = \frac{3!}{3!(3-3)!} = 1$ which is correct. There can only be one combination that can exist when $k = n$, and it's just the set X itself, which is {a, b, c}. Remember: {b, a, c} is not unique as a combination because order *doesn't* matter for combinations
- Example 2…
  - You are packing for a vacation. Available for packing are nine shirts, five pairs of pants, and seven pairs of socks.
    - How many ways are there to pack three shirts, two pairs of pants, and four pairs of socks?
      $\binom{9}{3} * \binom{5}{2} * \binom{7}{4}$. Writing this out in English, we can say "I choose 3 shirts AND choose 2 pants AND 4 socks. Remember that via the multiplication rule, AND means multiplication, so we'll be multiplying no matter what. Since the order of the shirts we choose doesn't matter (me choosing 2 red shirts and 1 blue shirt is no different than me choosing 1 blue shirt and then 2 red shirts), we have a combination. Thus, it's simply a matter of multiplying each combination together to get the number we're looking for.

        - You also have the option of taking a hat. You may choose from one of three hats, or choose not to take a hat. How many ways are there to pack the same number of shirts, pants, and socks as in part (a), now with the hat option added?
          $\binom{9}{3} * \binom{5}{2} * \binom{7}{4} * 4$. Remember to not jump immediately into applying formulas; think about our *choices* here. We can choose one of three hats, but we also have the option to not take a hat, so then we really have four opinions: hat_1, hat_2, hat_3, and no_hat. Furthermore, it's another AND option; shirts AND pants AND socks AND hat.
- Example 3…
    - How many ways are there to rearrange the letters in the word "banana"?
        - Seems like a permutation problem, right? No replacement and ordering matters, so permutation. The problem though is that we have multiple of the same letter. 2 n;s and 3 a's. Let's start it out like this: we have 6 slots for letters… _ _ _ _ _ _. For each spot, we can put any of the letters in "banana." We can *choose* 3 blank spaces for A's AND *choose* 2 blank spaces for N's AND choose 1 place for B. $\binom{6}{3}\binom{3}{2}\binom{1}{1}$

2.4: Combinatorial Proofs

- Typically in algebra when we want to show an equation is true, we show that the left hand side equals the right hand side
- In a **combinatorial proof**, we instead show that the two sides of the equation count the same thing.
- Strategy
    - 1) Look at the simpler side of the equation and see what it counts (i.e. compare it to strings, permutations, and combinations to see which one it's closely related to)
    - 2) Go to the other side of the equation and ask yourself "can we break it down into cases that match up with the summands on the other side of the equals sign"
- Example…
    - Show that

$$\sum_{k=0}^{m} \binom{m}{k} \binom{n}{r-k} = \binom{m+n}{r}.$$

Note that the top of the summation should be *r* instead of *m*. The right hand side indicates to us that we are counting the # of ways to choose *r* objects from a set of size $m + n$.

2.5: The Ubiquitous Nature of Binomial Coefficients

I) Counting Integer Solutions to Equations

- Formula: Given an equation

$$x_1 + x_2 + ... + x_{n-1} + x_n = M,$$

The number of integer solutions to the equation where $x_1, x_2, ..., x_{n-1}, x_n \geq 1$ is

$$\binom{M-1}{n-1}$$

- M is the objects to distribute, meaning there are M - 1 spaces between objects
- *n* bins/people to distribute the objects to, therefore we have n - 1 dividers to place
- Although the above formula looks incredibly generic and therefore not very useful, we can take steps to turn other problems into a problem of the form above.
    - Ex. How many integer solutions are there to $x_1 + x_2 + x_3 = 16$ where $x_1 \geq 2, x_2 \geq 4, x_3 \geq 1$

      To solve this, transform the problem into one such that it has the exact same answer and all the x_i's are all >= 1. In order to do this transformation, we need to "take away" or "predistribute" the unit from *M*, making it *M - 1*, and "give it" to a certain object x_i. We do this until all the constraints are at >= 1.

      $x_1 \geq 2$: take 1 object and place it in the x_1. Now, $x_1 + x_2 + x_3 = 15$ with constraints $x_1 \geq 1,\ x_2 \geq 4,\ x_3 \geq 1$

      $x_2 \geq 4$: take 3 objects and place them in x_2. Now, $x_1 + x_2 + x_3 = 12$ with constraints $x_i \geq 1$

      Now we are in the format where we can use the formula, so the answer is $\binom{11}{2}$.
- What happens when we have constraints that are less than 1?
    - Ex. How many integer solutions are there to the equation $x_1 + x_2 + x_3 + x_4 = 20$, where $x_i \geq 0$?

      To solve this, we need to "take out a loan." We need to take away objects from the bins so that it follows the format of the formula. Therefore, we need to "take" 1 item from each bin, meaning we take 4 objects. But since we have to "give the loan back", it gets added to the stack of things to distribute, meaning we now have 24 objects to distribute.

      $x_1 + x_2 + x_3 + x_4 = 24$, where $x_i \geq 1$, meaning our answer is $\binom{23}{3}$
- Extra Formula (shortcut): Given an equation

$$x_1 + x_2 + ... + x_{n-1} + x_n = M,$$

The number of integer solutions to the equation where $x_1, x_2, ..., x_{n-1}, x_n \geq 0$ is

$$\binom{M+n-1}{n-1}$$

- What about if there is an upper bound constraint for x_i ($x_i <= 10$)?
    - Ex. How many integer solutions are there to the equation $x_1 + x_2 + x_3 + x_4 = 36$ if $x_1, x_2, x_3, x_4 \geq 0$ and $x_4 \leq 10$

      Strategy: (Compute #, pretend upper bound constraint doesn't exist) - (# of solutions that violate the x_4 constraint)

      Ignoring the upper bound, all x_i have a bound of 0. Borrow one from each variable, we get $\binom{36+4-1}{3} = \binom{39}{3}$. However, we are not done. We must now find the # of solutions that violate the constraint. Since $x_4 <= 10$ is the upper bound, we are in trouble if $x_4 \geq 11$. Therefore, we'll say we allocate 11 to x_4, so any number we find is a case where x_4 is breaking its upper bound. To allocate, subtract 11 from 36, meaning we are distributing 25 objects in this case. Now, all of our bins must follow $x_1, x_2, x_3, x_4 \geq 0$ only. Using the formula, we'd get $\binom{25+4-1}{3} = \binom{28}{3}$. This is the number of cases that break the upper bound constraint. Therefore, the number of ways to count this is $\binom{39}{3} - \binom{28}{3}$
- Lastly, what happens when our equation is an inequality?
    - How many integer solutions are there to $x_1 + x_2 + x_3 + x_4 + x_5 \leq 42$, where $x_1, x_2, x_3, x_4, x_5 \geq 0$?

      To solve these, add an extra variable x_6 that will work to "hold" all the remaining objects not distributed to the other objects.

      $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 42$

      $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$

      NOTE: the constraint for this extra variable will always be $x_i \geq 0$

II) Counting Lattice Paths

- A **lattice path** is a path on a coordinate plane in R^2 composed of dots where each dot is length 1 away from each other, the path can only go up or right, and each line segment is either horizontal or vertical.
- Formula: The number of lattice paths from (a, b) to (c, d) is

$$\binom{(c-a)+(d-b)}{(c-a)}$$

- This formula can be used because going from one point to the next, for instance, (0,0) to (6,4), they will always have the same number of horizontal and vertical segments. In our example, we will always have 6 horizontal segments and 4 vertical segments
- Another way of looking at the formula is…

$$\binom{total\ \#\ of\ line\ segments}{\#\ of\ vertical\ line\ segments/\#\ of\ horizontal\ line\ segments}$$

2.6: The Binomial Theorem

- Theorem: The Binomial Theorem

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

- Example..
    - Multiply out $(x + y)^3$, but leave the order of the variables in the terms the same (xxx+xxy+xyx+xyy+yxx+yxy+yyx+yyy) = xxx+xxy+xyx+yxx+xyy+yxy+gyyx+yyy = $\binom{3}{3}x^3 + \binom{3}{2}x^2y + \binom{3}{1}xy^2 + \binom{3}{0}y^3$

2.7: Multinomial Coefficients

- Formula: Suppose we have n objects and r subsets, and we want to put k_1 objects in subset $r = 1$, and k_2 objects in subset $r = 2$, etc. We can utilize something called the **multinomial coefficient**, which is just a shorthand way of writing binomial coefficients multiplied together, and the equation for this is as follows…

$$\binom{n}{k_1, k_2, k_3 ..., k_r} = \frac{n!}{k_1! k_2! k_3! ... k_r!}$$

  - Ex. Suppose there are 16 people you want to assign into three teams. One team should have 7 people, the second team should have 5, and the third team should have 4. How many different ways can you assign the people into teams?

$$\binom{16}{7, 5, 4} = \frac{16!}{7!5!4!}$$

- Theorem: The Multinomial Theorem

$$(x_1 + x_2 + ... + x_r)^n = \sum_{k_1 + k_2 + ... + k_r = n} \binom{n}{k_1, k_2, ..., k_r} x^{k_1} x^{k_2} ... x^{k_r}$$

- Suppose we have n people, and the number of people we need is $< n$ (i.e. we have 16 people but only need three groups of 3). To solve this, we just pretend we have a fourth team where the remaining number of people go, so in this instance the answer would be $\binom{16}{3,3,3,7}$.

Chapter 3

3.1: Introduction and the Positive Integers are Well Ordered

- Theorem: *The positive integers $Z^+ := \{1, 2, 3, \ldots\}$ are **well-ordered**, meaning that any subset X of Z^+, even an infinite subset, will have a smallest integer.*
    - Note that the set of **all** integers $Z := \{..., -2, -1, 0, 1, 2, \ldots\}$ is **NOT** well-ordered

3.3: The Meaning of Statements

- This chapter pertains to the idea of "sequences of concepts," which is like a family of definitions, formulas, or true-false statements, each organized and corresponding to its own positive integer.
    - The could be formulas, e.g. factorials:

Positive Integer	1	2	3	4
Factorial	1!	2!	3!	4!

    - The could be definitions, e.g. prime numbers:
    - They could be counts of objects, e.g. the numbers of (0, 1)-strings:

Positive Integers	1	2	3	4
# of (0,1)-strings	2^1	$2^2 = 4$	$2^3 = 8$	$2^4 = 16$

    - They could be true-false statements, e.g. for any positive integer n…

Positive Integers	1	2	3
$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$	$1 = \frac{1(1+1)}{2}$	$1 + 2 = \frac{2(2+1)}{2}$	$1 + 2 + 3 = \frac{3(3+1)}{2}$

- Thus far, we have discussed factorials in the sense that

$$n! = n * (n - 1) * (n - 2) *...$$

- However, we can just as easily describe it recursively, as a *recursive definition*

$$\boxed{n! := \begin{cases} 1, & \text{if } n = 1, \\ n * (n-1)!, & \text{if } n > 1. \end{cases}}$$

    - Note there are two important parts of the recursive definition, the **base case** and the "**step up**"
- If we start from the base case and move up, we can use our knowledge of prior step-ups to get to our goal

- Ex. We know 1! = 1, so then solving for 2! it's 2 * 1!, but we know 1!, so it's 2 * 1. Now we know 2!. To solve for 3!, it's just 3 * 2!, and we know 2!, and so on

---

3.4 & 3.5: Binomial Coefficients Revisited & Solving Combinatorial Problems Recursively

- Consider our previously discussed *binomial coefficient*…

$$\boxed{\binom{n}{k} := \frac{n!}{(n-k)!k!}.}$$

- Using the same idea of recursion previously, we can redefine the binomial coefficient recursively…

$$\boxed{\binom{n}{k} := \begin{cases} 1, & \text{if } k = 0, \\ 1, & \text{if } n = k, \\ \binom{n-1}{k} + \binom{n-1}{k-1}, & \text{if } n > 1. \end{cases}}$$

- Although it may seem more complicated at first, in some cases using a recursive definition is easier to solve certain problems.
  - Ex. How many ternary strings of length *n* do not contain 201 (2, then 0, then 1)

n	1	2	3
T(*n*)	T(1)	T(2)	T(3)

  - First, look at T(*n*). Are we able to define T(*n*) in terms of T(*n* - 1), T(*n* - 2), …?
  - Three cases…
    - Case 0: the last digit is 0

__ __ __ __ … 0, n - 1 spaces here: populate the first n-1 spaces with a (0, 1, 2) string w/o 102 and of length n-1. **S(*n* - 1) * 1**

- Case 1: the last digit is 1

__ __ __ __ …1, exactly the same as Case 0. **S(n - 1) * 1**

- Case 2: the last digit is 2

__ __ __ __ … 2, *not* the same as the other cases, since the previous 2 characters can be 1 and 0. We need the string of length *n* -1 with no 102, and the last two digits are not 1 and 0 in that order. **S(n - 1) - # of (n-1) length strings that end in 10 => __ __ __ __ __** … 1 0 => with the 1 and 0 it's n - 1 spaces, without them its n - 3 spaces => **S(n - 3),** so overall its **S(n - 1) - S(n - 3)**

Case 0: S(n - 1)
Case 1: S(n - 1)
Case 2: S(n - 1) - S(n - 3)

Recursive Step: S(n) = 3S(n - 1) - S(n - 3)

Since we have n - 3, we need S(1), S(2), and S(3) as base cases.

$S(1) = 3^1$, $S(2) = 3^2$, $S(3) = 3^3 - 1$ (subtract the case where our string is 102)

3.6: Mathematical Induction

- **Induction** is used to prove things, and it's very similar to the idea of recursion.
- We need to prove these statements (assuming they *are* true) in two steps…
  - The **base case**: prove that the statement is true for the case $n = 1$
  - The **inductive step**: assume the statement is true for a particular n. We call this an *inductive hypothesis*. We then prove that the statement is true for $n + 1$.
- Ex. Prove that for every n > 0, the following holds…

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

  - Base Case: n = 1, $\sum_{k=1}^{1} 1 = \frac{1(1+1)}{2} = \frac{2}{2} = 1$
    - Proved that the base case is true. Next, it's the inductive step…
  - Inductive Step: Assume true for some n. $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$.

    Show that it's true for $n + 1$. $\sum_{k=1}^{n+1} k = \frac{(n+1)((n+1)+1)}{2}$

$$\sum_{k+1}^{n+1} k... = \sum_{k=1}^{n} k + (n + 1)$$

Note that now we have something come up that it's in the form of our *inductive hypothesis,* so we can plug that in to help us.

$$= \frac{n(n+1)}{2} + (n + 1) = \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+2)(n+1)}{2}$$

- Ex. Use mathematical induction prove that for all n >= 1, we have the $n^3 + (n + 1)^3 + (n + 2)^3$ is divisible by 9.

First, let's prove that the following holds true for n = 1.

$1 + 2^3 + 3^3 = 36$, which is divisible by 9. Now, let's assume that the equation

$n^3 + (n + 1)^3 + (n + 2)^3$ for some n. Let's prove it for $n + 1$.

$(n + 1)^3 + (n + 2)^3 + (n + 3)^3 = 9b$ where b is some integer

Notice that we have a match between some of our terms, $(n+1)^3$ and $(n+2)^3$. In that case, let's expand our $(n+3)^3$ term to get that final n^3 so we can substitute

$(n + 1)^3 + (n + 2)^3 + (n^3 + \binom{3}{1} n^2 * 3 + \binom{3}{2} n * 3^3 + \binom{3}{3} * 3^3)$

$(n + 1)^3 + (n + 2)^3 + n^3 + 9n^2 + 27n + 27$

By inductive hypothesis…

$9b + 9(n^2 + 3n + 3)$

We have two numbers being added together that are both divisible by 9, and two numbers that are both divisible by one number, their sum will also be divisible by that number, in this case it's 9.

3.9: Strong Induction

- Sometimes, the inductive step require more than just assuming the (k - 1)th case is true. We need to assume that the 1st *through* (k - 1)th cases are all true. This is **strong induction**.
  - Example…

Exercise 5.1 (Exercise 3.17 from the textbook). Consider the recursive function given by

$$f(n) := \begin{cases} 2, & \text{if } n = 0, \\ 4, & \text{if } n = 1, \\ 2f(n-1) - f(n-2) + 6, & \text{if } n \geq 2. \end{cases}$$

Prove that $f(n) = 3n^2 - n + 2$ for all $n \geq 0$.

Assume that f(n) =/= $3n^2 - n + 2$ for all n. So there is some smallest number in this set, call it m. So for all $n < m$, it must be that $f(n) = 3n^2 - n + 2$

Start at $f(m) = 2f(m-1) - f(m-2) + 6$ and we are going try and show that in fact $f(m) = 3m^2 - m + 2$

For $n < m$, $f(n) = 3n^2 - n + 2$

So $f(m-1) = 3(m-2)^2 - (m-1) + 2$

$f(m-2) = 3(m-2)^2 - (m-2) + 2$

$f(m) = 2(3(m-1)^2 - (m-1) + 2) - (3(m-2)^2 - (m-2) + 2) + 6$

We just inserted our definitions of f(m - 1) & f(m - 2) in the highlighted eqn.

If you were to multiply this out and simplify, you would get…

$3m^2 - m + 2$

<u>Verify Base Cases</u>

$f(0) = 3(0)^2 - (0) + 2 = 2$

$f(1) = 3(1)^2 - (1) + 2 = 4$

Chapter 4

4.1: Pigeon Hole Principle

- The **Pigeon Hole Principle** states that if we are comparing two sets, and one is bigger than the other, if you set up a correspondence where you try to correspond one element from one set to another element in another set, eventually have to "double up" because there's not enough to go around
- "If we have two sets, X and Y, we are trying to pair up each element $x \in X$ with an element $y \in Y$, then if $|X| > |Y|$ there must be two different x's in X that are associated to the same $y \in Y$"
  - Example: There are n married couples. How many people must we choose from the $2n$ to guarantee that of the people chosen, two are a married couple?
    - $n + 1$; say we have 4 married couples. In the worst case, we choose 1 from each couple every time. However, once we have chosen 4 people, we have 1 person from each couple in the worst case. Therefore, once we pick one more person, we can guarantee that that 5th person will be a married couple with a previously chosen person.
  - Example: In a round-robin tournament, show that there must be two players with the same number of wins if no player loses all matches. (In a round-robin tournament, each player plays all other players exactly once)
    - Assume we have n players. Each player must play n - 1 matches (because of round-robin tournament style). We have two sets, one of size n and one of size n - 1. We have to correspond a number of wins to each player (once again, because of round-robin). So if we have two sets of different sizes, and must correspond one element from 1 set to another. Therefore, via the Pigeon Hole Principle, two players must have the same number of wins.

Exam 1 Study Guide

Quizlet link [here]

- Addition & Multiplication Rule
  - *Addition Rule*: OR is equivalent to +
  - *Multiplication Rule*: AND is equivalent to *
- Strings, Permutations, and Combinations

	Replacement	Order Matters
String	YES	YES
Permutation	NO	YES
Combination	NO	NO

  - String Formula: $S(m,n) = m^n = size\ of\ set^{requested\ length}$
  - Permutation Formula: $P(m,n) = \frac{m!}{(m-n)!}$
  - Combination Formula: $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{P(n,k)}{P(k,k)}$
- Counting Integer Solutions
  - Formulas
    - $x \geq 1$Formula: $\binom{M-1}{n-1}$
    - $x \geq 0$Formula: $\binom{M+n-1}{n-1}$
  - Constraints & How to Deal w/ Them
    - Lower constraint $x_i \geq$: "predistribute" from M ([ex.])
    - Upper constraint $x_i \leq$: (Compute #, pretend upper bound constraint doesn't exist) - (# of solutions that violate the constraint) ([ex.])
    - Inequality instead of equation: Add an additional set x_j that will take the remaining items left over after distribution. Adding x_j turns it from an inequality to an equation. ([ex.])
- Lattice Paths
  - A **lattice path** is a path on a coordinate plane in R^2 composed of dots where each dot is length 1 away from each other, the path can only go up or right, and each line segment is either horizontal or vertical.
  - Formula: The number of lattice paths from (*a*, *b*) to (*c*, *d*) is…

$$\binom{(c-a)+(d-b)}{(c-a)}$$

- [Combinatorial Proofs]
- Binomial Theorem
  - The **binomial theorem** describes the expansion of powers of a binomial, and can be used to determine the particular coefficient of a specific term.

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

- Multinomial Coefficients
  - A **multinomial coefficient** is a short-hand way of writing binomial coefficients multiplied together. (Ex.)

$$\binom{n}{k_1, k_2, k_3 \ldots, k_r} = \frac{n!}{k_1! k_2! k_3! \ldots k_r!}$$

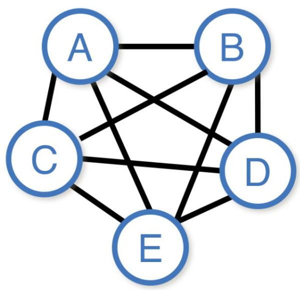
- Recursion
- Induction
- Pigeon Hole Principle

Chapter 5

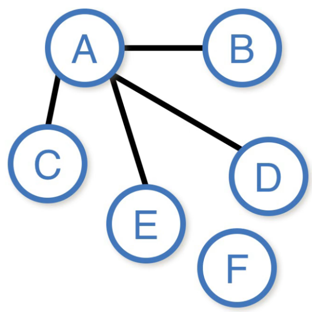
5.1: Basic Notation & Terminology for Graphs

- A **graph** G is a mathematical object consisting of two sets…
  - a **vertex** set $V(G)$, and
  - an **edge** set $E(G)$ consisting of unordered pairs of elements of the vertex set
    - *Unordered* because the edge AB is equivalent to the edge BA
- The elements of $V(G)$ are called **vertices** (**vertex** for singular) and the elements of $E(G)$ are called **edges**
- Simply put, a graph is points on a plane, where those points may or may not be connected to other points by a line.
- Example…
  - $V(G) = \{a, b, c, d, e\}$
  - $E(G) = \{(a, b), (b, c), (b, d), (c, d)\}$
- For now, assume that our graphs do not contain the following…
  - Loops (an edge that connects a vertex back to itself)
  - Multiple Edges (two or more edges from one point to another)
  - Directed edges (an edge where you can only travel in one direction)
- Terminology for Vertices
  - Two vertices x and y of a graph G are **adjacent** (**neighbors**) if there is an edge (x, y) i.e. an edge that connects the two.
  - The **degree** of a vertex x, given by the notation $d(x)$, is the number of edges that connect to x.
    - If a vertex has a degree of 1, we call that vertex a **leaf**
- Types of Graphs
  - A graph H is a **subgraph** of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$
    - A subgraph H is a **spanning subgraph** if $V(H) = V(G)$ (i.e. a subgraph that contains all the vertices, but not necessarily all the edges, of the original graph)
    - An **induced subgraph** is a subgraph where if two vertices are present and the original graph has an edge connecting them, they must also have an edge in the subgraph
      - So, say we have the vertices A, B, C and edges AB, BC, AC. In a *subgraph*, we could have vertices A, B and no edges. In a *spanning subgraph*, we would need to have all vertices A, B, C, but we still are not required to have any edges. In an *induced subgraph*, we can have vertices A, B, but because the original graph has edge AB, we are required to have the edge AB here because of the definition of an *induced subgraph*

- Sequences of Vertices
  - A **walk** is a sequence of vertices such that consecutive vertices x_i and x_{i+1} are connected by an edge.
  - A **path** is a walk that visits a vertex no more than once
  - A **cycle** is a path where you cannot use the same edge more than once & the starting and ending vertices are the same
- Connectivity
  - A graph G is **connected** if for any two vertices there exists a walk between them (i.e. there exists a series of edges you can traverse to get from that vertex to the other)
    - NOTE: A graph of a single vertex is, by default, connected
  - A graph G is **disconnected** if for two vertices, there does *not* exist a walk between them
    - A **component** C of a graph G is a subgraph of the disconnected graph that does not contain the other disconnected portion



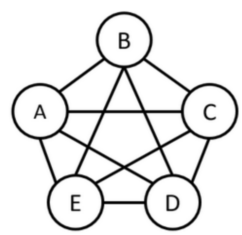
Dense, Connected Graph



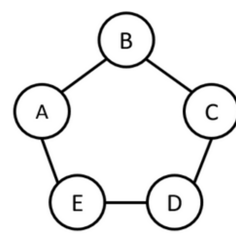
Sparse, Disconnected Graph

(Ignore the terminology *dense* and *sparse*, although you can probably figure out what it means just by looking at it. Also, we can say vertex F is it's own *component* whilst vertices and edges A,B,C,D,E is it's own *component*)

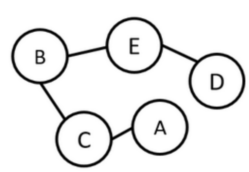
- Cycles & Trees
  - A graph is **acyclic** if it does not contain any cycles
    - An acyclic graph is a **tree** if it is *connected*
    - An acyclic graph is a **forest** if it is *disconnected*
      - A subgraph H is a **spanning tree** of graph G if it is both a *spanning subgraph* of G and a *tree*
- FIGURE: Ignore the Petersen Graph.
- Completeness & Independence
  - The **complete graph** on n vertices, with the notation K_n, is a graph with n vertices and an edge between every pair of vertices



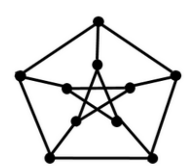
Complete Graph K_5



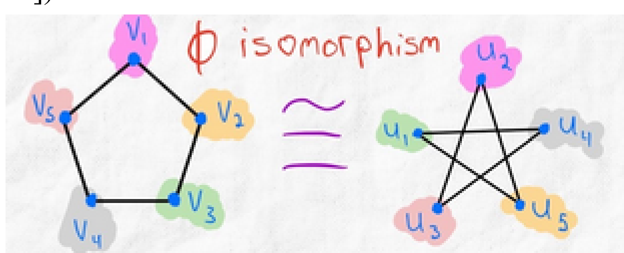
Cycle Graph C_5



Tree (Acyclic, Connected)


Petersen Graph

    - Note that in a complete graph, each vertex needs a degree of n - 1
  - The **independent graph** on n vertices, with the notation I_n, is the graph with n vertices and no edges
- Bijections
  - A **bijection** between two sets X and Y is a function f: X => Y that satisfies the following properties…
    - it is **surjective**, meaning every element in Y has some element that it is mapped to in X (i.e. everyone has a partner)
    - it is **injective**, meaning that no two elements in a single set are mapped to the same element (i.e. no one has the same partner)
- Graph Isomorphism
  - A **graph isomorphism** f between graphs G and H, usually given by the notation $f : G \cong H$ is a bijection $f: V(G) \to V(H)$ between the vertex sets of G and H that also satisfies the following property…
    - For any two vertices x and y in $V(G)$, we have that $f(x)$ and $f(y)$ are adjacent if and only if x and y are adjacent
  - NOTE: A graph is always isomorphic to itself, and it follows the transitive & symmetric property (i.e. if A is isomorphic to B and B is isomorphic to C, A is isomorphic to C [transitive], and if A is isomorphic to B, B is isomorphic to A [symmetric])



  - How to Show Graphs are Not an Isomorphism
    - We need to show that any bijection won't preserve adjacency. In other words, we need to show it is impossible to create an isomorphism.
    - Quick ways to show graphs aren't isomorphic…
      - Different # of vertices
      - Different # of edges
      - Different # of components
      - One is a tree, the other is not (i.e. one has a cycle, the other doesn't)
      - [Bestest one; use this] Vertices have different degree *sequences* (listing the degrees from biggest to smallest)
        - Ex. (4, 2, 2, 2, 1, 1) vs. (3, 3, 2, 2, 1, 1): Evidently not an isomorphism because they have different degree sequences

- Exercises
  - Draw a graph with 6 vertices having degrees 5, 4, 4, 2, 1, and 1, or explain why such a graph does not exist
    - Summing up the degrees, 5 + 4 + 4 + 2 + 1 + 1 = 17, we get an odd number. A sum of the degrees of a graph must be even; it is impossible otherwise. This is because the sum of degrees of a graph $G = 2 * |E(G)|$.
  - Suppose G is a graph where every vertex has degree 3. If there are 18 edges in G, how many vertices are there?
    - Sum of degrees = 2 * |E(G)|
    - 3 * |V(G)| = 2 * 18, solve for |V(G)|
  - Prove that if G is a tree, then removing any edge will disconnect G.
    - (1) Assume for a contradiction that there is some edge whose removal will not disconnect the tree G. (2) So there is a walk, and thus a path, between x and y that doesn't use the edge xy. (3) So when we include xy at the end of the path, that creates a cycle. (4) By definition, tree's cannot have cycle, and therefore to maintain that definition there must only be that one edge between x and y.
- Complements
  - If G is a graph, its **complement** G^* is the graph that has the same vertices of G, but has an edge between two vertices a and b if and only if there is not an edge between a and b in G. In other words, for each possible edge, if it existed in G, it doesn't exist in G^*, and if it did not exist in G, it exists in G^*
  - Prove that if G is disconnected, then G^* must be connected.
    - Take two disconnected vertices a and b that we will say make up graph G. Then, in the complement, there must be an edge between a and b, making them connected. No matter how many vertices there are, if there is one disconnected vertex, in the complement that vertex will connect to every other vertex that it's not connected to

5.3: Eulerian and Hamiltonian Graphs

- Recall that a **walk** is a sequence of vertices such that there is an edge between pairs of consecutive vertices. If we edit this definition, we can get more types of walks…
    - A **path** is a walk that does not repeat edges or vertices
    - A **cycle** does not repeat edges and does not repeat vertices and the starting vertex and the ending vertex are the same
    - A **trail** does not repeat edges but *can* repeat vertices
    - A **circuit** does not repeat edges but can repeat vertices, and the starting vertex is the same as the ending vertex (it's a trail that is also a cycle)
- <u>Eulerian Circuits</u>
    - A **Eulerian circuit** is a circuit that uses *every* edge exactly once. A graph is **Eulerian** if it contains no isolated vertices and has a Eulerian circuit
        - A graph G is Eulerian if, and only if, it is connected *and* every vertex has an even degree
- <u>Hamiltonian Cycle</u>
    - A **Hamiltonian cycle** is a cycle that visits *every* vertex exactly once. A graph G is **Hamiltonian** if it has a Hamiltonian cycle.
        - If G is a graph on n vertices and each vertex in G has at least n/2 neighbors, then G is hamiltonian
- <u>Eulerian Trail</u>
    - A **Eulerian trail** is very similar to a *Eulerian circuit* with the only different than you are *not* required to start and end on the same vertex
        - Prove that a graph has a Eulerian trail if and only if it is (1) connected and (2) has at most 2 vertices of odd degree.
            - Case 0: The graph is connected with 0 vertices of odd degree. If we have no vertices of odd degree, then all of our vertices are of even degree. By the definition of a Eulerian circuit, our graph contains a Eulerian circuit. Since a circuit is a type of trail, our Eulerian circuit is our Eulerian trail
            - Case 1: The graph has 1 vertex of odd degree. This is not possible, because the sum of degrees must be even.
            - Case 2: The graph has 2 vertices of odd degree. Add an extra edge between two vertices of odd degree. Those two vertices, and thus all vertices, have even degree. So we have a Eulerian circuit. Remove the edge that was added, and we have a Eulerian trail

5.4: Graph Coloring

- A **coloring** of graph *G* is a function from the vertex set of *G* to a set of elements *C*. (Usually we set *C* to be the set of integers from 1 to *n*, so the coloring looks like $\phi: V(G) \to \{1, 2, ..., n\}$.)
- A **proper coloring** of *G* is a *coloring* of *G* where no two adjacent vertices have the same colors.
    - That is, if $x, y \in V(G)$, then *x* adjacent to *y* implies $\phi(x) \neq \phi(y)$.
- We say that *G* is *k*-**colorable** if it has a proper coloring $\phi: V(G) \to C$ where *C* has size *k*.
- The **chromatic number** $\chi(G)$ of *G* is the smallest integer *k* for which *G* is *k*-colorable
    - NOTE: The number of vertices in *G* does not impose a constraint on how high the chromatic number must be. Neither does the degree of the vertex.
        - In other words, the number of vertices nor the degrees of the vertices impose a lower-bound constraint on the chromatic number. *However*, the number of vertices imposes an upper-bound on the chromatic number, which should be obvious since we can't have more colors than we have vertices.
    - NOTE: The chromatic number can get arbitrarily high. That is, for any positive integer *n* we can find a graph *G* whose chromatic number is *n*.

Cliques & Bipartites

- A **clique** of a graph *G* is a subset of the vertices whose induced subgraph is a complete graph.
    - In other words, a *clique* is a set of vertices such that any two vertices are adjacent.
    - The **clique number** $\omega(G)$ is the size of the largest clique in *G*.
    - NOTE: The clique number acts as a lower bound for the chromatic number.
- A graph is **bipartite** if its vertex set *V(G)* can be partitioned into two subsets *X* and *Y*, such that every edge of *G* has one endpoint in *X* and the other endpoint in *Y*
    - A **complete bipartite graph** is a bipartite graph that includes all possible edges between vertices in *X* and vertices in *Y*, denoted $K_{|X|,|Y|}$
    - NOTE: Being bipartite is the *same thing* as being 2-colorable.
    - THEOREM: A graph is bipartite if and only if it does not contain an odd cycle.

Approaches to Coloring

- *First-Fit*: Take a sequence of colors. At each vertex, start from the first color in the sequence, and check to see if that vertex can be colored with that color. If not, go to the second color and check. If not, go to the third color and so on. Once colored, go to the next numbered vertex (not exclusively following the edges, just go to the next node e.g. go from vertex 1 to vertex 2) and repeat.

    - Greedy algorithm. Ensures a proper coloring but does **not** ensure it's coloring will give the chromatic number. Puts an upper-bound on the chromatic number.
        - CROSSOVER: If you've taken/are taking CS 4240 (Compilers & Interpreters), you'd recognize that this is actually just *Chaitin's Algorithm* which is used for the register allocation step in the back-end of compilers.
- *Odd Cycles and Cliques*: Using knowledge of cliques and odd cycles, check to see if we can go any lower than the upper-bound set by the first-fit greedy algorithm.

Intersection & Interval Graphs

- If we have a family of of sets $\{S_\alpha\}_{\alpha \in 1}$, the **intersection** graph is the graph where each vertex corresponds to a set S_α, and two vertices are adjacent if and only if their intersection isn't empty
    - In other words, if we have a bunch of sets, where each set represents a node on a graph, the *intersection graph* is a graph where its edges connect two sets that share elements.
    - NOTE: For any graph *G*, one can find a family of sets for which G is the intersection graph.
    - An **interval graph** is an intersection graph where the sets in question are closed intervals of the real number line R.

5.5: Planar Graphs

- A graph is **planar** if it can be drawn in the plane R^2 without any edges crossing. More precisely, the only time two edges meet each other is a vertex

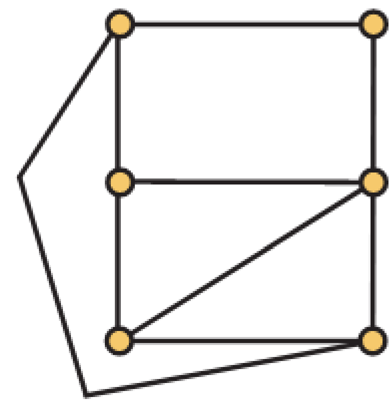
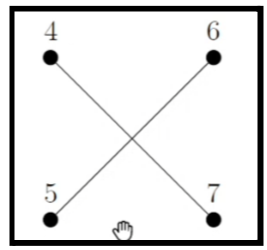


Figure 5.30. A planar drawing of a graph

  - FIGURE: Take a look at the figure to the right. This is **also** a planar graph. Why? Take a very close look at the definition: "A graph is planar ***if*** it can be drawn…" Although the graph in its current incarnation is not planar, it is still planar because it is *possible* to draw it in a planar manner, provided we either rearrange the vertices or maneuver the edges in specific ways.



- EQUATION: In a planar graph G, the equation…

$$n - m + f = 2$$

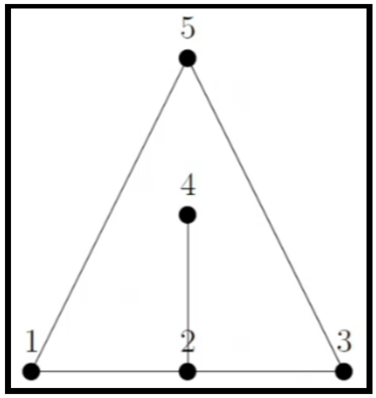
where,

n is the number of vertices of G

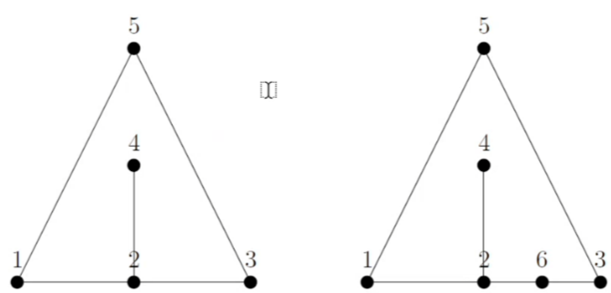
m is the number of edges of G

f is the number of *faces* of G

  - So, to test if a graph isn't planar, we can use this equation to demonstrate that a graph is, in fact, not planar
- A **face** of a planar graph G is a region of R that G divides R^2 into when G is drawn without any edge crossings
  - When considering the concept of *faces*, think of the plane R^2 like a flat sheet of cookie dough, and the graph as a cookie cutter. When imprinting our graph (cookie cutter) into R^2 (the cookie dough), what we have should be the number of regions on the inside of the graph + the region outside the graph, the stuff the cookie cutter didn't cut.
  - For example, in the below graph, there are 2 faces, one on the inside of the graph and one of the outside

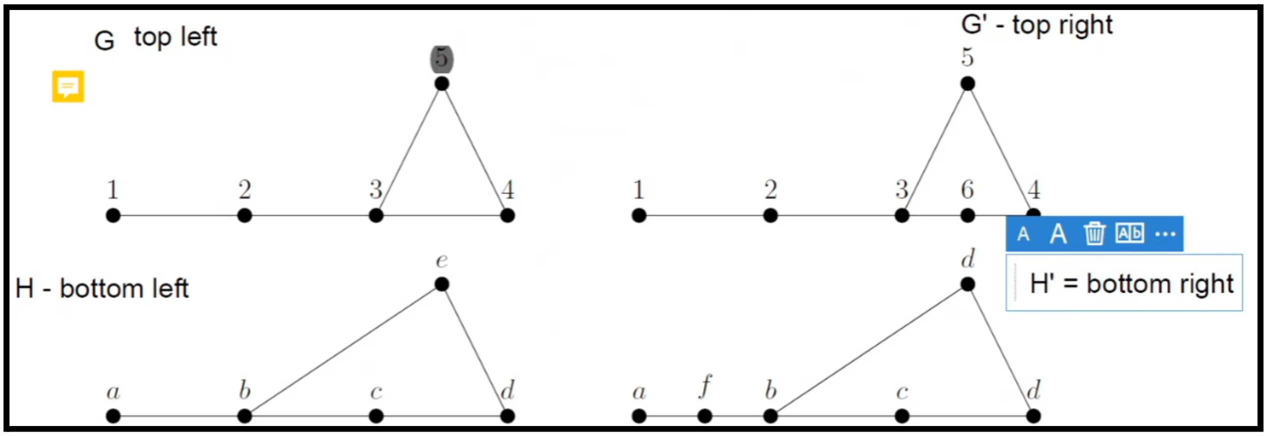


- By utilizing this definition, you'll notice that it satisfies Euler's formula. 5 - 5 + 2 = 2
- THEOREM: If G is a planar graph with n vertices and m edges, then $m \leq 3n - 6$
  - Ex. Is K_5, the complete graph on 5 vertices, planar?
    - Let's test to see if our inequality holds. m is the number of edges, and since our graph is K_5, that means $m = \binom{5}{2} = 10$, which means our inequality is $10 \leq 3(5) - 6 \Rightarrow 10 \leq 9$. Since the inequality doesn't hold, it must be that K_5 is not planar.
  - NOTE: This theory is useful because it gives us a *sufficient* condition, however it does **not** give a *necessary* condition. For example, even though $K_{3,3}$ doesn't fail the above theorem's requirement, it is **not** planar.
    - In other words, this theory is enough to prove that something **is not** planar, but it is not enough to prove that something **is** planar.
- Keep in mind these ideas of K_5 and $K_{3,3}$, we'll use them in a second.
- If G is a graph, then an **elementary subdivision** G' of G is a graph G' that is formed by "splitting" an edge into two. We take an edge xy, and place a vertex z in the middle of the edge. Now, we have two edges, xz and zy



  - NOTE: The elementary subdivision is *the entire graph*, not just that small area where we add the vertex 6.
  - NOTE: You can repeat the process of splitting edges as many times as you want; you're not limited to doing it once.

- Two graphs *G* and *H* are **homeomorphic** if there is an elementary subdivision *G'* of *G* that is isomorphic to an elementary subdivision *H'* of *H*

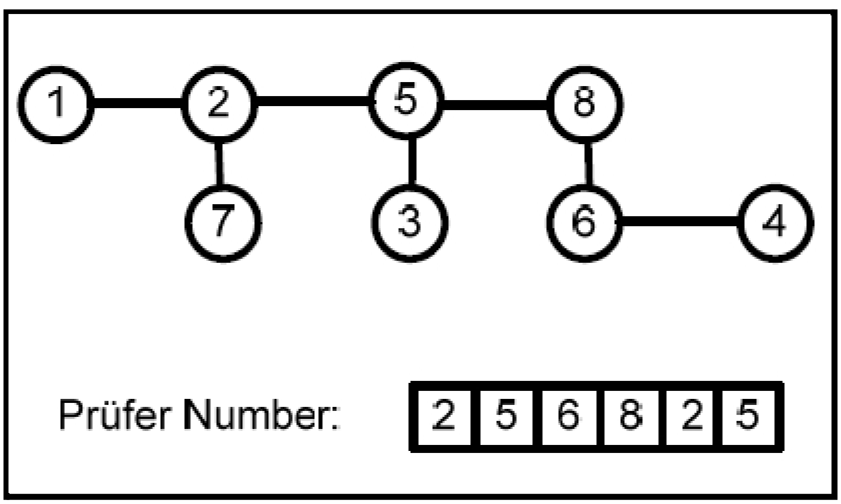


- THEOREM (Kuratowski's Theorem): A graph is planar if, and only if, it does not contain a subgraph *homeomorphic* to either K_5 or $K_{3,3}$
    - So look at your graph, and look for $K_{3,3}$ or K_5. If it contains them, it's not planar. Otherwise, it is.
- THEOREM (Four Color Theorem): If *G* is a planar graph, then the chromatic number $\chi(G) \leq 4$

5.6: Counting Labeled Trees

- This chapter will deal with counting labeled trees, as in how many types of structures can we create by taking a tree with n vertices and labeling each vertex with a unique integer from $\{1, 2, \ldots, n\}$
- We say two labeled trees are the **same** if their adjacency lists are the same. That is, if we draw out a list of all the vertices and write out each of their neighbors, if two of those lists are the same, we say the labeled trees are also the same
  - It's important to note that with this stipulation, the number of trees we can make is not simply a combination or permutation with input n.
- THEOREM (Cayley's Formula): The number of labeled trees on n vertices, where $n \geq 2$, is precisely n^{n-2}
- DEFINITION: The **Prüfer code** *P(T)* of a labeled tree *T* is defined recursively as follows…
  - *Base Case*: If *T* is the tree on two vertices, then *P(T)* is the empty string.
  - *Recursive Step*: If *P(T)* is a tree on $n \geq 2$ vertices, find the leaf *T* whose label *v* is smaller than the label of any other leaf in *T* (find the smallest leaf). Let *u* be the label of the vertex connected to the leaf with label *v*. Then…

$$P(T) = \{u, P(T \setminus \{v\})\}$$

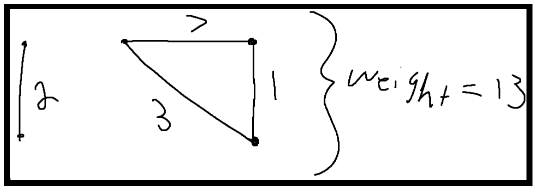


  - NOTE: Prüfer codes uniquely identify labeled trees. That is, if two trees have the same prüfer code, they must be the same. If two trees have different prüfer codes, they are not the same.
- If prüfer codes unique identify a labeled tree, it should be possible then to devise the labeled tree from the prüfer code
  - Ex. see professor's annotated example

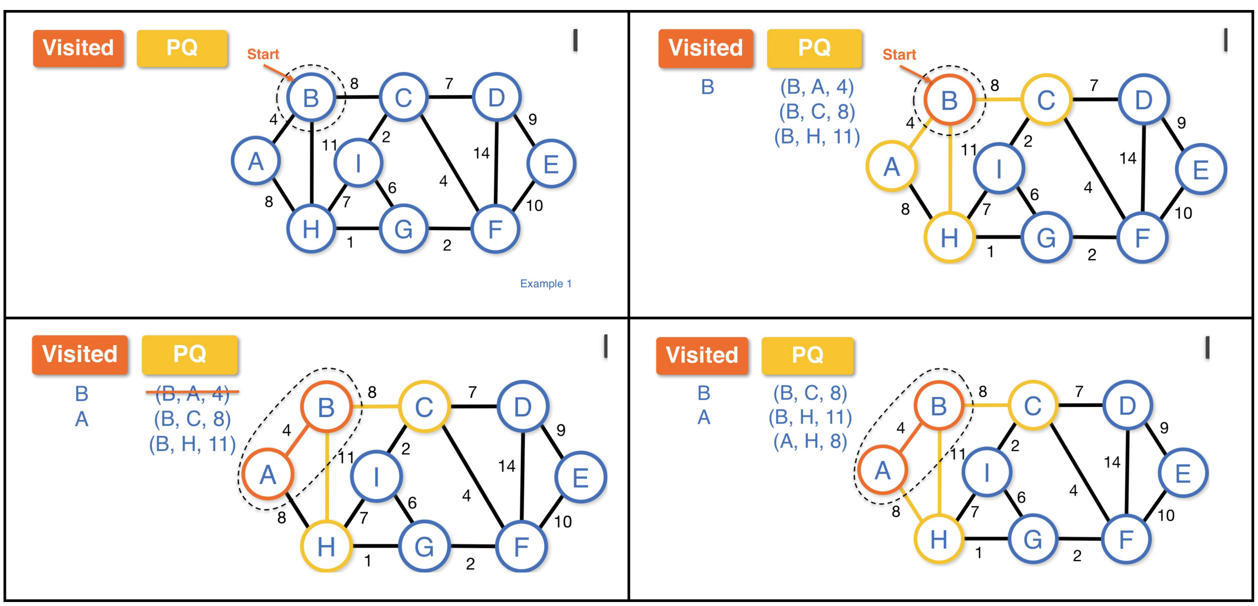
Chapter 12

12.1: Graph Algorithms

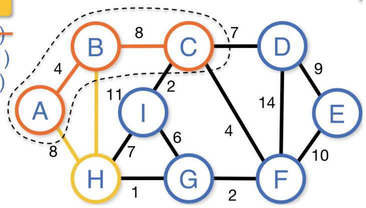
- DEFINITION: A **weighted graph** is a graph *G* in which every edge has been assigned a nonnegative integer, where each edge's respective integer is the **weight** of that edge.
    - NOTE: Weights do not have to be unique. That is, multiple edges can have the same weight. Furthermore, weights can be 0.
    - NOTE: The weight of a graph is simply the sum of the weights of all it's edges.



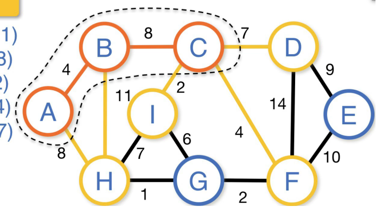
- Recall that a *spanning tree T* of a graph *G* is a subgraph of *G* that is both a tree (connected w/ no cycles) and a spanning subgraph of *G* (every vertex of *G* is included in *T*).
- With this definition in mind, given a weighted graph *G*, a **minimum weight spanning tree** (**minimum spanning tree**) *T* of *G* is a spanning tree of *G* such that its total weight is minimal. There are two algorithms for finding these…
    - *Prim's Algorithm* (chad algorithm)
        - The idea behind Prim's algorithm is something called the *graph cut*, which is an invisible line that surrounds all of the vertices. When deciding to choose your next vertex, it will simply be the next edge with the smallest weight that does not create a cycle



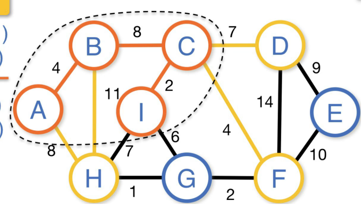
Visited	PQ
B	~~(B, C, 8)~~
A	(B, H, 11)
C	(A, H, 8)



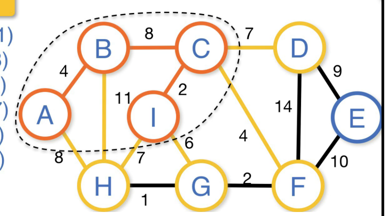
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, I, 2)
	(C, F, 4)
	(C, D, 7)



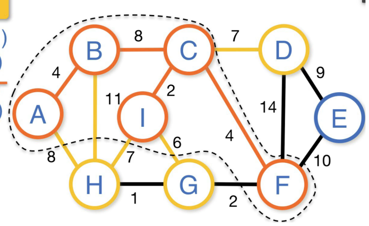
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	~~(C, I, 2)~~
I	(C, F, 4)
	(C, D, 7)



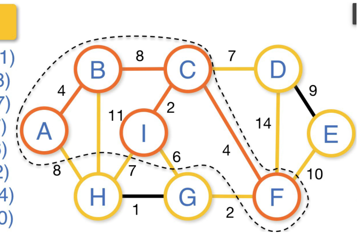
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, F, 4)
I	(C, D, 7)
	(I, H, 7)
	(I, G, 6)



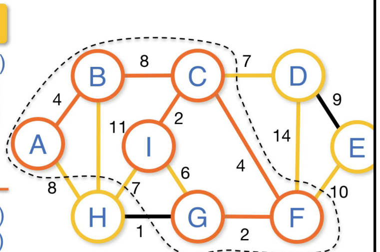
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	~~(C, F, 4)~~
I	(C, D, 7)
F	(I, H, 7)
	(I, G, 6)



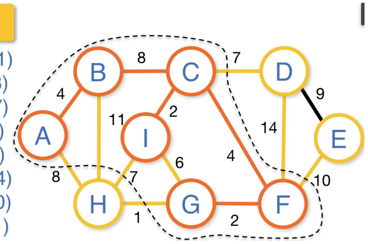
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, D, 7)
I	(I, H, 7)
F	(I, G, 6)
	(F, G, 2)
	(F, D, 14)
	(F, E, 10)



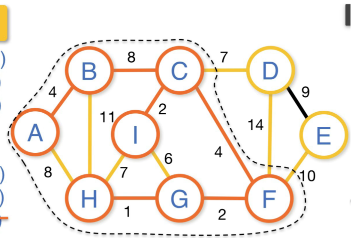
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, D, 7)
I	(I, H, 7)
F	(I, G, 6)
G	~~(F, G, 2)~~
	(F, D, 14)
	(F, E, 10)



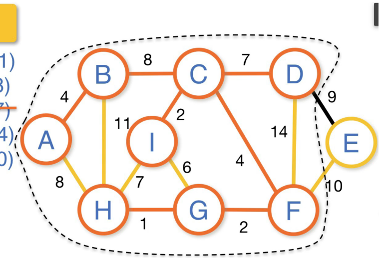
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, D, 7)
I	(I, H, 7)
F	(I, G, 6)
G	(F, D, 14)
	(F, E, 10)
	(G, H, 1)

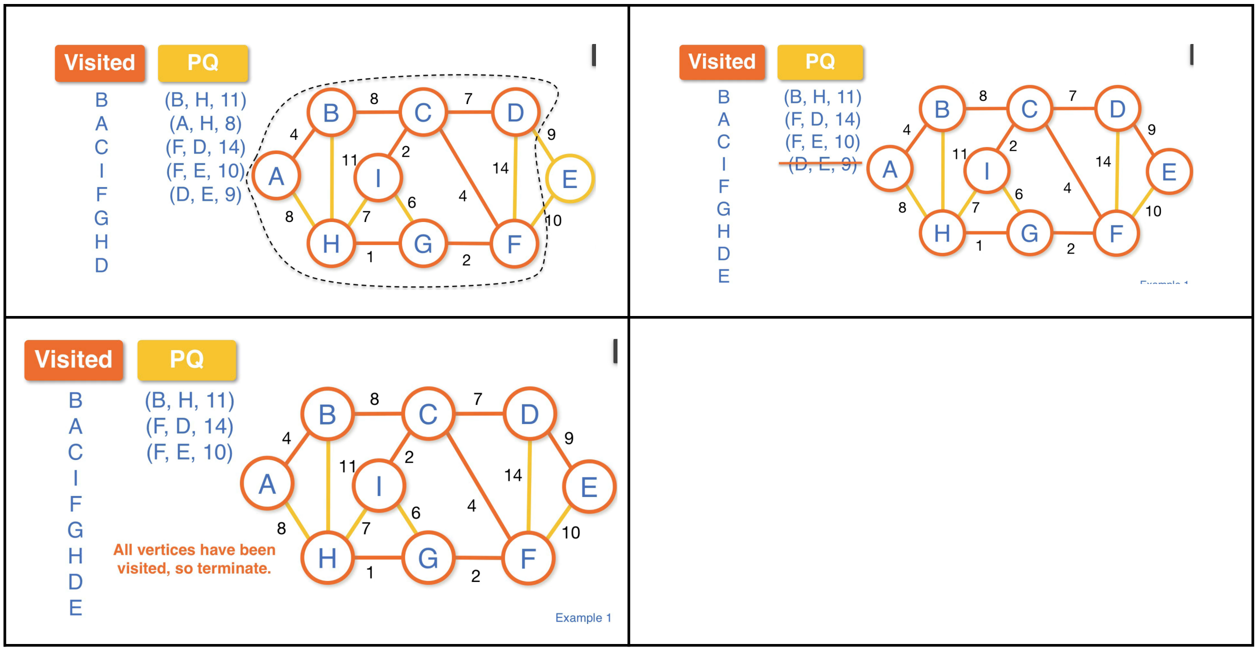


Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	(C, D, 7)
I	(I, H, 7)
F	(I, G, 6)
G	(F, D, 14)
H	(F, E, 10)
	~~(G, H, 1)~~

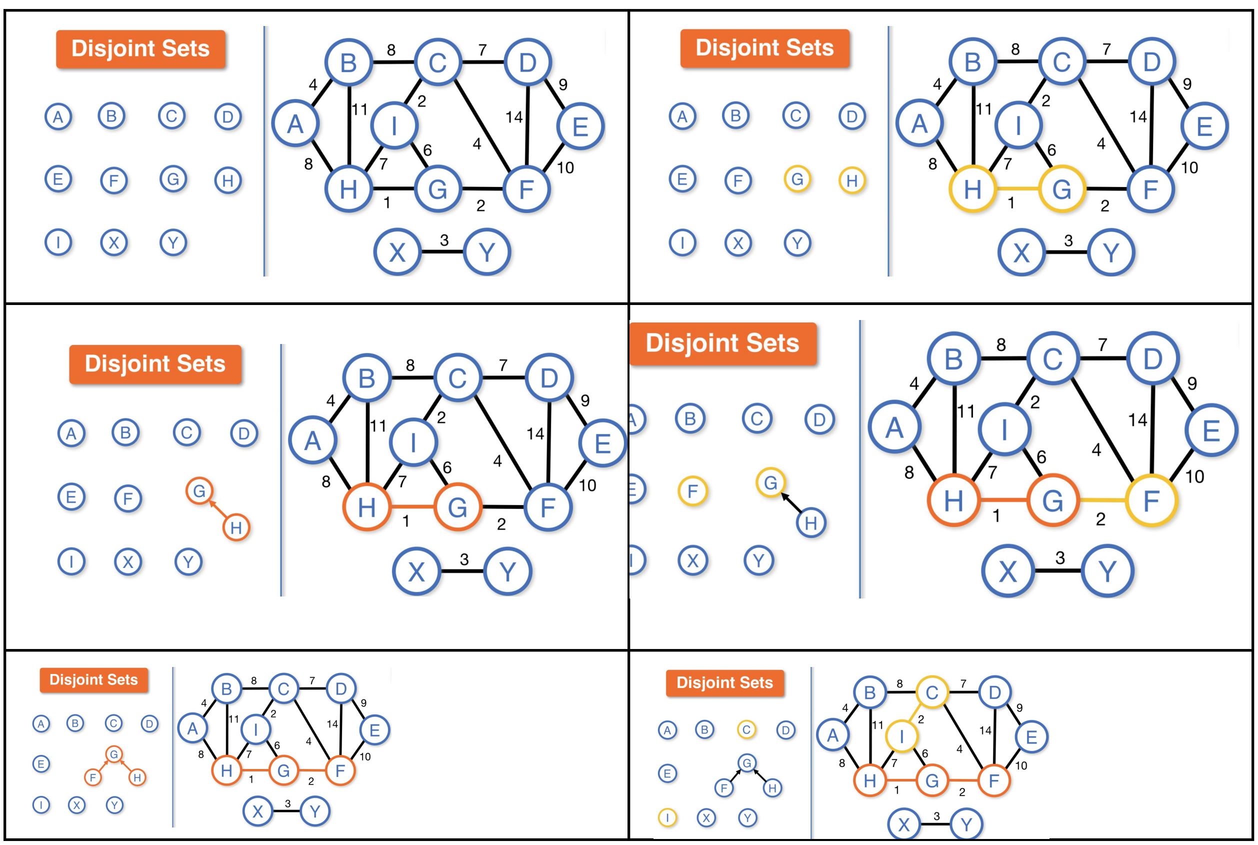


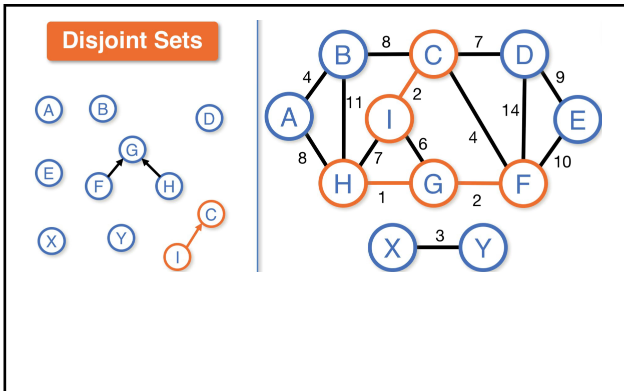
Visited	PQ
B	(B, H, 11)
A	(A, H, 8)
C	~~(C, D, 7)~~
I	(F, D, 14)
F	(F, E, 10)
G	
H	
D	

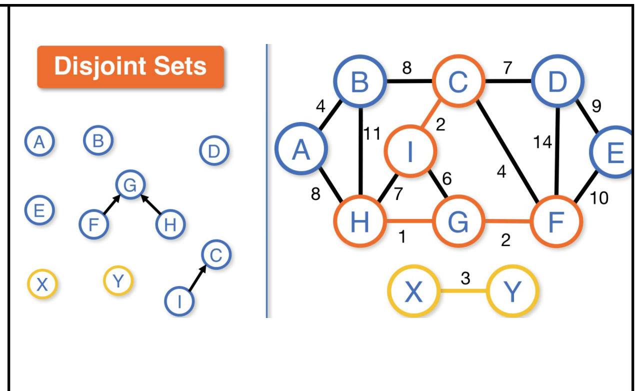


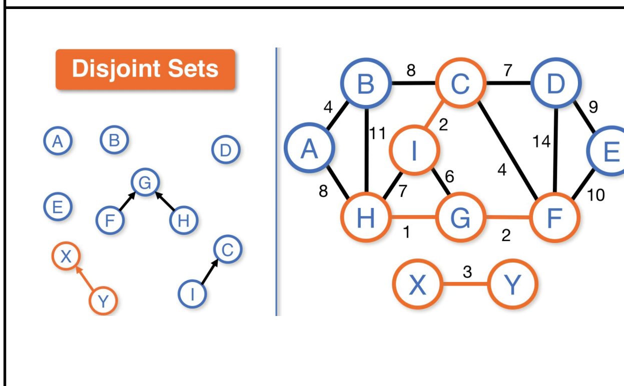


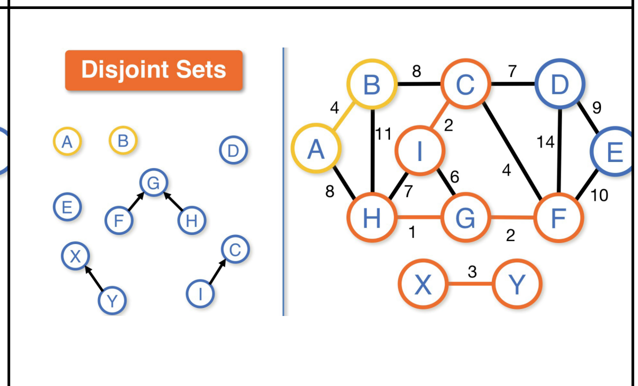
- *Kruskal's Algorithm* (sugma algorithm)
  - (1) Organize every edge from least to greatest weight
  - (2) At each edge, add it if it does not create a cycle, otherwise go to the next edge
  - (3) That's it

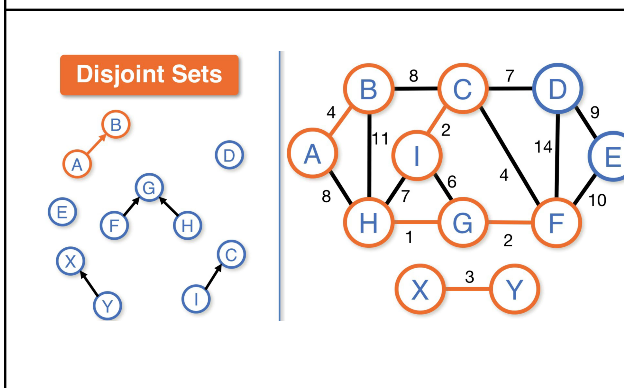


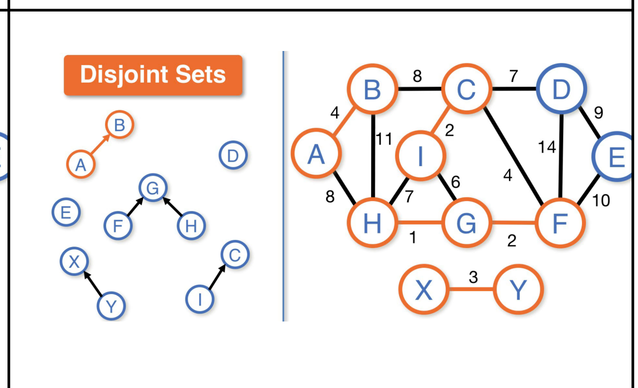

Disjoint Sets

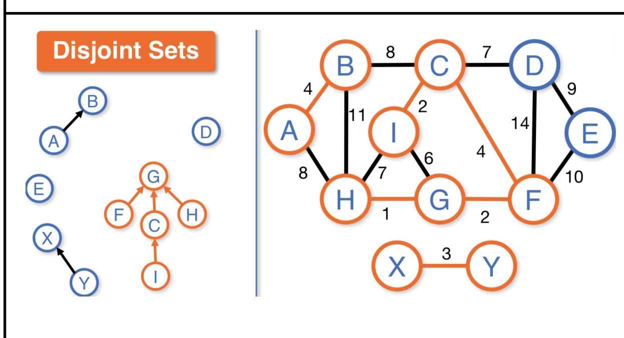

Disjoint Sets

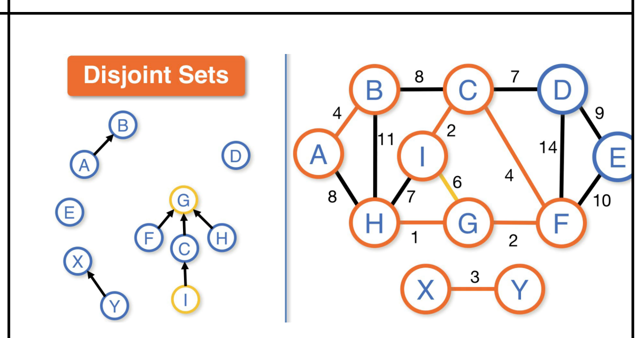

Disjoint Sets

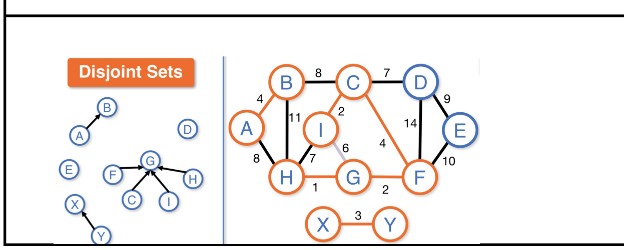

Disjoint Sets

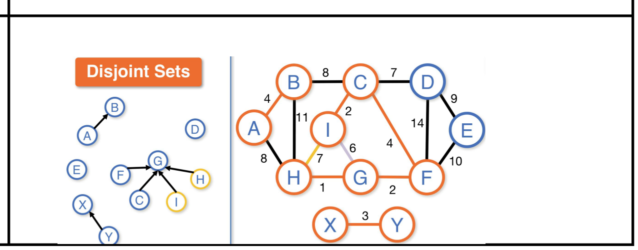

Disjoint Sets

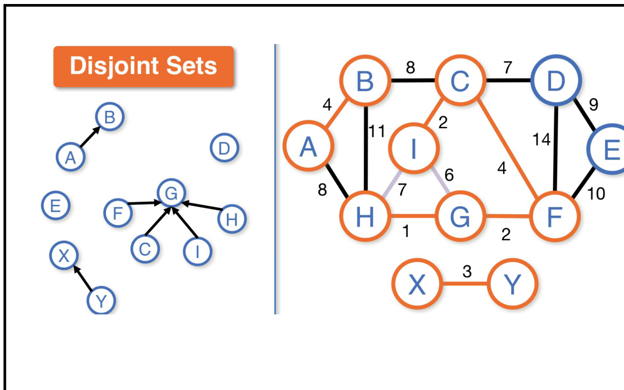

Disjoint Sets

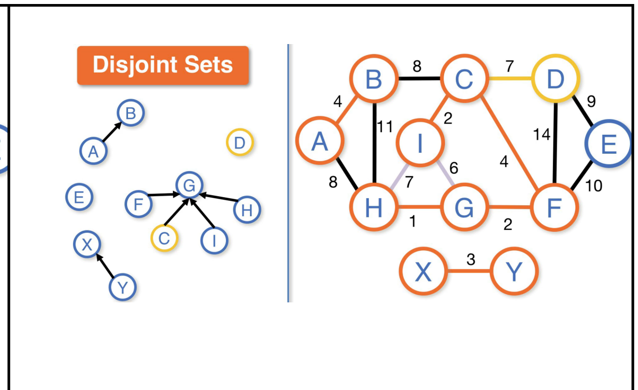

Disjoint Sets

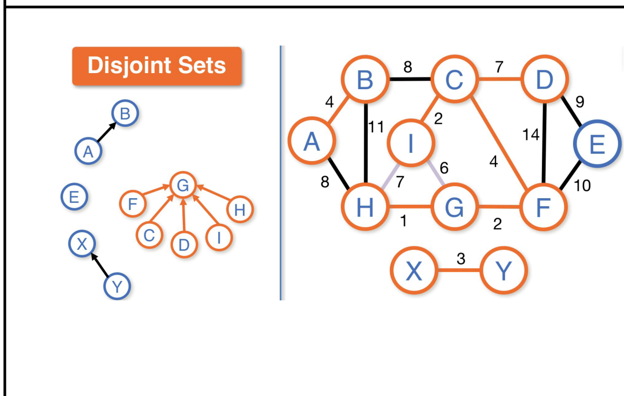

Disjoint Sets

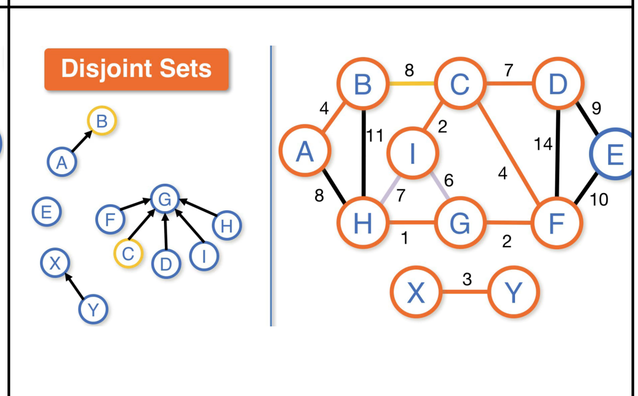

Disjoint Sets

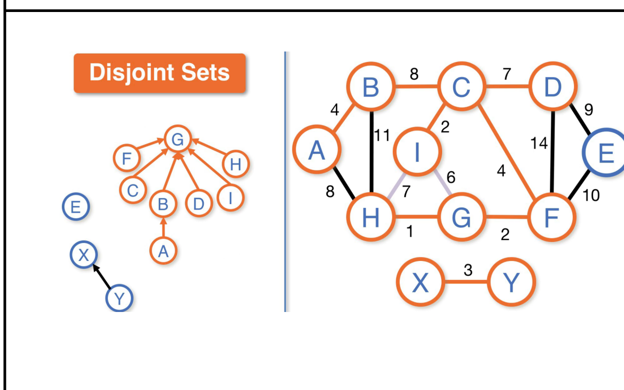

Disjoint Sets

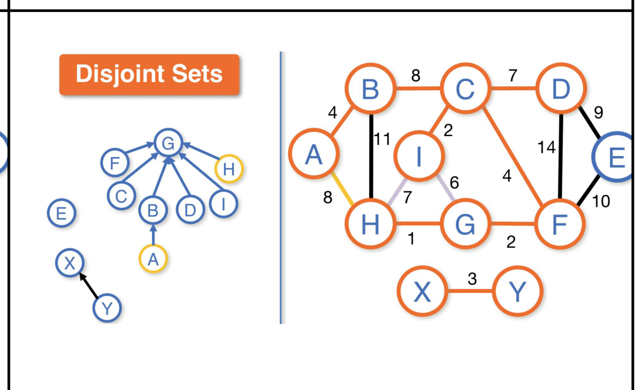

Disjoint Sets

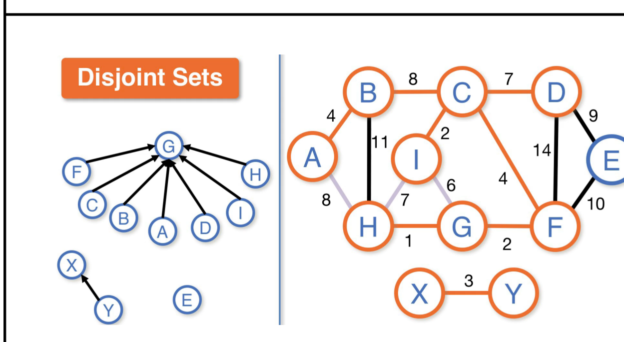

Disjoint Sets

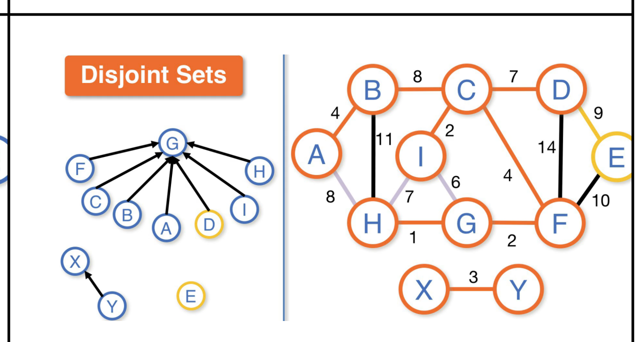

Disjoint Sets

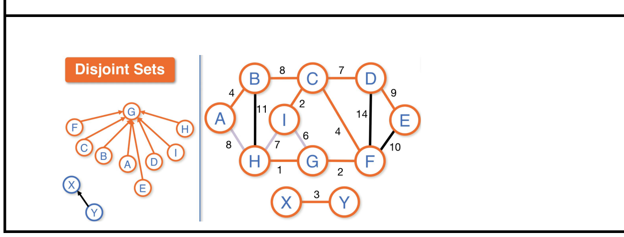

Disjoint Sets

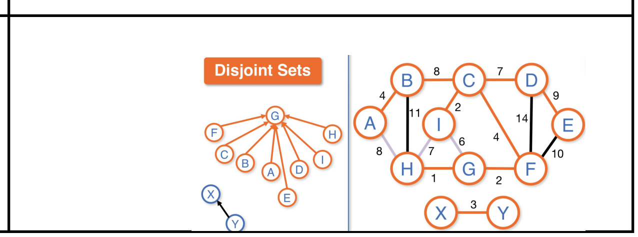

Disjoint Sets

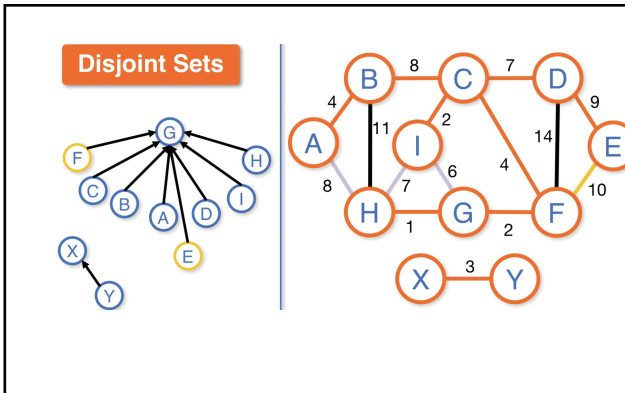

Disjoint Sets

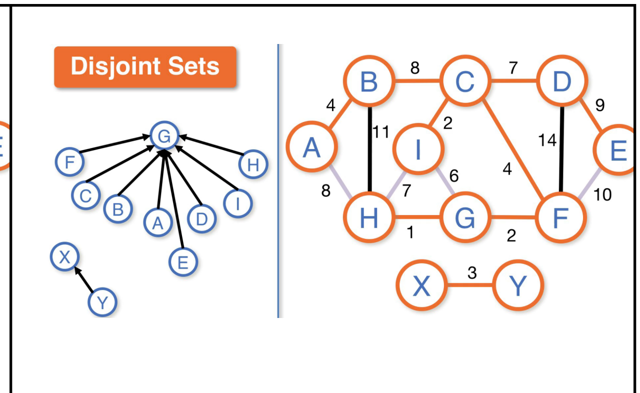

Disjoint Sets

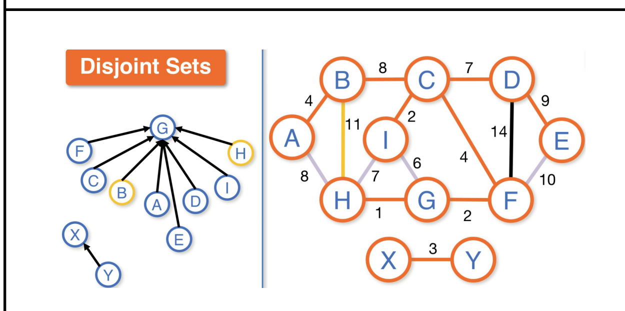

Disjoint Sets

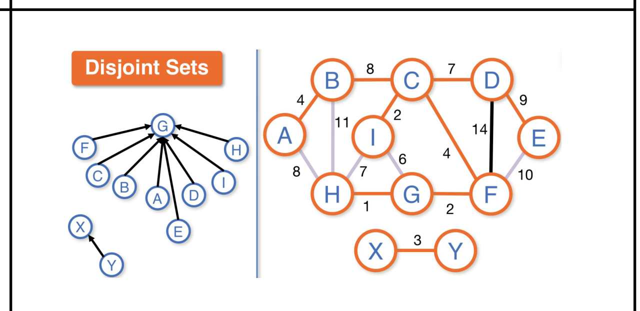

Disjoint Sets

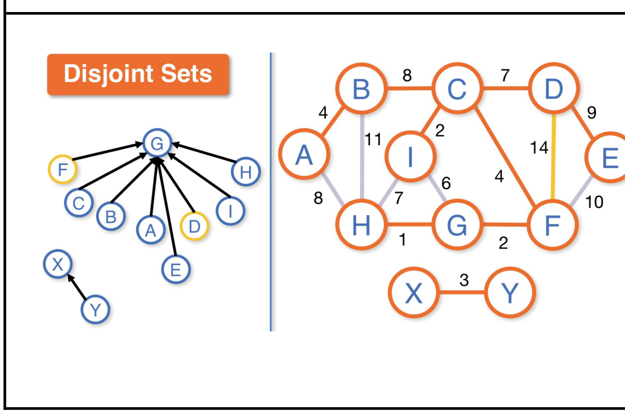

Disjoint Sets

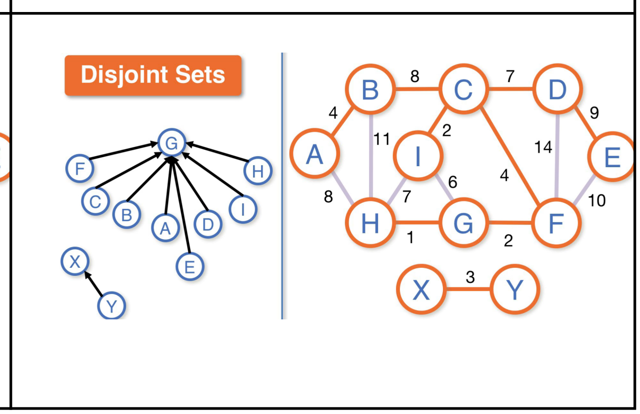

Disjoint Sets


Disjoint Sets


Disjoint Sets

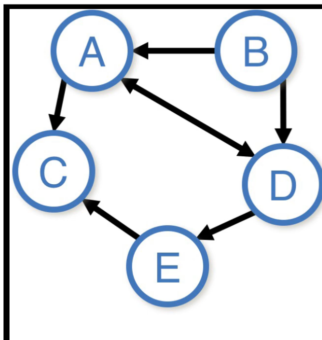

Disjoint Sets


Disjoint Sets


Disjoint Sets

12.2: Directed Graphs

- DEFINITION: A **directed graph**, also called a **digraph**, *D* is a graph where every edge is *directed* (one-way street) as opposed to *undirected* (two-way street).
  - A *directed edge* is an edge that can only be traversed one direction, indicated by the arrow head. Because of this, directed edges are classified as *ordered pairs* (in other words, *order matters*) e.g. (u, v)
  - NOTE: In an undirected graph, two vertices can either have an edge or not. In a directed graph, two vertices can either have an edge from u to v, and edge from v to u, or no edge.



Directed Graph (Digraph)

- DEFINITION: Just like undirected graphs, directed graphs also have **directed walks**, **directed paths**, **directed cycles**, **directed circuits**, and **directed trails** with the caveat that you can only transverse edges in their specified direction.

Exam 2 Study Guide

Quizlet link here

5.1: Intro & Terminology

- Use quizlet link to review terms
- Determining if two graphs are *isomorphic*: Write out the degree sequence of both graphs & see if they match

5.3: Hamiltonian Cycles and Eulerian Circuits

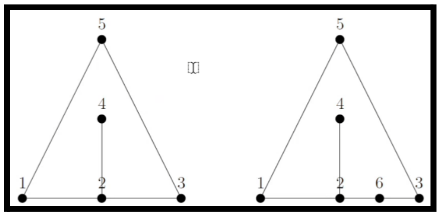
- Eulerian
  - *Eulerian Graph* - graph is *Eulerian* if, and only if, it is connected *and* every vertex has an even degree
  - *Eulerian Circuit* - a circuit that visits every ***edge*** exactly once
  - *Eulerian Trail* - a trail that visits every edge exactly once
- Hamiltonian
  - *Hamiltonian Graph* - graph is *Hamiltonian* if, and only if, with $n > 2$ vertices, each vertex in G has at least n/2 neighbors
  - *Hamiltonian Cycle* - a cycle that visits every ***vertex*** exactly once

5.4: Graph Coloring

- A **proper coloring** of G is a *coloring* of G where no two adjacent vertices have the same colors
  - k-colorable: size of coloring set is integer k
  - The **chromatic number** $\chi(G)$ of G is the smallest integer k for which G is k-colorable
- Clique
  - A **clique** of a graph G is a subset of the vertices whose induced subgraph is a complete graph.
  - The **clique number** $\omega(G)$ is the size of the largest clique in G, and sets the lower bound on the *chromatic number*
- Bipartite
  - Being bipartite is the same as being 2-colorable
  - A graph is only bipartite if it doesn't contain an odd cycle
- Algorithms for Finding the Chromatic Number
  - (1) Use first fit algorithm
  - (2) Use information on bipartites (odd cycles) & cliques to narrow down answer
- Intersection & Interval Graphs

5.5: Planar Graphs

- A graph is **planar** if it can be drawn in the plane R^2 without any edges crossing. More precisely, the only time two edges meet each other is a vertex
- If G is a graph, then an **elementary subdivision** G' of G is a graph G' that is formed by "splitting" an edge into two



- Two graphs G and H are **homeomorphic** if there is an elementary subdivision G' of G that is isomorphic to an elementary subdivision H' of H
- <u>Proving if Something is Planar</u>
    - Just try to draw the graph as planar, lul
- <u>Proving that Something **isn't** Planar</u>
    - Use Kuratowski's Theorem: A graph is planar if, and only if, it does not contain a subgraph *homeomorphic* to either K_5 or $K_{3,3}$

5.6: Counting Labeled Trees

- THEOREM (Cayley's Formula): The number of labeled trees on n vertices, where $n \geq 2$, is precisely n^{n-2}
- **Prüfer codes** uniquely identify a labeled tree
    - Know how to find a prufer code given a tree (easy)
    - Know how to find a tree given a prufer code (for this, I just opened MSPaint and did like 5 examples. Eventually you get good at it and it becomes pretty easy)

12.3: Dijkstra's Algorithm for Shortest Paths

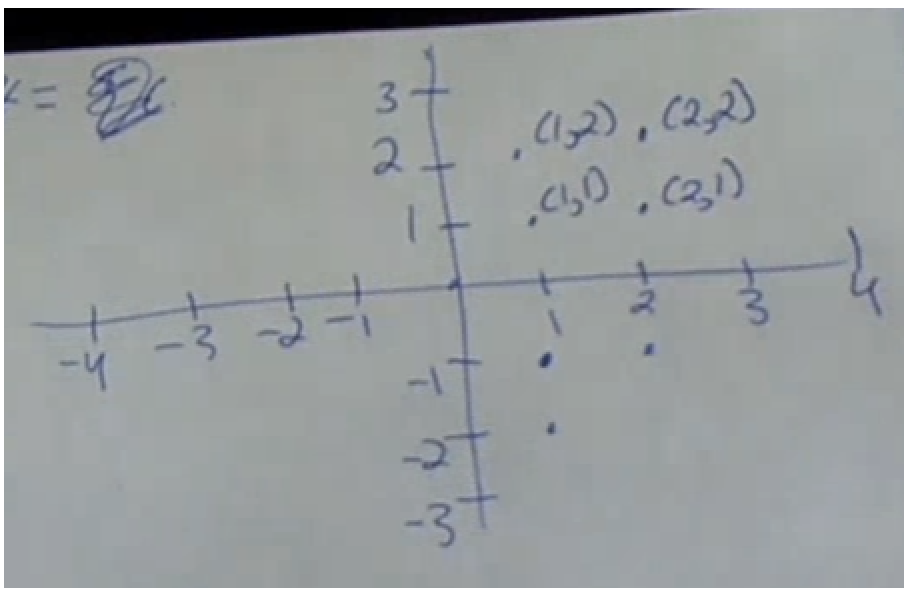
- For the purposes of Dijkstra's, we will be considering a weighted, directed graph *D* whose weight we'll say represents the **length**
- Given *D* and a specified vertex *r*, for every vertex *x* find…
    - The shortest path from *r* to *x,* and
    - The **distance** from *r* to *x*, which is the sum of the lengths of the edges in the shortest path R
- Here's probably the simplest resource I can provide to you for Dijkstra's. Try not to overcomplicate it: here

Chapter 6

6.1: Basic Notation and Terminology

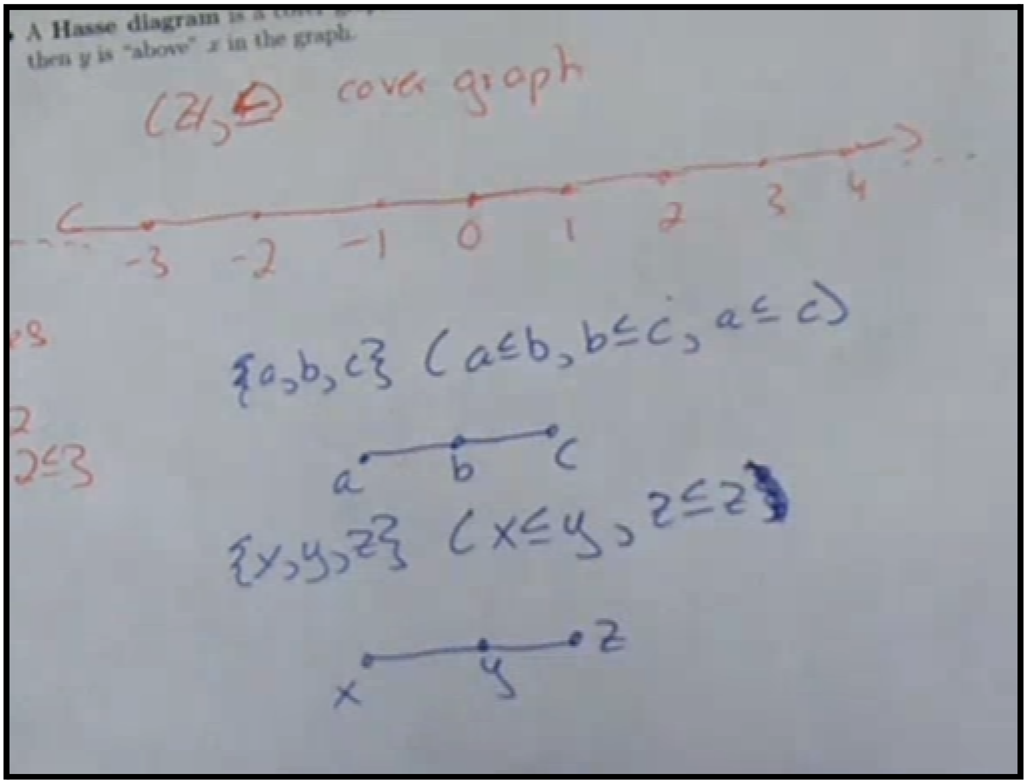
- DEFINITION: A **binary relation** on a set *X* is a set of ordered pairs (*x, y*) of elements in *X*
    - For example… $X = \{a, b, c, d\} \Rightarrow B = \{(a, b), (b, a), (c, c), (c, d), (d, a)\}$
- DEFINITION: A **partial order** on a set *X* is a *binary relation* that is…
    - *Reflexive* - For every $x \in X$, (*x*, *x*) is part of the binary relation
    - *Antisymmetric* - If $x \neq y$, then the binary relation can have either (*x, y*) or (*y, x*) (or possibly neither), but it *cannot* have both
    - *Transitive* - If (*x, y*) and (*y, z*) are in the binary relation, then so is (*x, z*)

  A set with a partial order is a **partially ordered set** (**poset**), where *P* is the *partial order* and *X* is the **ground set**
- For example…
    - Suppose we have the set of all integers {...-2, -1, 0, 1, 2, …}
        - (*a, b*) is in partial order if $a \leq b$
- If $\mathbb{Z}$ is the set of all integers, then $\mathbb{Z}^2$ is the set of all integer ordered pairs

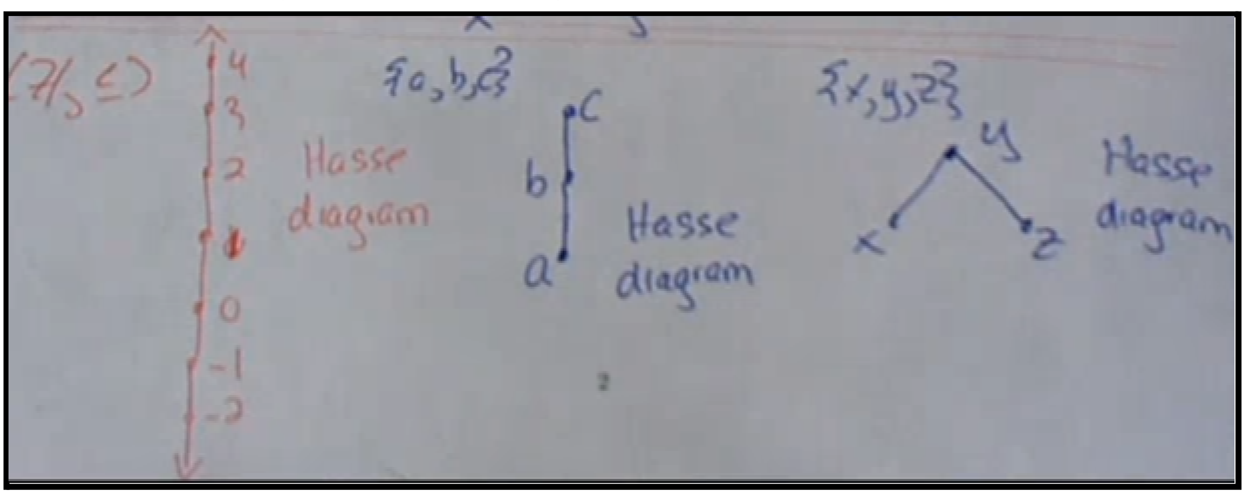


    - (a, b) and (c, d) is in partial order if $(a, b) \leq (c, d)$, that is $a \leq c$ and $b \leq d$
- DEFINITION: Take two points *x* and *y* in a poset where $x \neq y$. We say that *x* **is covered by** *y* if (*x*, *y*) is in the partial order, and there's no $z \neq x, y$ such that both (*x*, *z*) is in the partial order and (*z*, *y*) is in the partial order
    - In other words, we say *x* is covered by *y* if there does not exist an element in the poset that "comes between" *x* and *y*. For example, in our above poset with the set $\mathbb{Z}$, where $a \leq b$, we say 1 is covered by 2, but 1 is not covered by 3
    - It's the *smallest binary relation* possible in the poset

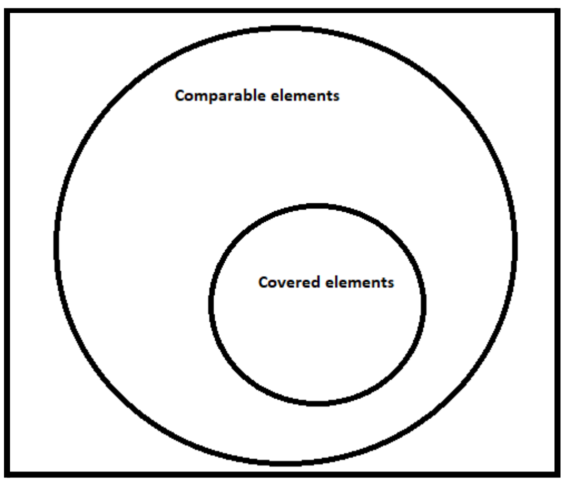
- DEFINITION: A **cover graph** of a poset is a graph where the vertices correspond to elements in the poset, and we have an edge between two vertices x and y if, and only if, either x is covered by y or y is covered by x



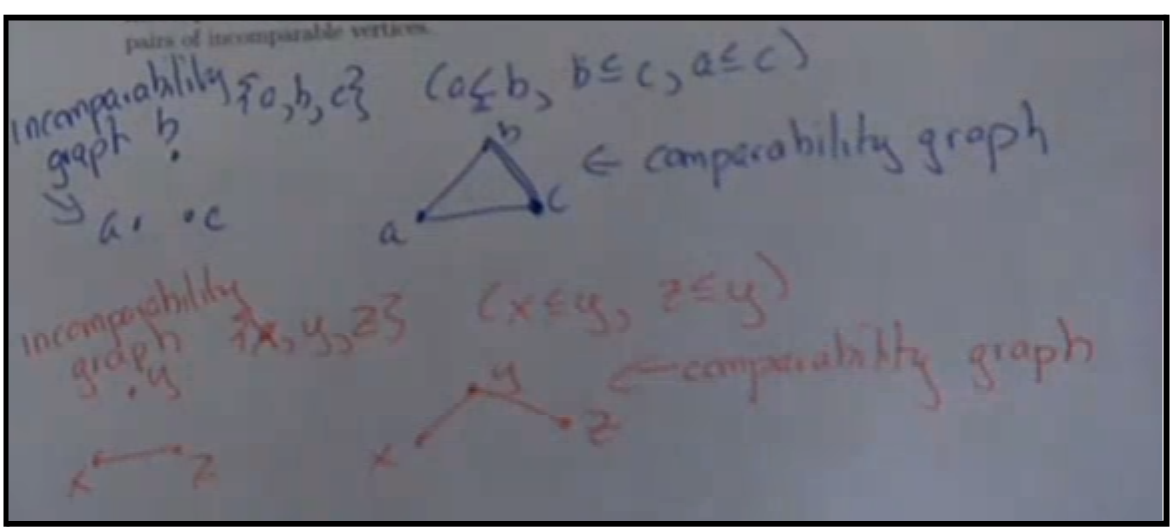
  - FIGURE: In the bottom poset, it should be $z \leq y$ instead of $z \leq z$
- Although these cover graphs are useful for showing these relations, it isn't necessarily explicit about which element is greater than or less than the other elements it's connected to. Notice that the top and the bottom posets have the same cover graph, even though their binary relations are different.
- DEFINITION: A **Hasse diagram** is a cover graph where the vertices are placed such that if $x \leq y$, then y is "above" x in the graph



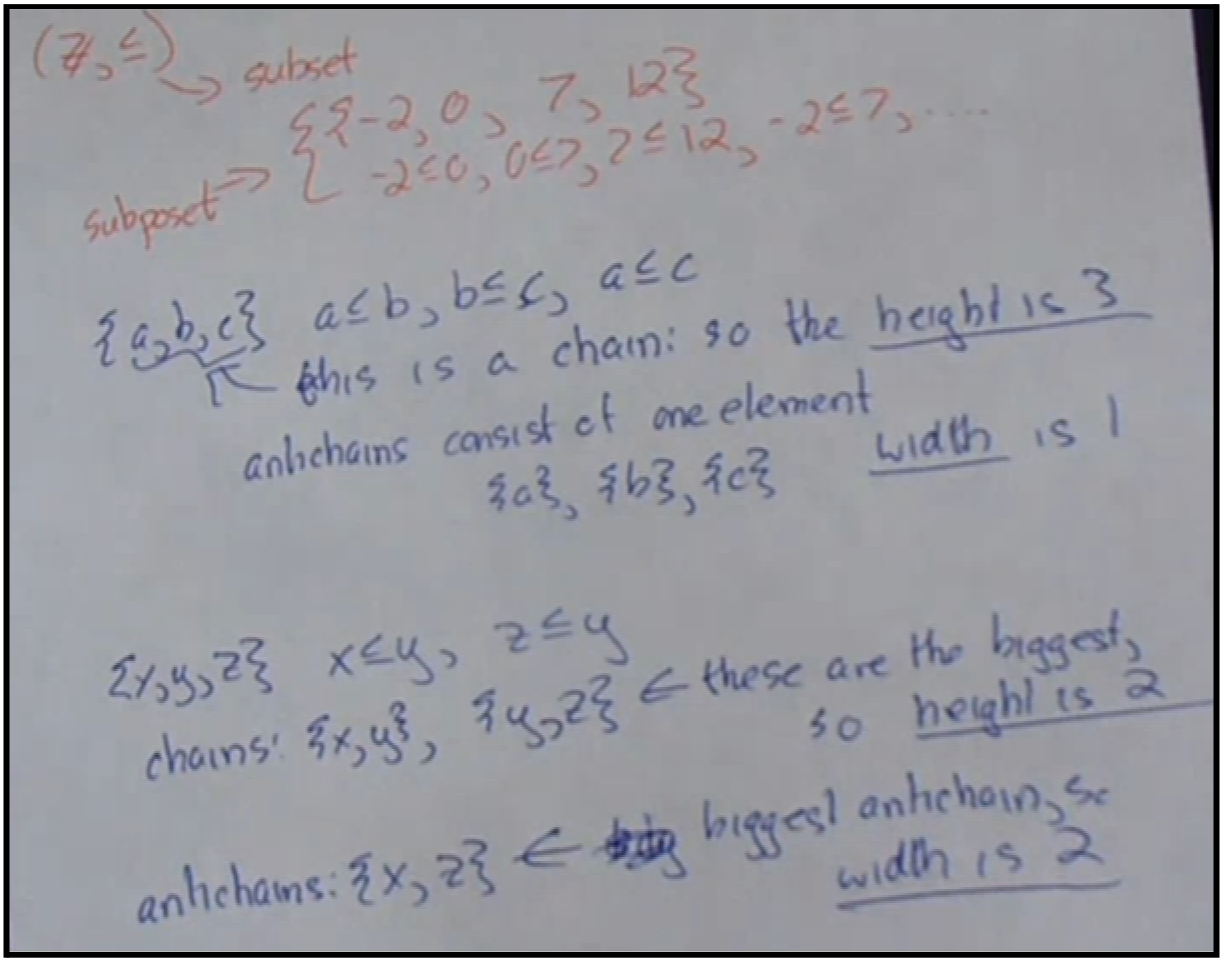
- DEFINITION: Two distinct points x, y in a poset are **comparable** if either (x, y) or (y, x) is in the partial order. Otherwise, we say that x and y are **incomparable**. A partial order is a **total order** (**linear order**) if any two distinct points are *comparable*. For example…
  - ($\mathbb{Z}$, $\leq$) is a total order because we can take any two integers (e.g. -4 & 7) and determine which is bigger
  - $\mathbb{Z}^2$ is *not* a total order because we cannot compare ordered pairs (e.g. (-1, 2) and (2, 0) are incomparable because we can't say either set is less than or greater than the other)
- DEFINITION: A **comparability graph** is a graph where the vertices correspond to the elements in the poset and there is an edge between two vertices if they are *comparable*. An **incomparability graph** is the same as a *comparability graph* except the edges are between pairs of *incomparable* vertices.
  - NOTE: You can almost view this as a cover graph, except we have "loosened" the restriction. Remember that an edge in a cover graph only exists if an element is covered by or covers another element. By definition, an element covering another element is comparable. Therefore, all elements that cover one another are comparable, but not all elements that are comparable cover one another.



  - NOTE: This is *not* a Hasse diagram; you are not required to use verticality to show which elements are bigger than the other elements
  - NOTE: The simplest way to view the *incomparability graph* is the *complement* of the *comparability graph*



- DEFINITION: If X is a poset whose partial order is P and Y is a subset of X, then Y can be made a **subposet** of X by giving Y the partial order P restricted to just the elements of Y
    - A **chain** is a subsposet where every pair of distinct elements is comparable. The **height** of a poset X is the size of its largest *chain*
    - An **antichain** is a subposet where every pair of distinct elements is incomparable. The **width** of a poset X is the size of its largest *antichain*
    - NOTE: A subgraph is to a graph as a subposet is to a poset

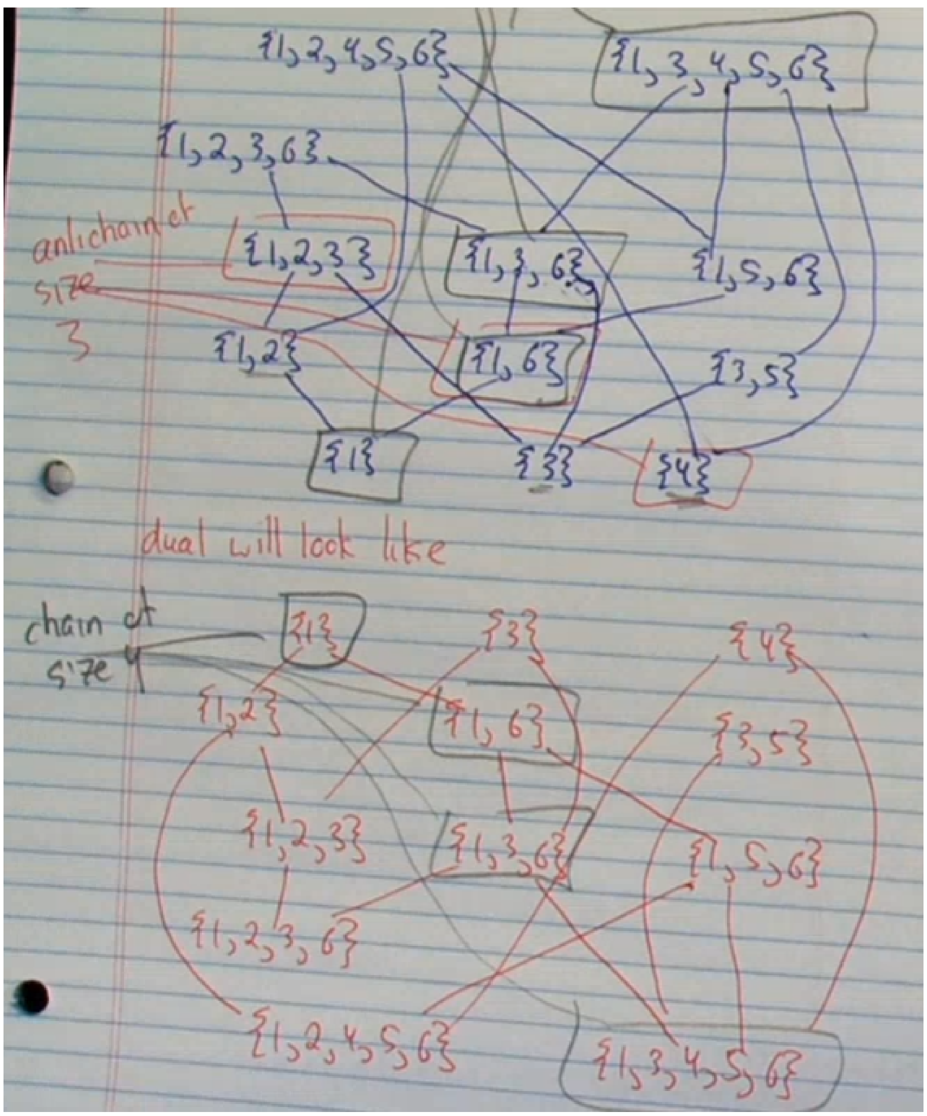


6.2: Additional Concepts for Posets

- DEFINITION: Let X and Y be posets. An **isomorphism** $f: X \to Y$ is a *bijection* that preserves the partial order. That is, for any two $x, y \in X$, we have…

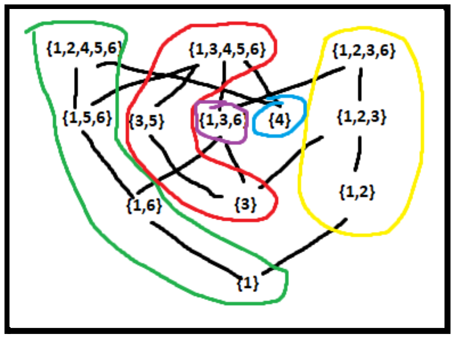
$$x \leq y \text{ if and only if } f(x) \leq f(y)$$

- DEFINITION: If X is a poset with partial order P, then the **dual** of X, denoted X^d, is the binary relation given by $\{(x, y) \mid (y, x) \in P\}$
  - NOTE: The dual of a poset is also a poset
  - NOTE: The dual of the dual of a poset X is X. That is, $(X^d)^d = X$
  - NOTE: A chain in X is also a chain in X^d. An antichain in X is also an antichain in X^d. Thus, X and X^d will always have the same height and width
- To draw the *dual* of a Hasse diagram, draw the normal Hasse diagram, and then simply flip the height positions of each element (i.e. if an element is at the bottom, then it should move to the top) but maintain the same edges

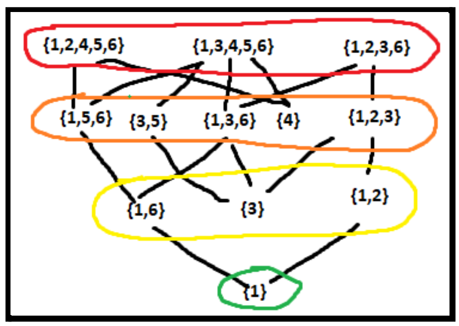


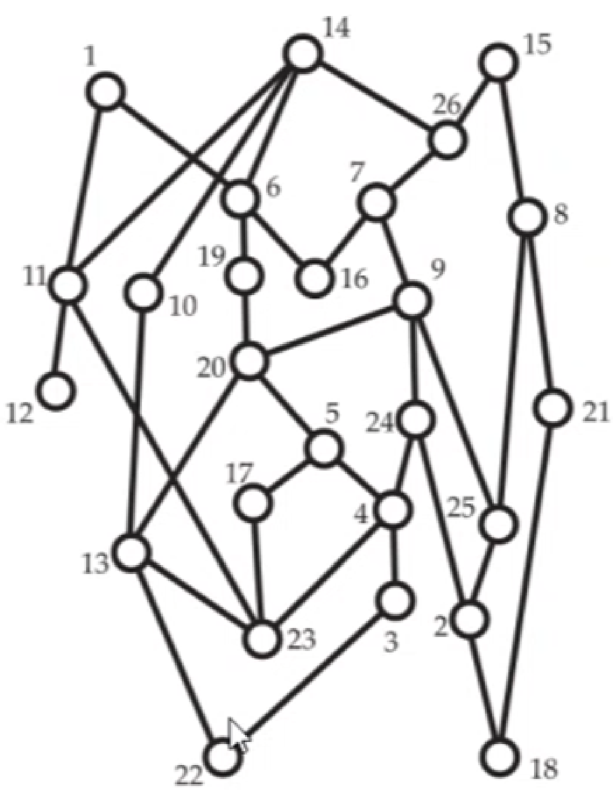
6.3: Dilworth’s Chain Covering Theorem and its Dual

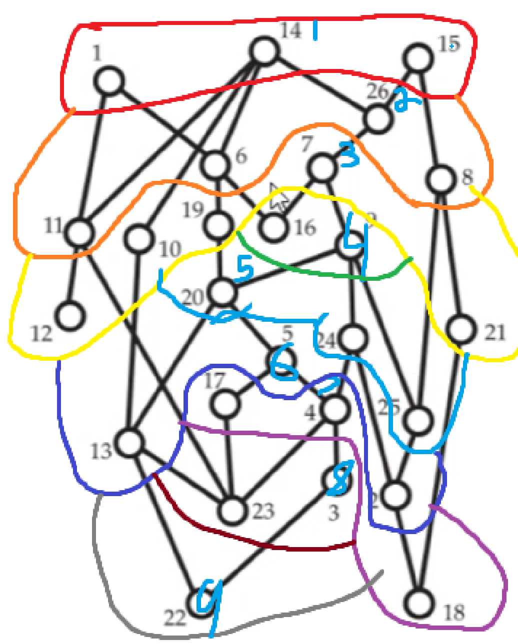
- THEOREM (Dilworth’s Theorem): If X is a poset with width w, then the elements of X can be partitioned into w sets, $C_1, C_2, \ldots, C_w$ such that each C_i is a chain



- THEOREM (Dual of Dilworth’s Theorem): If X is a poset with height h, then the elements of X can be partitioned into h sets, $A_1, A_2, \ldots, A_h$ such that each A_i is an antichain







Chapter 7

7.2: The Inclusion-Exclusion Formula

- This is basically just subtracting the portion of people who fall into both categories. For instance, if we were to sum up section A and section B of a Venn Diagram, we'd need to subtract the middle intersection because we want to avoid double counting
- EXAMPLE: A survey of 100 people found that 57 could speak French, 45 could speak German, and 30 could speak both French and German. How many people could speak French *or* German?
    - 57 + 45 - 30 = 72 → (# of people who speak French) + (# of people of speak German) - (# of people who speak both)
- EXAMPLE: A survey of 200 people asked whether they like to ski, surf, or hike. 75 said they like to ski, 32 said they like to surf, and 103 said they liked to hike. Furthermore, 25 said they liked to both ski and surf, 40 said they liked to ski and hike, and 5 said they liked to surf and hike. There were 2 people who liked to do all 3 activities. How many people didn't like doing any of the three activities?
    - Although it wasn't very evident in the above example, we can generalize the approach to these types of problems in the following way…
    - (# of people who like to do one) - (# of people who like to do two). Alternatively,
    - - (# of people who like to do one) + (# of people who like to do two)
    - Notice how they alternate; after each group of people, we either add or subtract based on if the previous group was added or subtracted. We can take the same approach here…

(total) - (# of ppl who like to do one) + (# of ppl who like to do two) - (# of ppl who like all 3)

- Doing this, we get the following…

200 - 210 + 70 - 2 = 58

- The above example demonstrates the *Inclusion-Exclusion Formula*.
- THEOREM: Suppose we have a set X, and elements of the set X can have properties P_1, P_2, ..., P_n. An element of X can have many of these P_i's or none
    - Define X_i to be the subset of X whose elements satisfy property P_i
    - Define X_{ij} to be the subset of X whose elements satisfy properties P_i and P_j
    - …
    - Define $X_{12\ldots n}$ to be the subset of X whose elements satisfy all properties $P_1, P_2, \ldots, P_n$

Then the formula for the number of elements of X that don't satisfy any single property P_i is…

$$\boxed{|X| - \sum_{i=1}^{n} |X_i| + \sum_{\{i,j\}\in\{1,2,\ldots,n\}} |X_{ij}| - \sum_{\{i,j,k,\}\in\{1,2,\ldots,n\}} |X_{ijk}| + \cdots + (-1)^n |X_{123\ldots n}|.}$$

- The above is just a long-winded way of doing the same process we did in example 2
- COROLLARY: The number of elements with at least 1, 2, …, *k* many properties is:

$$\boxed{-1\left(-\sum_{i=1}^{n}|X_i| + \sum_{\{i,j\}\in\{1,2,\ldots,n\}}|X_{ij}| - \sum_{\{i,j,k,\}\in\{1,2,\ldots,n\}}|X_{ijk}| + \cdots + (-1)^n|X_{123\ldots n}|\right)}$$

  - NOTE: The above equation is simply the formula from the theorem, but we remove the first term and flip all the signs
  - NOTE: Suppose we want the number of elements with at least two properties. Then we lob off the summation of all the elements that satisfy at least one property. In other words, we do the same thing we did with our first term and can cut it. Once doing this, ensure the first term is *always positive*
- EXAMPLE: How many positive integers less than or equal to 100 are divisible by neither 2 nor 5?
  - P_1: divisible by 2
  - P_2: divisible by 5
  - Want: Set of numbers that don't satisfy P_1 or P_2
  - $X = \{1, 2, 3, \ldots, 100\}$
  - $X_1 = \{2, 4, 6, 8, \ldots, 100\}$
  - $X_2 = \{5, 10, 15, 20\ldots, 100\}$
  - $X_{12} = \{10, 20, 30, \ldots, 100\}$
  - $|X| - \sum_{i=1}^{2}|X_i| + |X_{12}| \rightarrow 100 - (50 + 20) + 10 = 40$

7.3: Enumerating Surjections

- If A is a set of m objects, and B is a set of n objects, then the number of functions from A to B is n^m = (# of objects in B)$^{\text{\# of objects in } A}$
    - Another way of looking at this is for each n element, we have m unique choices
- But what if we only wanted to count the number of *surjections* i.e. only care that everyone has a partner (objects are allowed to have the same partner)
- We can use the inclusion-exclusion principle on the set of function from A to B
- FORMULA: The number of surjections from a set A of m objects to a set of B of n objects is…

$$\boxed{n^m - \binom{n}{1}(n-1)^m + \binom{n}{2}(n-2)^m - \cdots + (-1)^n\binom{n}{n}(n-n)^m.}$$

- How is this an inclusion-exclusion problem? Well, we can label the properties like so…

P_1 = the # of functions that do not map anything to the first element in B

P_2 = the # of functions that do not map anything to the second element in B

…

P_n = the # of functions that do not map anything to the nth element in B

- Since the inclusion-exclusion principle gives us the number of things that *don't* satisfy any single property P_i, we need to have our properties to specify the number of functions that *don't* map certain elements. This will then give us the number of functions that *do* map.
- But we're talking about more than just one property here; we need the number of surjections that break *all* single properties P_i, which is why we need to use the combination notation before each term. Effectively, we're using the inclusion-exclusion principle on every property P_i; the combination notation allows for us to conveniently do this

7.4: Derangements

- DEFINITION: A **permutation** is a bijection $\sigma: \{1, 2, ..., n\} \rightarrow \{1, 2, ..., n\}$ to itself
- DEFINITION: A **derangement** is a permutation where no element i is mapped to itself $(\sigma(i) \neq i)$
- The number of permutations on $\{1, 2, \ldots, n\}$ is n!
- What about counting the number of derangements? Well, we can also use the inclusion-exclusion principle to find that.
  - Define our properties as follows…

    P_1 means σ sends 1 to 1

    P_2 means σ sends 2 to 2

    P_n means σ sends n to n
  - Then, the number of derangements is the number of permutations that don't satisfy any of the properties $P_1, P_2, \ldots, P_n$
  - Since he number of permutations is $n!$, our equation will take a similar form of the inclusion-exclusion formula but it will use factorials
- FORMULA: The number of derangements is as follows…

$$n! - \binom{n}{1}(n - 1)! + \binom{n}{2}(n - 2)! -... (-1)^n \binom{n}{n}(n - n)!$$

Chapter 8

8.2: Generating Functions

- Recall the Binomial Theorem…

$$(1 + x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$$

Note that $(1 + x)^n$ encodes all the different combinations where you choose from *n* objects. If you want to determine how many ways there are to choose *k* objects from a set of *n* objects, you look at the coefficient of x^k. Thus we say $(1 + x)^n$ is a *generating function* for the sequence of choosing *k* objects from a set of *n* objects

- DEFINITION: Let $S = \{a_0, a_1, a_2, \ldots\}$ be a sequence of numbers. We say that a power series $F(x)$ is a **generating function** of *S* if for every $k = 0, 1, 2, \ldots$ we have that a_k is the coefficient of x^k in $F(x)$. That is…

$$F(x) = \sum_{k=0}^{\infty} a_k x^k$$

- EXAMPLE:You are packing for a trip. You can pick from 4 red shirts, 4 blue shirts, and 4 green shirts. Write the generating function for the number of ways to pack *k* shirts.
  - $(1 + x + x^2 + x^3 + x^4)^3$
  - Note each factor in the answer above. For 1 (which as we know is x^0), this factor represents the number of ways to pick 0 red shirts, and its coefficient is 1 because there is only 1 way to pick 0 red shirts. For *x*, this factor represents the number of ways to pick 1 red shirt, and its coefficient is 1 because there is only 1 way to pick 1 red shirt (in this problem we assume all shirts of the same color are equivalent to one another). For x^2, this factor represents the number of ways to pick 2 red shirts, and its coefficient is 1 because there is only 1 way to pick 2 red shirts. Etc.
  - So, $1 + x + x^2 + x^3 + x^4$ is the number of ways to pick a colored shirt. Therefore, the number of ways to pick *k* shirts is $(1 + x + x^2 + x^3 + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2 + x^3 + x^4)$ or $(1 + x + x^2 + x^3 + x^4)^3$
  - Put another way, if I expanded this out…

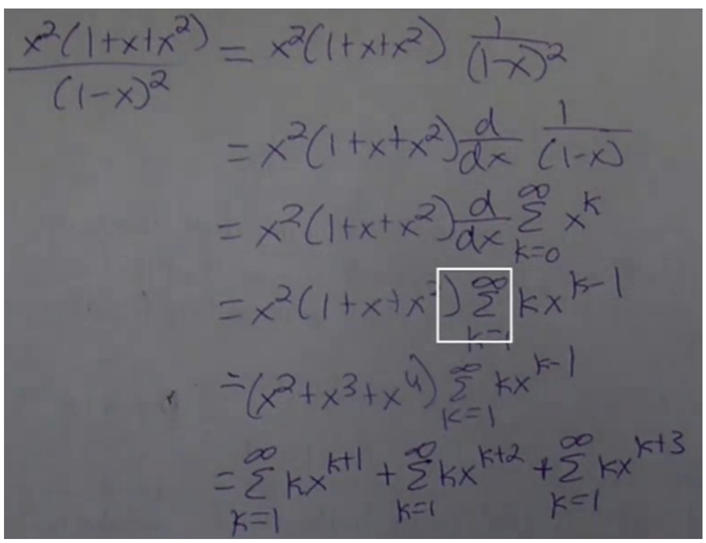
$$\boxed{x^{12} + 3x^{11} + 6x^{10} + 10x^9 + 15x^8 + 18x^7 + 19x^6 + 18x^5 + 15x^4 + 10x^3 + 6x^2 + 3x + 1}$$

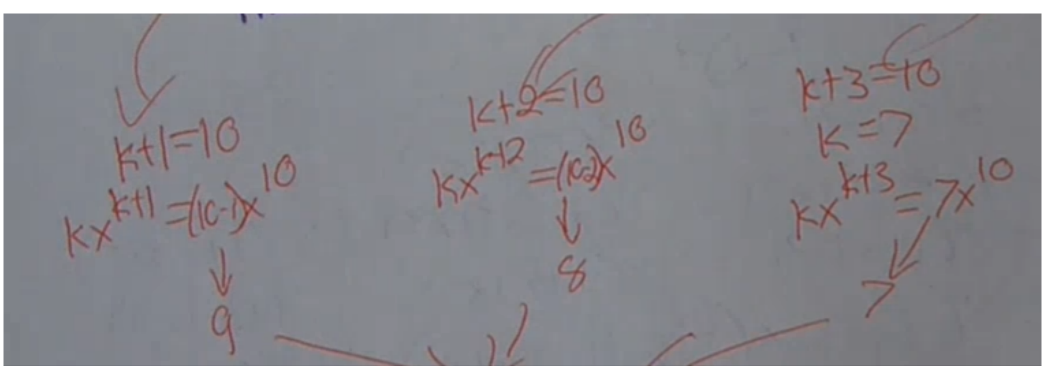
  - The coefficient of each term represents the number of ways to pick *k* shirts, where *k* is the exponent. Look at x^{12}. We only have 12 shirts, and so obviously it makes sense that we would only have 1 way to pick 12 shirts (x^{12} is really $1 * x^{12}$).

Going down, we have 3 ways to pick 11 shirts, which makes sense; the case where we don't take a red shirt, a case where we don't take a blue shirt, and a case where we don't pick a green shirt, 3 ways.

- EXAMPLE: Same as the previous one, except now you can pick from *up to* 7 red shirts, and you have to pack at least 1 blue shirt?
  - $(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)(1 + x + x^2 + x^3 + x^4)(x + x^2 + x^3 + x^4)$
  - $(\# \, of \, ways \, to \, pick \, red \, shirts)(\# \, of \, ways \, to \, pick \, green \, shirts)(\# \, of \, ways \, to \, pick \, blue)$
  - This is very similar to the first problem except…
    - We have more red shirts to work with
    - We can no longer take 0 blue shirts
  - Both of which are very simple to handle. For the red shirts, we extend out our clause to account up to 7 red shirts. For green, we leave it as is. For blue, it's the same clause except now we no longer have x^0 because we aren't allowed to pick 0 blue shirts
- EXAMPLE: Generating function for choosing k shirts when you can pick as many shirts of a color as you want (i.e. there is an infinite supply of red, blue, and green shirts)
  - $(1 + x + x^2 + ...)(1 + x + x^2 + ...)(1 + x + x^2 + ...)$
  - While the above answer *is* valid, this isn't all that pretty or concise. Ideally, we should smush these down to be simpler. Notice that this is actually a geometric series…
  - $1 + x + x^2 + ... = \frac{1}{1-x}$
  - The mathematically inclined among you might object and say *"Hey, we can only do this if $x \leq 1$,"* and you'd be right in any other setting. But thankfully in combinatorics, **we don't care**
    - The real answer is that in math, x is actually supposed to mean something, whereas here in combinatorics we are merely using x as a means to give us our answer; in combinatorics, for generating functions, x doesn't equal anything
  - So the real answer is $(\frac{1}{1-x})^3$
- Alright, so we have a generating function. Now how do we find the coefficient for some kth item, or just the coefficients in general?
- Well, it's kinda screwy…

- $\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k$
- $\frac{1}{1+x} = \sum_{k=0}^{\infty} (-1)^k x^k$
- $\frac{1}{1-x^2} = \sum_{k=0}^{\infty} x^{2k}$
- $\frac{1}{1-x^n} = \sum_{k=0}^{\infty} x^{nk}$
- $\frac{1}{(1-x)^2} = \frac{d}{dx}(\frac{1}{1-x}) = \frac{d}{dx}(\sum_{k=0}^{\infty} x^k) = \sum_{k=1}^{\infty} kx^{k-1}$
- $\frac{1}{(1-x)^3} = \frac{1}{2}\frac{d^2}{dx^2}(\frac{1}{1-x}) = \frac{1}{2}\frac{d^2}{dx^2}(\sum_{k=0}^{\infty} x^k) = \sum_{k=2}^{\infty}(\frac{1}{2}k(k-1))x^{k-2}$
- $\frac{1}{(1-x)^4} = \frac{1}{3!}\frac{d^3}{dx^3}(\frac{1}{1-x}) = \frac{1}{3!}\frac{d^3}{dx^3}(\sum_{k=0}^{\infty} x^k) = \sum_{k=3}^{\infty}(\frac{1}{3!}k(k-1)(k-2))x^{k-3}$
- $\frac{1}{(1-x)^n} = \frac{1}{(n-1)!}\frac{d^{n-1}}{dx^{n-1}}(\frac{1}{1-x}) = \frac{1}{(n-1)!}\frac{d^{n-1}}{dx^{n-1}}(\sum_{k=0}^{\infty} x^k) = \sum_{k=3}^{\infty}(\frac{1}{(n-1)!}k(k-1)(k-2)\cdots(k-n+2))x^{k-n+1}$





- FIGURE: You add up 9 + 8 +7 to make 24. Got cut off, sowwy ; (

Chapter 9

- DEFINITION: A **recurrence equation** is an equation using the symbols…
$$n, s_n, s_{n+1}, s_{n+2}, \ldots, s_{n+k}$$
for some k, with an integer function $S : Z \rightarrow R$ a solution for all $n \in Z$
$$s_n = S(n), s_{n+1} = S(n + 1), \ldots, s_{n+k} = S(n + k)$$
satisfies the equation
- Think all the way back to Chapter 3 where we used recursion in order to solve things like the number of strings with certain properties
  - For example, how many strings of length *n* with digits 0, 1, or 2 are there if 102 is not allowed in the string?
$$S(n) = 3S(n - 1) - S(n - 3)$$
  - Here, we use recursion to answer our problem because our answer depends on the answers of smaller S(*n*)'s (i.e. we need S(1), S(2), S(3), … if we want to solve for bigger *n*'s)
- Instead of having to rely on recursion and previous answers, it'd be convenient if we could convert this equation to a form where we can simply plug in our input and get the same answer without having to do recursion
- This is what Chapter 9 is about: taking recurrence equations and putting them in a form where we don't have to rely on recursion
- DEFINITION: A **linear recurrence equation** is a recurrence equation of the form
$$c_k s_{n+k} + c_{k-1} s_{n+k-1} + \ldots + c_1 s_{n+1} + c_0 s_n = g(n)$$
where $c_0, c_1, c_2, \ldots, c_k$ are constants
  - A linear recurrence equation is **homogeneous** if $g(n) = 0$, and **nonhomogenous** otherwise
- Solving Recurrence Equations
  - *Homogenous* - RHS is 0 in a recurrence equation
    - (1) If not already, move everything to one side of the = sign

$$\boxed{g_{n+2} = 3g_{n+1} - 2g_n \Rightarrow S(n + 2) = 3S(n + 1) - 2S(n) \Rightarrow S(n + 2) - 3S(n + 1) + 2S(n) = 0}$$

    - (2) Replace instances of $S(n)/g_n$ with r^n

$$\boxed{S(n + 2) - 3S(n + 1) + 2S(n) = 0 \Rightarrow r^{n+2} - 3r^{n+1} + 2r^n = 0}$$

    - (3) Take out r^n and convert it into a polynomial to get its roots

$$\boxed{r^{n+2} - 3r^{n+1} + 2r^n = 0 \Rightarrow r^2 - 3r + 2 = 0 \Rightarrow (r - 2)(r - 1) = 0}$$

    - (4) Use the roots to create a general solution by adding constants before the terms

$$(r - 2)(r - 1) = 0 \Rightarrow c_1 2^n + c_2 * 1^n = S(n)$$
$$S(n + 1) = c_1 2^{n+1} + c_2 * 1^{n+1}$$
$$S(n + 2) = c_1 2^{n+2} + c_2 * 1^{n+2}$$

- I) If your roots have multiplicities, you add in additional terms to your original terms with escalating powers of *n*, up to the power of *k* - 1 where *k* is your multiplicity

$$(r - 2)(r + 3)^2(r + 2)^3 \Rightarrow$$
$$c_1 2^n$$
$$+ d_1(-3)^n + d_2 n(-3)^n$$
$$+ b_1(-2)^n + b_2 n(-2)^n + b_3 n^2(-2)^n$$

- (5) If given base cases, we can determine specific solutions by solving for our constants (example below might not work out properly but it's the idea that counts)

$$S(1) = 3\,; S(2) = 7;\ c_1 2^n + c_2 * 1^n = S(n)$$
$$c_1 2^1 + c_2 = 3$$
$$c_1 2^2 + c_2 = 7$$
$$Solve\ for\ c_1\ \&\ c_2$$

- *Nonhomogenous* - RHS is Non-0 in a recurrence equation
  - (1) Pretend it's homogenous and find the general solution for that homogeneous equation (get up to step 4). Call it S_g
    - DEFINITION: The **advancement operator** (*A*) symbol "advances" our *n*, whether it's in an *S*(*n*) or if we just have *n* in general
      - e.g. $AS(n + 1) = S(n + 2),\ An = n + 1,\ An^2 = (n + 1)^2$
      - e.g. $A^2 S(n) = S(n + 2)$
    - *f* in our equations can be substituted with $S(n)$
      - e.g. $(A - 5)(A + 2)f \Rightarrow (A - 5)(A + 2)S(n)$
    - If our given recurrence equation is already displaying its roots, however it has *advancement operators* instead of *r*, we can shortcut to our general solution by simply replacing those instances of the advancement operator with *r*. We can also throw out the *f*'s that are there, as well (apparently)

- e.g. $(A - 5)(A + 2)f \Rightarrow (r - 5)(r + 2)$

$$(A - 2)(A - 4)f = 3n^2 + 9^n$$

Using advancement operator skip: $(A - 2)(A - 4)f = (r - 2)(r - 4) \Rightarrow c_1 2^n + c_2 4^n$

- (2) Find a particular solution to the nonhomogenous equation. Call it S_p
  - Want to find a suitable replacement for $S(n)$ using the right-hand side of our equation (i.e. the thing we set to 0 to get our general solution in step 1)
  - Two Approaches
    - *Exponential Component* (c^n): Start by trying dc^n, where d is some unknown constant. If that doesn't work, try increasing powers of n: $d * nc^n$, then $d * n^2 c^n$, etc.
    - *Power Component* (n^k): Use a polynomial of degree k. For instance, if n^3 then try $d_3 n^3 + d_2 n^2 + d_1 n + d_0$

$$(A - 2)(A - 4)f = 3n^2 + 9^n$$
$$3n^2 \Rightarrow c_2 n^2 + c_1 n + c_0$$
$$9^n \Rightarrow d9^n$$
$$3n^2 + 9^n \Rightarrow c_2 n^2 + c_1 n + c_0 + d9^n = S(n)$$
$$(A - 2)(A - 4)f \Rightarrow (A - 2)(A - 4)S(n) \Rightarrow (A - 2)(A - 4)(c_2 n^2 + c_1 n + c_0 + d9^n) = 3n^2 + 9^n$$

  - After creating a guess $S(n)$, attempt to solve for the constants c_2, c_1, c_0, d . If you run into a contradiction, go back to step 2 and create a new replacement for $S(n)$

$$(A - 2)(A - 4)(c_2 n^2 + c_1 n + c_0 + d9^n) \Rightarrow 3c_2 n^2 - 8c_2 n + 3c_1 n + 3c_0 - 4c_1 - 2c_2 + 35d9^n$$
$$n^2: 3c_2 = 3\ [3cn^2 = 3n^2]$$
$$n: - 8c_2 + 3c_1 = 0$$
$$const.: 3c_0 - 4c_1 - 2c_2 = 0$$
$$9^n\ term: 35d = 1$$
$$c_0 = \frac{38}{9}, c_1 = \frac{8}{3}, c_2 = 1, d = \frac{1}{35}$$
$$S_p = n^2 + \frac{8}{3}n + \frac{38}{9} + \frac{1}{35}9^n$$

- (3) Add S_g & S_p together, that is your general solution to the nonhomogenous equation

$$c_1 2^n + c_2 4^n + n^2 + \frac{8}{3}n + \frac{38}{9} + \frac{1}{35}9^n$$

Exam 3 Study Guide

[quizlet]

[Chapter 6: Partially Ordered Sets]

Chapter 7: Inclusion-Exclusion

- The Inclusion-Exclusion Formula
  - for counting how many satisfy 0 properties

$$\boxed{|X| - \sum_{i=1}^{n} |X_i| + \sum_{\{i,j\} \in \{1,2,\dots,n\}} |X_{ij}| - \sum_{\{i,j,k,\} \in \{1,2,\dots,n\}} |X_{ijk}| + \dots + (-1)^n |X_{123\dots n}|.}$$

  - for counting how many satisfy exactly k properties (not technically in study guide)
    - Use the same formula as above, but instead of solving for the right-hand side, replace the item we want to solve for with x and solve for x
    - For example, setting up the solution for #10 in the textbook would be…

$$268 - (174 + 139 + 112) + (102 + 81 + 71) - x = 37$$

  - for counting how many satisfy at least $k > 0$ properties
    - Use the same formula as above, but lob off the summations whose number of properties is less than the number of properties we are looking for.

satisfy at least one property
$$\sum_{i=1}^{n} |X_i| - \sum |X_{ij}| + \sum |X_{ijk}| - \dots + (-1)^{n+1} |X_{123\dots n}|$$

satisfy at least two properties
$$\sum |X_{ij}| - \sum |X_{ijk}| + \sum |X_{ijkl}| - \dots + (-1)^{n} |X_{123\dots n}|$$

satisfy at least k many properties
$$\sum |X_{i_1 i_2 \dots i_k}| - \sum |X_{i_1 i_2 \dots i_k i_{k+1}}| + \dots + (-1)^{n-k+1} |X_{123\dots n}|$$

    - NOTE: It will still use the same +, -, +, … sequence. Always ensure that the first element is positive
- Counting Surjections
  - Where m is the |A| and n is |B|

$$\boxed{n^m - \binom{n}{1}(n-1)^m + \binom{n}{2}(n-2)^m - \dots + (-1)^n \binom{n}{n}(n-n)^m.}$$

- Counting Derangements
  - Where n is |A|

$$\boxed{n! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! - \dots (-1)^n \binom{n}{n}(n-n)!}$$

Chapter 8: Generating Functions

- DEFINITION: A **generating function** is a function that encodes all the different combinations where you can choose *k* objects from a set of *n* objects by looking at the coefficient of x^k
- Setting Up a Generating Function
    - Also see Q2 in Exercises in Ch. 8 in textbook
- Extracting Coefficients from Generating Functions
    - Important considerations…
        - $\frac{1}{1-(something)} = \sum_{k=0}^{\infty}(something)^k$
            - For example: $\frac{1}{1-2x} = \sum_{k=0}^{\infty}(2x)^k = (1 + 2x + 4x^2 + 8x^3 + ...)$
            - For example: $\frac{1}{1-x^2} = \sum_{k=0}^{\infty}(x^2)^k = \sum_{k=0}^{\infty}x^{2k} = (1 + x^2 + x^4 + ...)$
        - $\frac{1}{1+x} = (1 - x + x^2 - x^3 + ...)$ [alternating signs starting w/ +]
        - $\frac{1}{(1-x)^n} = \frac{1}{(n-1)!}\frac{d^{n-1}}{dx^{n-1}}\sum_{k=0}^{\infty}x^k$ [we want to get to the form of $\frac{1}{1-(something)}$, so we have to take derivatives to do that]
            - $\rightarrow \frac{1}{(n-1)!}\frac{d^{n-1}}{dx^{n-1}}\sum_{k=0}^{\infty}x^k = \sum_{k=n-1}^{\infty}\frac{1}{(n-1)!}k(k-1)(k-2)...(k-n+2))x^{k-n} + 1$
            - i seriously doubt we'd have to memorize this cause he said he'd give us some of these on the exam but if not then w/e gg i guess
            - For example: $\frac{1}{(1-x)^2} = \frac{1}{1!}\frac{d}{dx}(\frac{1}{1-x}) = \frac{d}{dx}\sum_{k=0}^{\infty}x^k = \sum_{k=0}^{\infty}kx^{k-1}$
    - The case of $g(x) = \frac{1}{1-x}$ $\Rightarrow$ $\frac{1}{1-(something)} = \sum_{k=0}^{\infty}(something)^k$
    - The case where it's of the form $g(x) = (f(x))^m$
        - See $\frac{1}{(1-x)^n}$ above
    - The case where the answer is a product of terms from multiple generating functions (i think this is right idk)
        - For example: x^3 $from\ g(x) = (\frac{1}{1-2x})(\frac{1}{1-x^2})$

$$\Rightarrow \sum_{k=0}^{\infty}(2x)^k\sum_{k=0}^{\infty}(x^2)^k = (1 + 2x + 4x^2 + 8x^3 + ...)(1 + x^2 + x^4 + ...)$$

- Go through and look for all the possible combination of terms that'd be x^3

$$(2x, x^2),\ (8x^3,\ 1)$$

- Multiply them together and then add the coefficients

$$2x^3 + 8x^3 \Rightarrow 2 + 8 = 10$$

Chapter 9: Recurrence Equations

- Definitions
  - See hyperlink.
- Setting Up Recurrences
  - Basically go through all the different possible cases you can have and generate recurrence equations for each case. At the end, add up all of those generated equations.
- Solving Recurrence Equations
  - See main hyperlink.
  - Using Generating Functions
    - I have absolutely no idea what he’s on about and he didn’t record last Tuesday’s lecture ¯_(ツ)_/¯