LECTURE #7

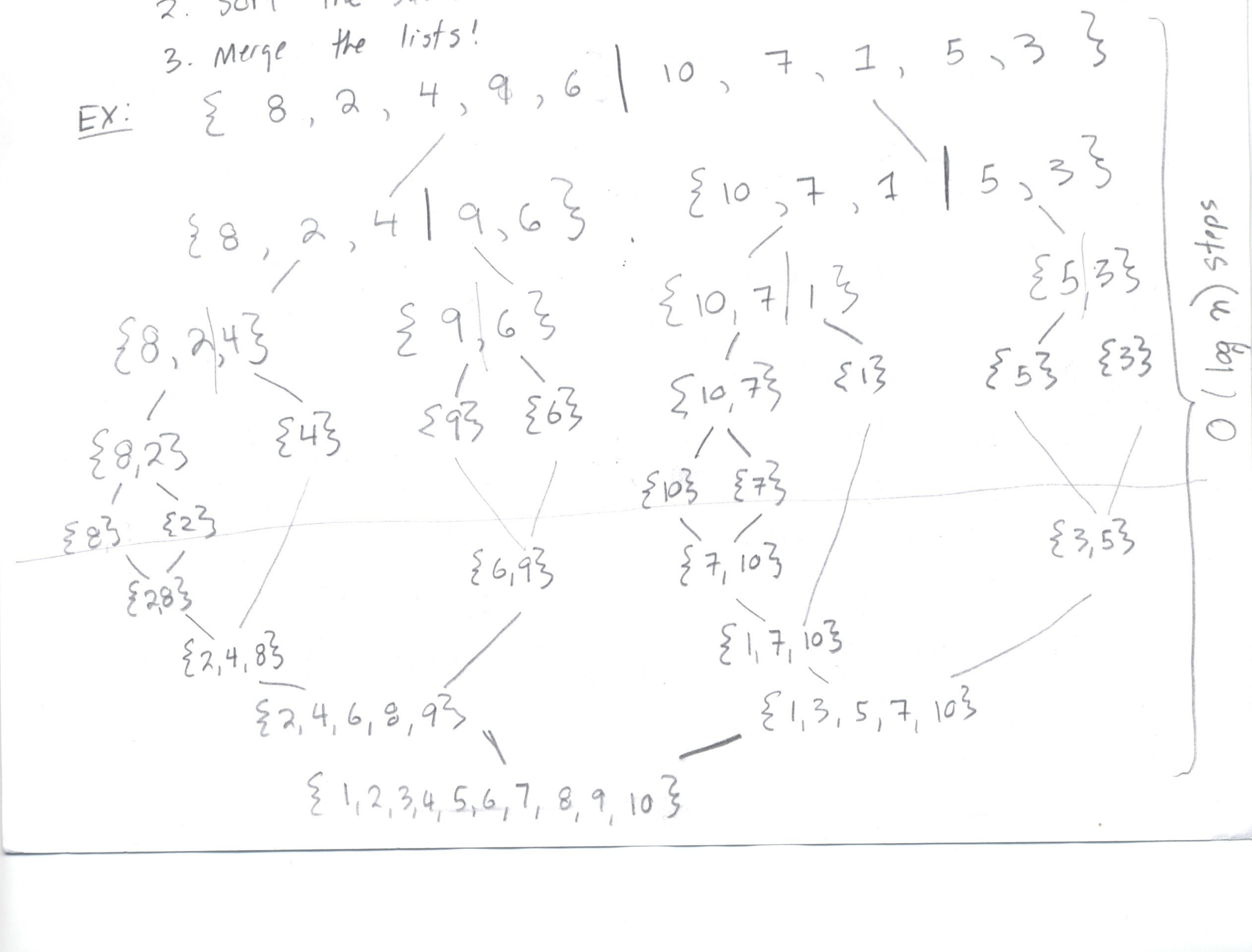
Recursive Algorithms: Mergesort

We saw two algorithms Bubblesort & Insertion Sort that require $O(n^2)$ time to sort a list of n items. This is a poor result, the best sorting algorithms sort a general list of n items in $O(n \cdot \log n)$ time.

The key idea of merge sort is that it is very easy to sort two (or more) already sorted lists into a single sorted list.

So given an unsorted list
1. subdivide list into two lists.
2. sort the sublists recursively
3. Merge the lists!

EX: $\{8, 2, 4, 9, 6 \mid 10, 7, 1, 5, 3\}$



Pseudocode

Mergesort ($L = a_1 a_2 a_3 \ldots a_n$)

in $n > 1$ then

$L_1 \leftarrow a_1 a_2 \ldots a_{\lceil \frac{n}{2} \rceil}$ ← example uses $\lceil \frac{n}{2} \rceil$.

$L_2 \leftarrow a_{\lceil \frac{n}{2} \rceil + 1} \ldots a_n$

return Merge (Mergesort (L_1), Mergesort (L_2))

Merge (L_1, L_2)

$L \leftarrow$ empty

while L_1 and L_2 are both unempty do

remove the smallest element of L_1 or L_2 and append to L.

if L_1 is empty then

add rest of L_2 to L.

else add rest of L_1 to L.

return L.

So, how many comparisons?

Consider Merge (L_1, L_2)

leave one in each list	$\lvert L_1 \rvert + \lvert L_2 \rvert - 2$
last removal	$+1$
total :	$\lvert L_1 \rvert + \lvert L_2 \rvert - 1$

(this is of course a worst case example).

What about for running the entire algorithm.

We will assume, in order to keep things simple that $n = 2^m$. - $k = 1$

- At the first stage we get two sublists of size 2^{m-1} $k = 2$
- At the second " " " four sublists of size 2^{m-2} $k = 3$

- At the m'th stage we get $\underline{\underline{2^m}}$ sublists of size $\underline{\underline{1}} = 2^{m-m} = 2^0$

▷ In general, 2^{k-1} lists at level $k-1$, each with 2^{m-k+1} elements which are split into 2^k lists of 2^{m-k} elements at level k.

Since we are only counting comparisons (between elements) we have not incurred any cost yet, so now we start merging.

At level k, ($k = m, m-1, m-2, \ldots$) 2^k lists each with 2^{m-k} elements are merged into 2^{k-1} lists with 2^{m-k+1} elements. at level $k-1$.

↳ Each merge takes: $2^{m-k} + 2^{m-k} - 1$ comparisons

$= 2^{m-k+1} - 1$ comparisons

Now we sum over levels 1 to m.

$$\sum_{k=1}^{m} (2^{k-1})(2^{m-k+1} - 1) = \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1}$$

since $(2^{k-1})(2^{m-k+1} - 1) = 2^m - 2^{k-1}$

$= m \cdot 2^m - (2^m - 1)$

Since m terms each of 2^m elements, & $2^m - 1$ from formula for geometric progression (p. 153).

$= \lg n \cdot n - (n - 1)$

$= n \log n - n + 1$

Counting

Suppose you want to estimate how secure a computer system is. You might need to know how strong a password is — this will depend on how likely an attacker is to randomly pick a correct password, which in turn is dependent on how many passwords are possible. So given our password policy what is the total number of passwords the attacker might have to guess from.

The Product Rule

Suppose we want to perform a procedure that can be broken down into 2 different tasks; the first can be accomplished in n_1 ways, and the second in n_2 ways. Then the whole procedure can be performed in $n_1 \cdot n_2$ ways.

Ex: You need to connect one of 25 computers to a router with 4 ports. How many ways can this be done?

task 1: pick a computer (25 ways)
task 2: pick a router (4 ways)

∴ By the product rule we use $25 \times 4 = 100$ ways.

Ex: How many bit strings are there of length 3.

pick a bit for position 1 in 1 of 2 ways.
" " " " " 2 in 1 of 2 "
" " " " " 3 " 1 " 2 "

$2 \cdot 2 \cdot 2 = 2^3$ bit strings of length 3.

In general there are 2^n bitstrings of length n.
(This is why 32-bit systems can address
2^{32} bits = 4 GB of RAM)

Ex: How many subsets are there for a set of size 'n'.
To create a subset decide if each element is part of it, or not:

element 1 in or not 2ways
" 2 " " " 2ways
⋮
" n in or not 2ways
} n

$\underbrace{2 \cdot 2 \cdot 2 \cdots 2}_{n} = 2^n$

Ex: How many passwords are there of length 6 that consist of lowercase latters.

$26 \times 26 \times 26 \times 26 \times 26 \times 26 = 26^6 \approx 300$ millicn.

What if we allow either case:

$52 \times 52 \times 52 \times 52 \times 52 \times 52 = 52^6 \approx 20$ billion.

(This is why we use both for secure passwords).

Ex: How many functions from a set of m elements to a set of n elements.

- for each element of the domain pick one element from the codomain.

$$\underbrace{n \cdot n \cdot n \cdot n \cdots n}_{m} = n^m$$

Ex: How many injective functions (one-to-one) are there from the set of m elements to a set on n elements.

① If $m > n$ then there are no such injective functions.

② Assume $m \leq n$

Assume domain $= \{a_1, a_2 \cdots, a_m\}$

$f(a_1)$ could be anything in the codomain: n
$f(a_2)$ could be anything except $f(a_1)$: $n-1$
$f(a_3)$ " " " " $f(a_1)$ or $f(a_2)$: $n-2$

⋮

$f(a_i)$ could be anything except $f(a_j)$: $n-i+1$ ways.
where $j < i$

Thus i can be as large as m so total is:

$$n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \ldots (n-m+1)$$

$$= n! - (n-m)!$$

The Sum Rule

Sometimes tasks <u>cannot</u> be done at the same time. If there are n_1 ways to do the first task and n_2 ways to do the second, there are $n_1 + n_2$ ways to do <u>one</u> of them.

<u>Ex</u>: Suppose I want to pick a winner of a scholarship from among 12 CS students and 17 Math students, I would be pick from.

$$12 + 17 = 29 \text{ students}$$

The sum rule also generalizes to multiple tasks.

<u>Ex:</u> 12 CS Students, 17 Math Students, 9 Engineering students.

$$12 + 17 + 9 = 38 \text{ students.}$$

More Complex Examples

In a version of the BASIC programming language variables are either 1 or 2 alphanumeric characters long, upper and lowercase characters are not distinguished. The variable must begin with a letter and cannot be one of the 5 2-character strings for reserved words.

Case I: 1 alphanumeric

26 ways (no digits)

Case II: 2 alphanumeric

$26 \cdot 36 = 936 - 5$

↑ no digits ↑ all alpha-numerics

$= 931.$

Case I & II are distinct so

$26 + 931 = 957$ ways.

Ex: Pick a password

- 6-8 characters
- all uppercase and digits accepted / unique.
- must have at least <u>one</u> digit.

How many?

Let P = total number of such passwords.

P_6, P_7, P_8 = total number of such passwords of length 6, 7, 8.

By sum rule $P = P_6 + P_7 + P_8$.

P_6 - tough to find directly, but we could instead find the number of all passwords (assuming one digit is not necessary) and subtract the number that don't satisfy the restriction.

- # of six-char strings is 36^6
- # of strings with no digits is 26^6

So $P_6 = 2,176,782,336 - 308,915,772$

$= 1,867,866,560$

Similarly

$P_7 = 70,332,353,920$

$P_8 = 2,612,282,842,880$

So $P_6 + P_7 + P_8 = 2,684,483,063,360.$

The technique used above is very important.

- Split up into exclusive cases (Sum Rule).
- try to use the product in each case:

If it still doesn't work

try counting counting the complimend.

no restrictions − # not satisfying the restrictions.

The Inclusion-Exclusion Principle.

The sum rule requires that the two tasks cannot be done at the same time, but sometimes we might like them to. What do we do?

Ex: How many bitstrings of length 8 start with a 1 or end with 00?

Note: these could happen at the same time, so we can't just sum two cases together.

Idea: if we did sum them directly, we would count the strings that start with a 1 and end with 00 twice (once each time). We only want to count them once, so subtract this number (the duplicates) at the end.

Start with 1: 1 ∘ (pick 7 remaining bits in 2^7 ways) $= 2^7$

End with 00: (pick 6 bits is 2^6 ways) 00 $= 2^6$

Start with 1 and end with 00

1 ∘ (pick 5 bits, 2^5 ways) 00 $= 2^5$

So total $= 2^7 + 2^6 - 2^5 = 128 + 64 - 32 = 160$.

This is the idea of inclusion - exclusion?

Inclusion / Exclusion

If A_1 is a set and A_2 is a set and T_1 is a task of picking from A_1, and T_2 is a task of picking from A_2, the number of ways to do either T_1 or T_2 is:

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$$

Ex: How many positive integers ≤ 1000 are divisible by 7 or 11.

A: Set of all positive ints ≤ 1000 div by 7.

B: _____ by 11.

We want $|A \cup B|$

$$|A| = \left\lfloor \frac{1000}{7} \right\rfloor, \quad |B| = \left\lfloor \frac{1000}{11} \right\rfloor$$

Since 7 and 11 have no common factors any integer divisible by 7 and 11 must be divisible by $7 \cdot 11$ so

$$|A \cap B| = \left\lfloor \frac{1000}{7 \cdot 11} \right\rfloor \quad \text{and}$$

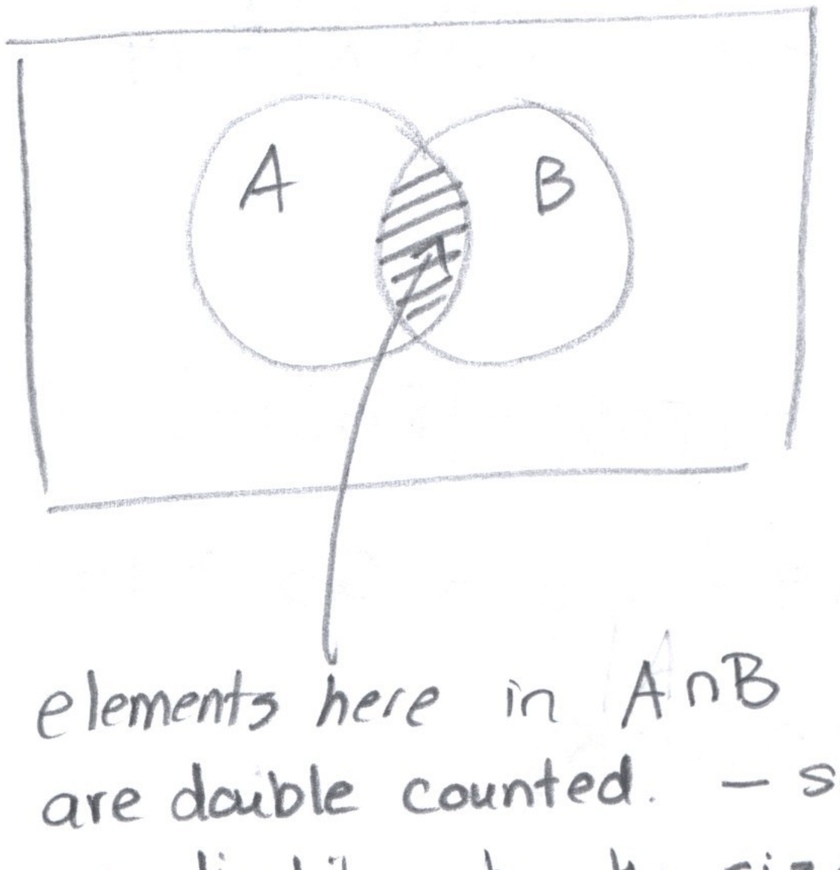
$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$= \left\lfloor \frac{1000}{7} \right\rfloor + \left\lfloor \frac{1000}{11} \right\rfloor - \left\lfloor \frac{1000}{77} \right\rfloor$$

$$= 142 + 90 - 12$$

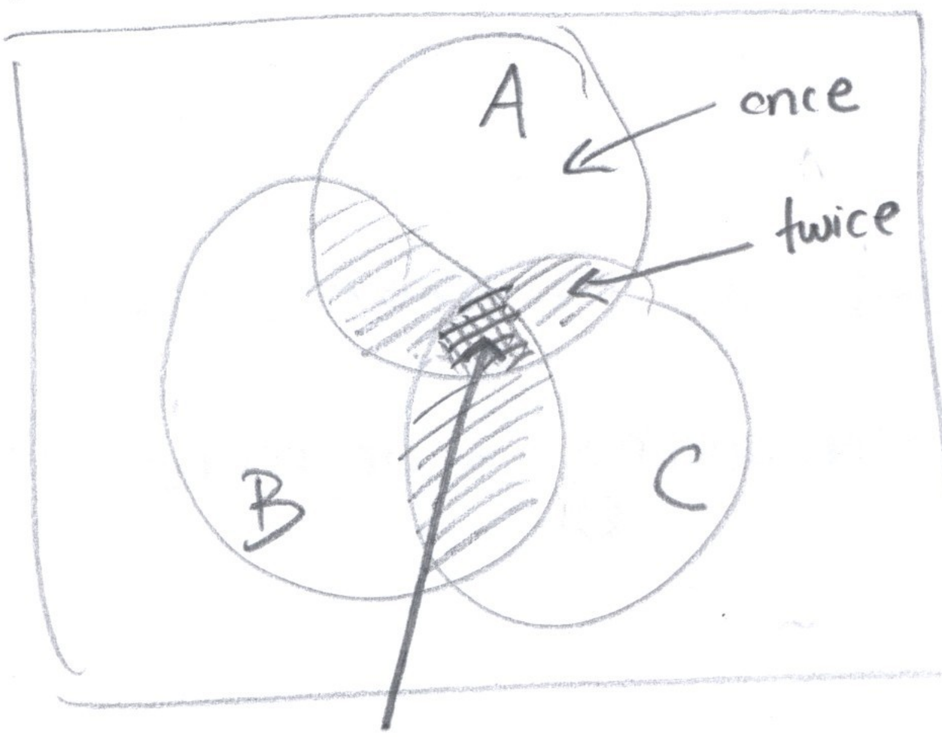
$$= 220.$$

Visually, what is happening.



elements here in $A \cap B$ are double counted. — so we must reduce the cardinality by the size of this intersection.

What about 3 sets?



$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$ since we have now removed all reference to the area overlapped by all three.

Ex: Students in language classes.

1232 French
879 Spanish
114 Russian
103 French & Spanish
14 Spanish & Russian
23 French & Russian

2092 Have taken at least one of Spanish, French, Russian.

$|F| = 1232 \quad |S| = 879 \quad |R| = 114$

$|F \cap S| = 103 \quad |F \cap R| = 23 \quad |S \cap R| = 14$

$|F \cup S \cup R| = 2092$

$= |F| + |S| + |R| - |F \cap S| - |F \cap R| - |S \cap R| + |F \cap S \cap R|$

$= 1232 + 879 + 114 - 103 - 23 - 14 + |F \cap S \cap R|$

$2092 = 2085 + |F \cap S \cap R|$

$|F \cap S \cap R| = 7$

The Pigeonhole Principle

If $k+1$ objects are placed into k boxes, then there is at least one box containing 2 or more objects.

Ex: Among 367 people, $\geq$ must have the same birthday.

Ex: In a group of 27 words, at least two must begin with the same letter.

Ex: How many students need to be in a class to ensure that at least 2 have the same grade, if grades are integers in the range from 0 to 100?

- 101 possible grades, so we must have 102.

The pigeonhole principle generalizes to N objects and k boxes as follows:

- If N objects are placed into k boxes, then there is at least one box containing at least $\lceil \frac{N}{k} \rceil$ objects.

Ex: Among 100 people, at least $\lceil \frac{100}{12} \rceil = 9$ were born in the same month.

Ex: How many students must we have to ensure ≥ 6 students get the same letter grade, $\{A, B, C, D, F\}$. are the possibilities.

- The smallest N such that $\lceil \frac{N}{5} \rceil = 6$, so the smallest such N is $5 \cdot 5 + 1 = 26$.

(if only 5 we could have 5 with each grade).

Ex: Let $A = \{1, 2, 3, \ldots, 14\}$. Prove that if I selected 9 elements from A, then at least two pairs sum to exactly 15. [Note - I am selecting 9 distinct elements].

Consider pairs adding to 15: (there are 7)

1, 14 2, 13 3, 12 4, 11 5, 10 6, 9 7, 8

We select 9 elements, from 7 pairs, so we must have selected one pair. Now we have 6 pairs remaining and 7 elements left to choose, so again we must select one of the pairs.

[Note: with each step we eliminate 1 pair and 2 elements.]

Example: Show that among any $n+1$ positive integers not exceeding $2n$ there must be an integer that divides one of the integers.

Let $a_1, a_2, a_3, \ldots a_{n+1}$ be the set of integers.

We express each integer as a power of 2 times an odd integer.

$$a_i = 2^{k_i} \cdot q_i$$

where k_i is a non-negative integer and q_i is odd.

Eg.

$$28 = 2^2 \cdot 7$$
$$35 = 2^0 \cdot 35$$
$$64 = 2^6 \cdot 1$$

The integers $q_1, q_2, q_3 \ldots q_{n+1}$ are all odd integers $< 2n$ (since $2^{k_i} \geq 1$). There are only n such odd integers $< 2n$.

Since this is the case, by the 'pigeonhole principle there are two q_s q_i and q_j such that $q_i = q_j = q$

So $a_i = 2^{k_i} q$ and $a_j = 2^{k_j} q$

Now if $k_i < k_j$ then a_i divides a_j, else if $k_j < k_i$ then a_j divides a_i.

Eg. $2^2 \times 7 = 28$ .. $2^3 \times 7 = 56$ .. $2^4 \times 7 = 112$ . . . etc.
($56/28 = 2$) ($112/28 = 4$)