

## CS 5602 Introduction to Cryptography Lecture 03 Mathematical Preliminaries I (Stuff You Mostly Know)

George Markowsky  
Computer Science Department  
Missouri University of Science & Technology

1

## Describing Sets

- List all elements  $\{1, 2, \{3, 4\}\}$ 
  - **Singleton sets** are sets that have exactly one element, such as  $\{1\}$ ,  $\{2\}$ , etc.
- Use Python lists to represent sets  $[1, 2, [3, 4]]$
- $\{x \mid x \text{ has property } P\}$ 
  - $\{n \mid n \text{ is a positive prime}\}$
- Python version  $[i \text{ for } i \text{ in range}(7) \text{ if } 0 == i \% 2]$ 
  - **List comprehensions**
- Sets can be members of sets to any depth

4

## Goal of Lectures 3 & 4

- These lectures aim to quickly review the material in Appendix A of the textbook on Mathematical Terminology
- It also covers some mathematics and not just the terminology
- Will review very quickly – you are responsible for reading all the slides and making sure you are acquainted with the concepts described
- Lecture 3 will go through stuff you should have covered in undergraduate courses
- Lecture 4 will go through stuff that you most likely have not covered in undergraduate courses
- I will post a book I wrote that covers most of Lecture 3

2

## The Empty Set

- **Definition:** The **empty set** is the set that contains no members.
- The empty set is often written as  $\{\}$  or  $\emptyset$
- In Python we would represent it as  $[]$ 
  - Note that Python uses  $\{\}$  for dictionaries, which are quite different from sets
  - Be careful when switching between mathematics and Python

5

## Sets

- **Definition:** A **set** is a collection of objects called **elements** or **members**.
- In mathematics, sets are accepted as primitive objects that cannot be defined in terms of other objects.
- There are several ways to describe a set.

3

## Universal Sets

- As we shall soon see, **there is no such thing as the set of all sets!**
- However, for set theory problems there is generally a set that contains all elements of interest for the problem
- This is called the **universal set** for the problem
- **Different problems have different universal sets**

6

## Set Membership

- The Greek letter  $\in$  (**epsilon**) is used to denote membership in a set
- Thus,  $x \in S$ , means that the element  $x$  is in the set  $S$
- Python uses **in** to test whether something belongs to a list and we will use that when modeling sets by Python lists
- The symbol  $\notin$  means not in a set
  - In Python we can write **x not in S**
- $\in$  has very high precedence, even higher than not

7

## Python Sets

- Python has two built-in set types: sets and frozen sets
- Python sets are not as general as the sets we consider
  - in particular, a Python set cannot contain a Python set as a member
- For this reason, we mostly focus on our list representation of sets
- Frozen sets are sets that cannot be changed
- Frozen sets are to sets how tuples are to lists
- Because it is a built-in type, can often speed algorithms up when looking for distinct elements

10

## Set Equality

- **Set Equality Postulate:** Two sets are **equal** if and only if they have the same elements.
- Thus,  $X = Y$  iff  $\forall x(x \in X \leftrightarrow x \in Y)$
- In general, to prove two sets are equal is done in two parts
- First, we prove that if  $x \in X$  then  $x \in Y$
- Second, we prove that if  $x \in Y$  then  $x \in X$

8

## Python Sets

```
>>> empty = set()
>>> X = { 1, 2, 8}
>>> print len(empty)
0
>>> print len(X)
3
>>> Y = { 78, 2, 3, 2}
>>> print Y
set([2, 3, 78])
>>> print X.union(Y)
set([1, 2, 3, 8, 78])
>>> for x in X:
    print x
8
1
2
```

This is how to generate the empty set

len() gives size

Duplicated elements are eliminated

print works for sets

Unusual syntax for union

For loops work with sets!  
Note that elements in a set are in no particular order!

11

## Set Equality in Python

- Sets do not have the concept of order or multiplicity
- For example, the set  $\{1,2\}$  and the set  $\{2,1,2\}$  are identical because  $\{1,2\}$  and  $\{2,1,2\}$  have exactly the same elements
- Of course, in Python  $[1, 2]$  and  $[1, 2, 1]$  are very different lists

9

## Things That Don't Work With Python Sets

```
>>> Z = { 1, 2, X, 3}
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    Z = { 1, 2, X, 3}
TypeError: unhashable type: 'set'
>>> type(empty)
<type 'set'>
>>> type({})
<type 'dict'>
>>> type(X)
<type 'set'>
>>> X.append(99)
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    X.append(99)
AttributeError: 'set' object has no attribute 'append'
```

Can't embed one set into another!

{ } is always the empty dictionary and never the empty set. set() generates the empty set.

append does not work with sets – use union

12

### More On Python Sets

```
>>> 78 in Y
True
>>> 78 not in Y
False
>>> Y.add(99)
>>> Y
set([99, 2, 3, 78])
>>> Y.remove(99)
>>> Y
set([2, 3, 78])
>>> Y.add(2)
>>> Y
set([2, 3, 78])
```

Same in as for lists

add instead of append – does not return a value

If an element is already in the set, it cannot be added again

13

### Warning

- Unfortunately, the textbook uses an older notation that I intensely dislike
- The book uses  $\subset$  instead of  $\subseteq$  and  $\subsetneq$  instead of  $\subset$
- I will only use  $\subset$  and  $\subseteq$  for proper subset and subset
- One problem with  $\subsetneq$  is that it is hard to produce

16

### Frozen Sets

```
>>> Z = frozenset(Y)
>>> Z
frozenset([2, 3, 78])
>>> Z.add(99)

Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    Z.add(99)
AttributeError: 'frozenset' object has no attribute 'add'
```

14

### Supersets

- Is A is a subset of B, we say that B is a **superset** of A.
- The relation  $\subseteq$  is called **set containment** or **set inclusion**
- The relation  $\subset$  is call **proper set containment** or **proper set inclusion**
- The notation  $A \not\subseteq B$  means that A is not a subset of B

17

### Subsets

- Definition:** If the elements of a set A are all elements of a set B, then we say that A is a **subset** of B. Equivalently, we say that B **contains** A.
- We use the notation  $A \subseteq B$  to indicate that A is a subset of B or that B contains A.
- If A is a subset of B that is not equal to B, we call A a **proper subset** of B and write  $A \subset B$  (note the analogy between  $\leq$  and  $<$ )

15

### Theorem

- Let U be a set and A, B and C subsets of U. Then the following are true where the quantifiers have U as the universe.
- $A \subseteq B$  iff  $\forall x (x \in A \rightarrow x \in B)$
  - $\forall x (x \in A \ \& \ A \subseteq B \rightarrow x \in B)$
  - $(A \subseteq B \ \& \ B \subseteq A \leftrightarrow A = B)$
  - $A \subseteq B \ \& \ B \subseteq C \rightarrow A \subseteq C$
  - $A \neq B$  iff  $\exists x (x \in A \ \& \ x \notin B)$
  - $A = B$  iff  $\forall x (x \in A \leftrightarrow x \in B)$

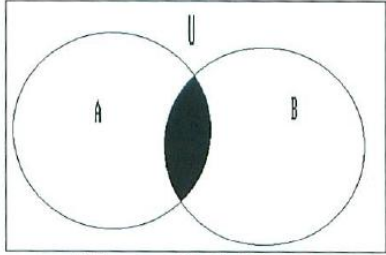
18

### Cardinality of Sets

- **Definition:** The **cardinality** of a set  $A$ , denoted by  $|A|$ , is the number of elements that the set contains.
- Cardinality can also be applied to infinite sets, but we will discuss this point later.
- The empty set has cardinality 0

19

### Set Operations: Intersection



Related to AND

$A \cap B$  is represented by the filled area.

$\forall x (x \in A \cap B = x \in A \ \& \ x \in B)$

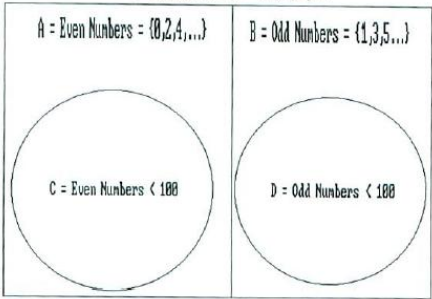
22

### Venn Diagrams

$U = \text{Natural Numbers} = \{0, 1, 2, \dots\}$

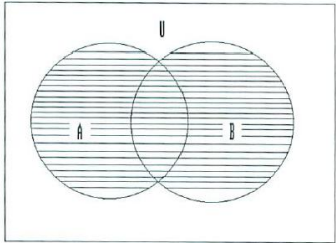
$A = \text{Even Numbers} = \{0, 2, 4, \dots\}$        $B = \text{Odd Numbers} = \{1, 3, 5, \dots\}$

$C = \text{Even Numbers} < 100$        $D = \text{Odd Numbers} < 100$



20

### Set Operations: Union



Related to OR

$A \cup B$  is represented by the shaded area.

$\forall x (x \in A \cup B = x \in A \mid x \in B)$

23

### A More Complicated Venn

$U = \text{Natural Numbers} = \{0, 1, 2, \dots\}$

$A = \text{Even Numbers} = \{0, 2, 4, \dots\}$        $B = \text{Odd Numbers} = \{1, 3, 5, \dots\}$

$P = \text{Primes} = \{2\}$        $\{101, 103, \dots\}$

$C = \text{Even Numbers} < 100$        $D = \text{Odd Numbers} < 100$

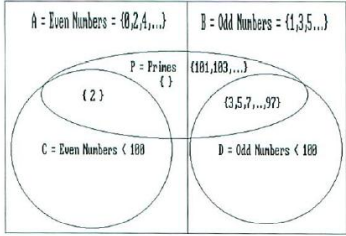
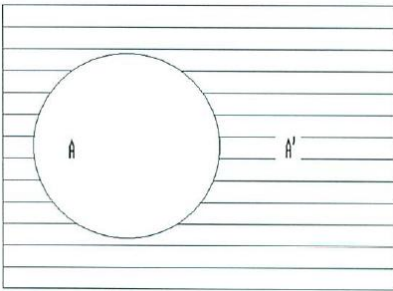


Figure 13.2 A more complicated Venn Diagram

21

### Set Operation: Complement



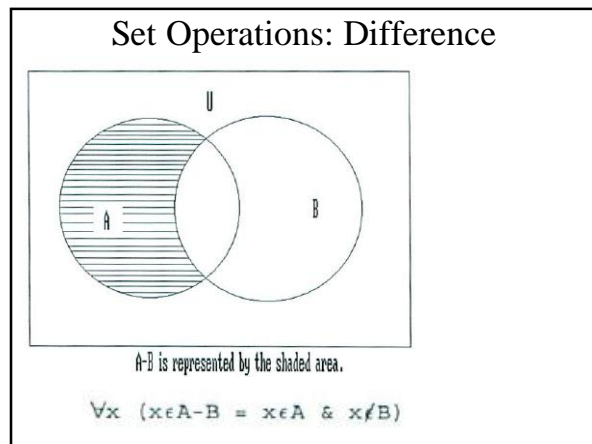
Related to NOT

Depends on U and not just on A

$A'$  is represented by the shaded area.

$\forall x (x \in A' = x \in U \ \& \ x \notin A)$

24

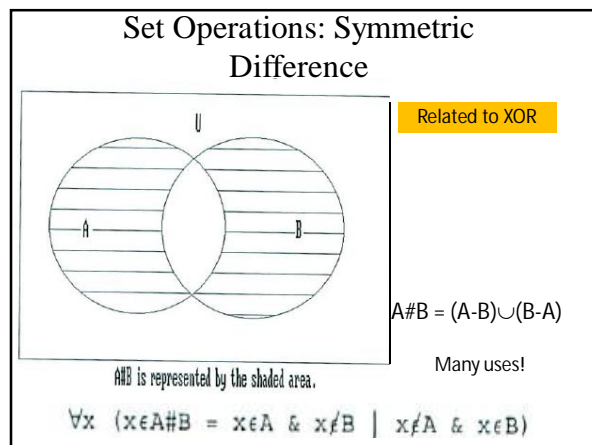


25

### Laws of Boolean Algebra for Sets Similar to Propositional Logic

- **Idempotent Laws**
- $P \cap P = P$  and  $P \cup P = P$
- **Commutative Laws**
- $P \cap Q = Q \cap P$  and  $P \cup Q = Q \cup P$
- **Associative Laws**
- $(P \cap Q) \cap R = P \cap (Q \cap R)$  and
- $(P \cup Q) \cup R = P \cup (Q \cup R)$

28

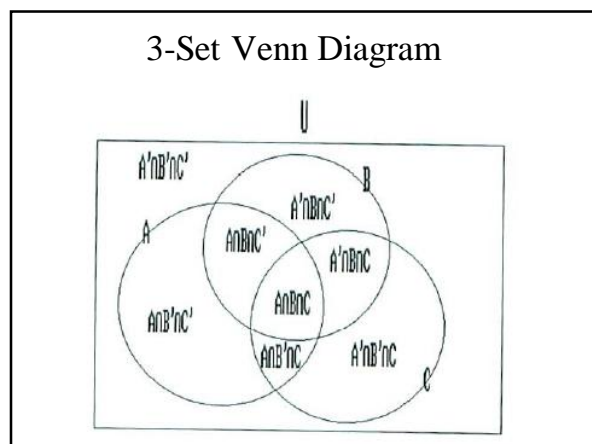


26

### Laws of Boolean Algebra for Sets

- **Identity Laws**
- $\emptyset \cap P = P \cap \emptyset = \emptyset$  and  $U \cap P = P \cap U = P$
- $\emptyset \cup P = P \cup \emptyset = P$  and  $U \cup P = P \cup U = U$
- **Complement Laws**
- $P \cap P' = P' \cap P = \emptyset$  and  $P \cup P' = P' \cup P = U$
- $\emptyset' = U$ ,  $U' = \emptyset$  and  $P'' = P$

29



27

### Laws of Boolean Algebra for Sets

- **Distributive Laws**
- $P \cap (Q \cup R) = (P \cap Q) \cup (P \cap R)$  and
- $P \cup (Q \cap R) = (P \cup Q) \cap (P \cup R)$
- **DeMorgan's Laws**
- $(P \cap Q)' = P' \cup Q'$  and  $(P \cup Q)' = P' \cap Q'$

30

## Laws of Boolean Algebra for Sets

- **Absorption Laws**
- $P \cap (P \cup Q) = P$
- $P \cup (P \cap Q) = P$
- How do you prove these laws?
- Since we have expressed all the membership information in terms of predicates you can just use Boolean Algebra for Logic

31

## The Associative Law

- To prove that  $A \cap (B \cap C) = (A \cap B) \cap C$  requires showing that
- $\forall x (x \in A \cap (B \cap C) \Rightarrow x \in (A \cap B) \cap C)$
- is true. To prove this, replace all occurrences of  $\cap$  by the corresponding expressions involving  $\&$ . This yields
- $\forall x (x \in A \& (x \in B \& x \in C) \Rightarrow (x \in A \& x \in B) \& x \in C)$
- which is just the Associative Law for  $\&$
- The proof for  $\cup$  is similar and uses the Associative Law for  $\mid$

34

### Idempotent Laws

$$A \cap A = A \quad A \cup A = A$$

### Commutative Laws

$$A \cap B = B \cap A \quad A \cup B = B \cup A$$

### Associative Laws

$$(A \cap B) \cap C = A \cap (B \cap C) \quad (A \cup B) \cup C = A \cup (B \cup C)$$

### Identity Laws

$$A \cap \{ \} = \{ \} \quad A \cup U = A \quad A \cap U = A \quad A \cup \{ \} = A$$

### Complement Laws

$$A \cap A' = \{ \} \quad A \cup A' = U$$

### Distributive Laws

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

### DeMorgan's Laws

$$(A \cap B)' = A' \cup B' \quad (A \cup B)' = A' \cap B'$$

### Absorption Laws

$$A \cap (A \cup B) = A \quad A \cup (A \cap B) = A$$

32

**Theorem 13.5 (First Distributive Law):** For any sets A, B and C,

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

**Proof:** To prove that  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$  requires showing that

$$\forall x (x \in A \cap (B \cup C) \Rightarrow x \in (A \cap B) \cup (A \cap C))$$

is true. To prove this, replace all occurrences of  $\cap$  and  $\cup$  by the corresponding expressions involving  $\&$  and  $\mid$ . This yields the expression

$$(*) \quad \forall x (x \in A \& (x \in B \mid x \in C) \Rightarrow (x \in A \& x \in B) \mid (x \in A \& x \in C))$$

It should be clear that  $(*)$  is true, because  $\&$  distributes over  $\mid$ . As before,  $(*)$  can be shown to be true using the truth tree method.

Since an element of one set is always an element of the other, it follows from the Set Equality Axiom that the two sets,  $A \cap (B \cup C)$  and  $(A \cap B) \cup (A \cap C)$ , are equal. ■

35

## The Commutative Laws

- **Theorem:** For all sets,  $A \cap B = B \cap A$  and  $A \cup B = B \cup A$ .
- **Proof:** We need to show that
  - $\forall x (x \in A \cap B \text{ iff } x \in B \cap A)$  which is equivalent to
  - $\forall x (x \in A \& x \in B \text{ iff } x \in B \& x \in A)$
  - The previous step is true because  $\&$  is commutative
  - Similarly,  $\cup$  is commutative because  $\mid$  is commutative

33

Suppose that A and B are subsets of a universal set U and it is desired to prove that  $A \subseteq B$ . One way to prove this is to show that

$$\forall x (x \in A \rightarrow x \in B)$$

is a tautology. As statements get more complicated, it is clumsy to operate with them. For this reason, the following slightly different approach is often taken in set theory proofs:

1. The universal quantifier is not mentioned explicitly.
2. It is assumed that x is an arbitrary element in A.
3. It is shown that x must be in B.
4. Since x is arbitrary (this means that nothing special can be assumed about x) it follows that  $\forall x (x \in A \rightarrow x \in B)$  is true.

36

## First of DeMorgan's Laws

True Expressions	Reason
$x \in (A \cap B)'$	Given
$= x \in U \ \& \ \neg(x \in A \cap B)$	Definition of '
$= x \in U \ \& \ \neg(x \in A \ \& \ x \in B)$	Definition of $\cap$
$= x \in U \ \& \ (\neg(x \in A) \mid \neg(x \in B))$	DeMorgan's Law for '
$= (x \in U \ \& \ \neg(x \in A)) \mid (x \in U \ \& \ \neg(x \in B))$	Distributive Law of Logic
$= x \in A' \mid x \in B'$	Definition of '
$= x \in A' \cup B'$	Definition of $\cup$

37

## Cartesian Product

- Definition:** Given two sets A and B, the **Cartesian product** of A and B, written  $A \times B$ , is the set consisting of all ordered pairs whose first element belongs to A and whose second element belongs to B. A and B are called the **factors** of  $A \times B$ .

40

## Rules for Difference & Symmetric Difference

- $A \# A = \emptyset$
- $A \# \emptyset = A$
- $A \# U = A'$
- $A \# B = B \# A$
- $A - B = A \cap B'$
- $A \# B = (A - B) \cup (B - A) = A \cap B' \cup A' \cap B$

**A#B is the set equivalent of XOR**

38

## Cartesian Product

- Symbolically, a Cartesian product can also be described by  $\{ (a,b) \mid a \in A \ \& \ b \in B \}$
- Recall that ordered pairs have the property that  $(a,b) = (c,d)$  if and only if  $a=c$  and  $b=d$ .
- In general, the sets  $A \times B$  and  $B \times A$  are quite different.
- The only time they are equal is when  $A = B$ .

41

## Associativity of #

$$\begin{aligned}
 (A \# B) \# C &= (A \# B) \cap C' \cup (A \# B)' \cap C = (A \cap B' \cup A' \cap B) \cap C' \cup (A \cap B' \cup A' \cap B)' \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup (A' \cup B) \cap (A \cup B') \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup (A' \cup B) \cap (A \cup B') \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup (A' \cap A \cup B \cap A \cup A' \cap B' \cup B \cap B') \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup (\emptyset \cup B \cap A \cup A' \cap B' \cup \emptyset) \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup (B \cap A \cup A' \cap B') \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup B \cap A \cap C \cup A' \cap B' \cap C \\
 &= A \cap B' \cap C' \cup A' \cap B \cap C' \cup A \cap B \cap C \cup A' \cap B' \cap C \\
 \\
 A \# (B \# C) &= A \cap (B \# C)' \cup A' \cap (B \# C) = A \cap (B \cap C' \cup B' \cap C)' \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap ((B' \cup C) \cap (B \cup C')) \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap (B' \cup C) \cap (B \cup C') \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap (B' \cap B \cup B' \cap C' \cup C \cap B \cup C \cap C') \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap (\emptyset \cup B' \cap C' \cup C \cap B \cup \emptyset) \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap (B' \cap C' \cup C \cap B) \cup A' \cap (B \cap C' \cup B' \cap C) \\
 &= A \cap B' \cap C' \cup A \cap C \cap B \cup A' \cap B \cap C' \cup A' \cap B' \cap C \\
 &= A \cap B' \cap C' \cup A \cap B \cap C \cup A' \cap B \cap C' \cup A' \cap B' \cap C
 \end{aligned}$$

39

## Example

Let  $A = \{1, 2, 3\}$  and  $B = \{1, 3, 5\}$ . Then

$$\begin{aligned}
 A \times B &= \\
 \{ (1,1), (1,3), (1,5), (2,1), (2,3), (2,5), (3,1), \\
 (3,3), (3,5) \}
 \end{aligned}$$

$$\begin{aligned}
 B \times A &= \\
 \{ (1,1), (1,2), (1,3), (3,1), (3,2), (3,3), (5,1), \\
 (5,2), (5,3) \}
 \end{aligned}$$

42

## Larger Cartesian Products

- **Definition:** Let  $S = \{A_1, A_2, \dots, A_n\}$  be a family of sets. The **Cartesian product** of  $S$ , written  $\times S$  or  $A_1 \times A_2 \times \dots \times A_n$ , is the set of all ordered  $n$ -tuples, where the  $i$ -th element belongs to  $A_i$ . The  $A_i$  are called the **factors** of  $\times S$ .

43

## Cartesian Product

- You can think of Cartesian products as definitions of record or struct types.
- Cartesian products are the basis for relational database theory.

46

## Cartesian Product

- The Cartesian Product of a family of sets can be described by
- $\{ (a_1, a_2, \dots, a_n) \mid \forall i (a_i \in A_i) \}$

```
def Cartesian(S1,S2):
    cart = []
    for s in S1:
        for t in S2:
            cart.append((s,t))
    return cart
```

```
print Cartesian(range(3),range(4))
```

```
[(0, 0), (0, 1), (0, 2), (0, 3),
 (1, 0), (1, 1), (1, 2), (1, 3),
 (2, 0), (2, 1), (2, 2), (2, 3)]
```

44

## First Counting Theorem

- $|A_1 \times A_2 \times \dots \times A_n| = |A_1| \times |A_2| \times \dots \times |A_n|$
- You can generate the Cartesian product just by using the correct number of for loops

47

Let  $A = \{1,2\}$ ,  $B = \{c,d\}$ ,  $D = \{?, \$\}$ . Then

$A \times B \times C =$   
 $\{(1,c,?), (1,c,\$), (1,d,?), (1,d,\$), (2,c,?), (2,c,\$), (2,d,?), (2,d,\$)\}$

$A \times (B \times C) =$   
 $\{(1,(c,?)), (1,(c,\$)), (1,(d,?)), (1,(d,\$)), (2,(c,?)), (2,(c,\$)), (2,(d,?)), (2,(d,\$))\}$

Strictly speaking,  $A \times B \times C \neq A \times (B \times C)$  because the first set consists of ordered 3-tuples, while the second set consists of ordered pairs. People generally use sets of the form  $A \times B \times C$  rather than sets that look like  $A \times (B \times C)$ .

45

## Power Sets

- **Definition:** Let  $S$  be a set. The power set of  $S$  is the set of all subsets of  $S$ .
- In mathematics, the power set of  $S$  is usually denoted by  $2^S$ . Symbolically,  $2^S$  can be described as

$$\{ A \mid A \subseteq S \}$$

- Recall that  $|2^S| = 2^{|S|}$

48



### Example

- Let  $S = \{a,b,c\}$ . Then  $2^{\{a,b,c\}} = \{\{\}, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$
- Note that  $S$  has 3 elements and  $2^S$  has 8 elements, which is  $2^3$ .
- In general,  $|2^S| = 2^{|S|}$
- This justifies the use of  $2^S$  to denote the power set of  $S$ .

49

```
def EqualSets(S1,S2):
    return subset(S1,S2) and subset(S2,S1)

def subset(X,Y):
    if 0 == len(X):
        return True
    if 0 == len(Y):
        return False
    return element(X[0],Y) and subset(X[1:],Y)

def element(e,X):
    if 0 == len(X):
        return False
    return equalElt(e,X[0]) or element(e,X[1:])
```

52

```
def Power(S):
    if 0 == len(S):
        return [ [] ]
    e = S[-1]
    P = Power(S[:-1])
    for X in P[:]:
        Y = X[:]
        Y.append(e)
        P.append(Y)
    return P

print Power(range(3))
```

```
[[], [0], [1], [0, 1], [2], [0, 2], [1, 2], [0, 1, 2]]
```

50

```
def equalElt(e1,e2):
    if type(e1) != type(e2):
        return False
    if e1 == e2:
        return True
    if type([ ]) == type(e1):
        return EqualSets(e1,e2)
    return False
```

```
print EqualSets([1, 2], [1, 2, 1, 2, 2, 1])
print EqualSets([1, [1,2], 2], [1, 2, 1, [2,1], 2, 2, 1])
```

```
True
True
```

53

### The Set Equality Problem

- Suppose you are using Python lists to represent sets and you want to determine whether two sets are equal
- For example,  $[1, 2]$  and  $[1, 2, 1, 2, 2, 1]$  are equal as sets but not as lists
- Furthermore,  $[1, [1,2], 2]$  and  $[1, 2, 1, [2,1], 2, 1]$  are equal as sets but not as lists
- How do you write a program that adjusts for nesting, multiplicity and different order?

51

### Relations

- Definition:** A binary relation,  $R$ , between two sets  $A$  and  $B$  is a subset of  $A \times B$ .  $R$  is called a **binary relation on A and B**. The set  $A$  is called the **domain** of  $R$ , while  $B$  is called the **codomain** of  $R$ .
- If  $A = B$ ,  $R$  is called a **binary relation on A**.
- In this case,  $A$  is called the **domain** of  $R$ . The sets  $A$  and  $B$  are called **factors** of  $R$ .

54

### Relations

- **Definition:** An **n-ary relation**,  $R$ , among the  $n$  sets  $A_1, A_2, \dots, A_n$ , is a subset of  $A_1 \times A_2 \times \dots \times A_n$ .  $R$  is called a relation on  $A_1, A_2, \dots, A_n$ . The  $A_i$  are called the **factors** of  $R$ .

55

### Relations and Predicates

- Relations are closely related to predicates and people sometimes do not distinguish between them.
- If  $R$  is a relation on the sets  $A_1, \dots, A_n$ , it can be represented by the predicate  $P$  where  $P(a_1, \dots, a_n)$  is True iff  $(a_1, \dots, a_n) \in R$ .

58

### Notation

- In this lecture we will use a slightly different notation from what was used earlier.
- Capital letters will represent sets and relations, while small letters from the beginning of the alphabet such as  $a, b$ , and  $c$  will represent constant elements from sets.
- Small letters from the end of the alphabet will continue to represent variables.
- Thus,  $x, y$  and  $z$  will be variables.

56

### Relations and Predicates

- Similarly, if  $P$  is a predicate, the universe of each variable is a set, so  $P$  can be represented by the relation  $R$  where  $(a_1, \dots, a_n) \in R$  iff  $P(a_1, \dots, a_n)$  is True.

59

### Example

- Let  $A = \{1, 2, 3\}$ .
- A binary relation on  $A$  is
- $R = \{(1, 2), (1, 3), (2, 3)\}$ .
- Note that this is the relation  $<$  restricted to the set  $A$
- In other words,  $R$  consists of all pairs  $(a, b)$  where  $a, b \in A$  and  $a < b$

57

### 0,1-Matrices and Boolean Matrices

- **Definition:** Let  $A$  and  $B$  be sets. A **0,1-matrix** or **Boolean matrix** on  $A$  and  $B$  is a rectangular arrangement of zeroes and ones such that the rows are indexed by the elements of  $A$  and the columns are indexed by the elements of  $B$ .

60

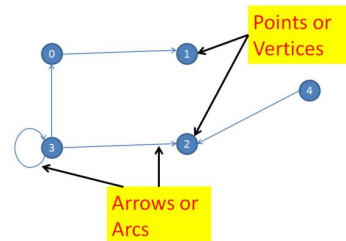
### Relations and Matrices

- Let  $R$  be a binary relation on  $A$  and  $B$ , and  $M$  a 0,1-matrix on  $A$  and  $B$ .  $M$  is said to represent  $R$  if  $\forall x \forall y ((M[x,y] = 1) \text{ iff } xRy)$  is true.
- In other words, for any pair  $(a,b)$  such that  $aRb$ ,  $M[a,b] = 1$ .
- If  $\neg aRb$ , then  $M[a,b] = 0$ .
- The universal quantifiers range over the sets  $A$  and  $B$  respectively.

61

### Example

- Let  $A = \{0,1,2,3,4\}$  and  $R$  be the binary relation defined on  $A$  which is given by  $\{(0,1), (3,0), (3,2), (3,3), (4,2)\}$ .



64

### Example

- Let  $A = \{0,1,2,3,4\}$  and  $R$  be the binary relation defined on  $A$  which is given by  $\{(0,1), (3,0), (3,2), (3,3), (4,2)\}$ .

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	1	0	1	1	0
4	0	0	1	0	0

62

### Undirected Graphs

- If the relation  $R$  happens to be **symmetric**, i.e., any two points in  $A$  have either no arrows between them, or have arrows going in both directions, you can reduce the number of arrows in the picture by
- combining the two arrows between each pair of points into a single undirected edge which is drawn without any indication of direction.
- Such a representation is called an undirected graph and may only be used to represent symmetric relations.

65

### Directed Graphs

- We use **nodes** or **vertices** to represent elements of a set and **arrows** or **arcs** to represent relationships
- We will give a more formal definition of a graph in a later lecture – for now we will use the idea informally
- Note that the relationships have a **direction**
- Directed graphs are often called **digraphs**

63

### Example

- Let  $A = \{1, 2, 3\}$  and  $R = \{(1,1), (1,2), (2,1), (1,3), (3,1)\}$ . Then  $R$  is symmetric and can be represented by an undirected graph.



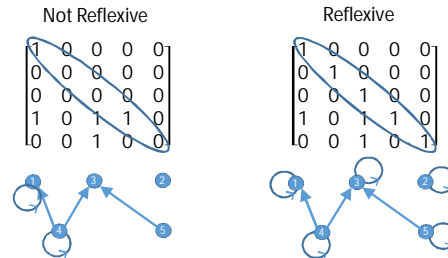
66

## Bipartite Graphs

- A binary relation  $R$  can always be represented as a directed graph on the set  $A \cup B$ .
- A useful way to represent relations is to use a bipartite graph which consists of two sets,  $A$  and  $B$ , of disjoint nodes such all arrows are shown going from  $A$  to  $B$ .
- If  $A$  and  $B$  have elements in common, they will have to be represented once in  $A$  and once in  $B$ .
- The next slide gives an example of a bipartite graph.

67

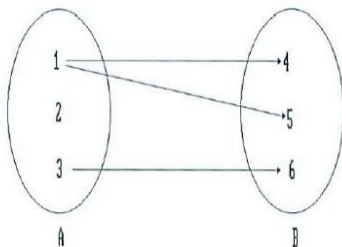
## The Reflexive Property



70

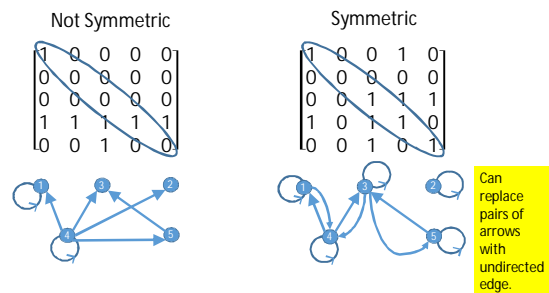
## Example

- Let  $A = \{1,2,3\}$ ,  $B = \{4,5,6\}$  and  $R = \{(1,4), (1,5), (3,6)\}$ .



68

## The Symmetric Property



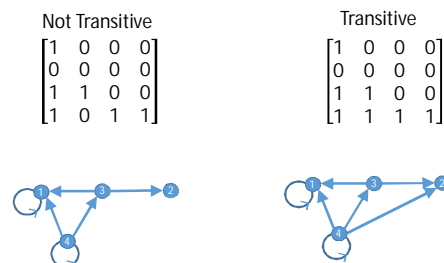
71

## Properties of Relations

- Definition:** Let  $A$  be a set and  $R$  a binary relation on  $A$ .
- $R$  is called **reflexive** if  $\forall x(xRx)$  is true.
- $R$  is called **symmetric** if  $\forall x\forall y(xRy \rightarrow yRx)$  is true.
- $R$  is called **antisymmetric** if  $\forall x\forall y(xRy \& yRx \rightarrow (x=y))$ .
- $R$  is called **transitive** if  $\forall x\forall y\forall z(xRy \& yRz \rightarrow xRz)$ .
- $A$  is the universe for all quantifiers

69

## The Transitive Property



72

## The Composition of Relations

- Often want to create new relations from given relations
- The following example will motivate our definition

73

$$\begin{array}{c}
 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
 \hline
 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}
 \end{array}
 \begin{array}{c}
 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \\
 P
 \end{array}
 \begin{array}{c}
 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \\
 Q
 \end{array}$$

76

## Example

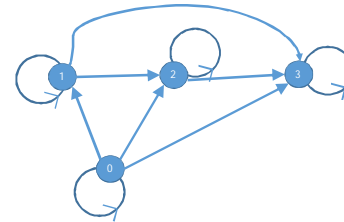
Name	Office
Doe, Jake	A-45
Doe, Jane	A-68
Doe, Jim	B-40
Doe, John	C-33

Phone	Office
3245	A-45
4786	A-68
7621	B-40
5346	C-33

Name	Phone
Doe, Jake	3245
Doe, Jane	4786
Doe, Jim	7621
Doe, John	5346

74

## Digraph Representation of $\leq$



77

## Composition of Relations

- **Definition:** a) Let  $A$ ,  $B$  and  $C$  be sets,  $R_1$  a relation between  $A$  and  $B$ , and  $R_2$  a relation between  $B$  and  $C$ . Let  $R$  be a relation between  $A$  and  $C$  defined by
  - $aRc$  iff  $\exists b \in B$  such that  $aR_1b$  and  $bR_2c$
  - $R$  is called the **composition** of  $R_1$  and  $R_2$  and is written  $R_1 \circ R_2$  or even sometimes just  $R_1 R_2$ .
- b) Let  $M$ ,  $P$  and  $Q$  be 0,1-matrices on  $A$ .  $Q$  is said to be the product of  $M$  and  $P$ , written  $Q = M \times P$  if
  - $\forall x \forall y ((\exists z ((M(x,z)=1) \& (P(z,y)=1))) \Rightarrow (Q(x,y)=1))$  is true. In other words,  $Q(a,c) = 1$  if and only if there is a  $b$  such that  $M(a,b) = 1$  and  $P(b,c) = 1$ .

75

## Matrix Representation of $\leq$

$$\begin{array}{c}
 0 \quad 1 \quad 2 \quad 3 \\
 0 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 1 \\
 2 \\
 3
 \end{array}$$

78

### Two Digraphs Representing <



79

```
Rel3 = [ ]
for i in Dom:
    for j in Dom:
        if (i-j)**2 < 5:
            Rel3.append((i,j))
```

```
print "Rel1 =",Rel1
print 60*"***"
print "Rel2 =",Rel2
print 60*"***"
print "Rel3 =",Rel3
```

82

### Matrix Representation of <

	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

80

```
Rel1 = [(0, 1), (0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
*****
Rel2 = [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 1), (1, 2), (1, 3), (1, 4), (2, 2),
(2, 3), (2, 4), (3, 3), (3, 4), (4, 4)]
*****
Rel3 = [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2),
(2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (3, 4), (4, 2), (4, 3), (4, 4)]
```

83

""" These programs illustrate how to check the properties of relations. They assume that relations are represented as pairs of elements on some domain."""

```
Dom = range(5)

Rel1 = [ ]
for i in Dom:
    for j in Dom:
        if i < j:
            Rel1.append((i,j))
```

```
Rel2 = [ ]
for i in Dom:
    for j in Dom:
        if i <= j:
            Rel2.append((i,j))
```

81

```
def Reflexive(D,R):
    for x in D:
        if (x,x) not in R:
            return False
    return True
```

Note two parameters, D and R!

```
print 60*"***"
print "Reflexive(Dom,Rel1)=",Reflexive(Dom,Rel1)
print 60*"***"
print "Reflexive(Dom,Rel2)=",Reflexive(Dom,Rel2)
print 60*"***"
print "Reflexive(Dom,Rel3)=",Reflexive(Dom,Rel3)
```

```
Reflexive(Dom,Rel1) = False
*****
Reflexive(Dom,Rel2) = True
*****
Reflexive(Dom,Rel3) = True
```

84

```
def Symmetric(R):
    for (a,b) in R:
        if (b,a) not in R:
            return False
    return True

print 60****
print "Symmetric(Rel1) =",Symmetric(Rel1)
print 60****
print "Symmetric(Rel2) =",Symmetric(Rel2)
print 60****
print "Symmetric(Rel3) =",Symmetric(Rel3)

Symmetric(Rel1) = False
*****
Symmetric(Rel2) = False
*****
Symmetric(Rel3) = True
```

85

## Partial Orders

- **Definition:** A reflexive, antisymmetric and transitive relation is called a **partial order**. A set with a partial order is called a **partially ordered set** or **poset**.

88

```
def AntiSymmetric(R):
    for (a,b) in R:
        if (a != b) and ((b,a) in R):
            return False
    return True

print "AntiSymmetric(Rel1) =",AntiSymmetric(Rel1)
print 60****
print "AntiSymmetric(Rel2) =",AntiSymmetric(Rel2)
print 60****
print "AntiSymmetric(Rel3) =",AntiSymmetric(Rel3)

AntiSymmetric(Rel1) = True
*****
AntiSymmetric(Rel2) = True
*****
AntiSymmetric(Rel3) = False
```

86

```
def PartialOrder(D,R):
    return Reflexive(D,R) and \
           AntiSymmetric(R) and \
           Transitive(R)

print 20****
print "PartialOrder(D,Rel1) =",PartialOrder(Dom,Rel1)
print 20****
print "PartialOrder(D,Rel2) =",PartialOrder(Dom,Rel2)
print 20****
print "PartialOrder(D,Rel3) =",PartialOrder(Dom,Rel3)

PartialOrder(D,Rel1) = False
*****
PartialOrder(D,Rel2) = True
*****
PartialOrder(D,Rel3) = False
```

89

```
def Transitive(R):
    for (a,b) in R:
        for (c,d) in R:
            if (b == c) and ((a,d) not in R):
                return False
    return True

print "Transitive(Rel1) =",Transitive(Rel1)
print 60****
print "Transitive(Rel2) =",Transitive(Rel2)
print 60****
print "Transitive(Rel3) =",Transitive(Rel3)

Transitive(Rel1) = True
*****
Transitive(Rel2) = True
*****
Transitive(Rel3) = False
```

87

## Total Orders

- **Definition:** A binary relation on a set A, R, is called a total order if it is a partial order such that  $\forall a \forall b (aRb \mid bRa)$

90

### Some Remarks

- Partial orders are very common and very useful.
- Besides the partial orders discussed above, another useful partial order is the lexicographical order on words also known as alphabetical order.

91

### There Are Infinite Total Orders that Are Not Well Orders

- Look at  $\{0, -1, -2, -3, \dots\}$  which is a total order, but is not a well order because it does not have a smallest element

94

### Well Orders

- **Definition:** Let  $S$  be a set and  $\leq$  a partial order on  $S$ . An element  $x \in S$  is called the smallest element or least element if  $x \leq y$  for all  $y \in S$ .
- A total order on a set  $S$  is called a **well-order** if every nonempty subset of  $S$  contains a smallest element.
- A set together with a well order defined on it is called a well-ordered set. The well order is said to **well-order** the set or to be a **well-ordering** of the set.

92

### The Natural Numbers are Well Ordered

- This is equivalent to the Principle of Induction
- It can be shown from Peano's Postulates
- We can assume it as another Postulate

95

### All Finite Total Orders Are Well Orders

- Just keep comparing elements until you find the smallest element
- This is the process of sorting
- Many different sorting algorithms

93

### Important Theorem

- Let  $S_1$  and  $S_2$  be well ordered sets with the well orders  $R_1$  and  $R_2$  respectively. Let  $S = S_1 \times S_2$ , and define a relation  $R$  on  $S$  so that
- $(a,b)R(c,d)$  iff  $((a \neq c) \ \& \ aR_1c) \text{ or } ((a=c) \ \& \ bR_2d)$
- Then  $R$  well orders  $S$ .
- I leave the proof as an exercise or we can do it in class if people want to see it.

96



### Example

- Let  $A$  be any well-ordered alphabet, and for each natural number,  $N$ , let  $\text{Words}_{A,N}$  be
- the set of all words of length  $N$ . Then for each  $N$ ,  $\text{Words}_{A,N}$  is well ordered using the standard alphabetical order.
- This is just a generalization of the preceding theorem that follows from it by using induction on  $N$ .

97

### Equivalence Relations

- **Definition:** A relation  $R$  on a set  $A$  is called an **equivalence relation** if it is reflexive, symmetric and transitive.
- Equivalence relations are very useful in computer science and mathematics, because they allow many complicated structures to be decomposed into substructures.

100

### The Space of Words

- Consider, the union  $\bigcup_{N=0,\dots} \text{Words}_{A,N}$ , which we will simply call  $\text{Words}_A$ .
- It seems reasonable to assume that  $\text{Words}_A$  is well ordered by the standard alphabetical order, but this turns out to be false.
- To see this, let  $A$  be the two letter alphabet consisting only of the letters  $a$  and  $b$ , and well-ordered so that  $a < b$ .

98

### Equivalence Classes

- Equivalence relations divide sets into equivalence classes
- We will handle this informally with an example, but we will use this feature later

101

### Funny Sequence

- $bba > babba > bababba > babababba \dots$
- Just keep adding  $ba$  to the beginning and you get a word that comes earlier in alphabetical ordering!

99

```
def EquivalenceRel(D,R):
    return Reflexive(D,R) and \
           Symmetric(R) and \
           Transitive(R)

Rel4 = []
for i in Dom:
    for j in Dom:
        if ((i-j)%2) == 0:
            Rel4.append((i,j))

print 20*"***"
print "EquivalenceRel(D,Rel1) =",EquivalenceRel(Dom,Rel1)
print 20*"***"
print "EquivalenceRel(D,Rel2) =",EquivalenceRel(Dom,Rel2)
print 20*"***"
print "EquivalenceRel(D,Rel3) =",EquivalenceRel(Dom,Rel3)

EquivalenceRel(D,Rel1) = False
*****
EquivalenceRel(D,Rel2) = False
*****
EquivalenceRel(D,Rel3) = False
EquivalenceRel(D,Rel4) = True
```

102

### Example

- Let  $R$  be defined on  $\mathbb{N} \times \mathbb{N}$  by  $R(a,b)$  iff  $(a-b)\%3 = 0$
- $R$  is an equivalence relation. Why?
- Reflexive:  $(a-a)\%3 = 0$
- Symmetric:  $(a-b)\%3 = 0$  iff  $(b-a)\%3 = 0$
- Transitive: if  $(a-b)\%3 = 0$  and  $(b-c)\%3 = 0$ , then  $(a-c)\%3 = (a-b)\%3 + (b-c)\%3 = 0$

103

### Functions

- **Definition:** A **function** or **mapping**  $f$  between two sets  $A$  and  $B$ , written  $f: A \rightarrow B$  is a relation that to each element of  $A$  associates one element of  $B$ .  $A$  is called the **domain** and  $B$  is called the **codomain**.
- Functions are special cases of relations.
- The big difference is that
- Every element in  $A$  is assigned exactly one element in  $B$

106

### Example

- Divides  $\mathbb{N}$  into 3 classes:
  - $\text{Mod}_0 = \{0, 3, 6, 9, \dots\}$
  - $\text{Mod}_1 = \{1, 4, 7, 10, \dots\}$
  - $\text{Mod}_2 = \{2, 5, 8, 11, \dots\}$
- These classes behave like numbers since  $(a+b)\%m = a\%m + b\%m$  and
- $(ab)\%m = (a\%m)(b\%m)$

104

### Functions

- When we write  $f: A \rightarrow B$  we call  $A$  the **domain** of  $f$  and  $B$  the **codomain** of  $f$
- When we write  $f(a) = b$  we call  $a$  the argument and  $b$  the value
- This is a bit cumbersome, so we write  $f(a) = b$  instead of  $(a,b) \in f$

107

### Functions

- Functions can be thought of as special types of relations
- We write  $f: A \rightarrow B$  to mean that  $f \subseteq A \times B$  to mean that  $\forall a \in A \exists! b \in B$  such that  $(a,b) \in f$ 
  - $\exists!$  means there exists a **unique** element
- This is a bit cumbersome, so we write  $f(a) = b$  instead of  $(a,b) \in f$

105

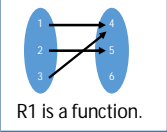
### Functions

- If  $f: A \rightarrow B$  and  $g: C \rightarrow D$ , then to say that  $f = g$  means
  - $A = C$
  - $B = D$
  - $f(a) = g(a) \forall a \in A$
- We use  $\underline{n}$  to mean the set  $\{0, 1, \dots, n-1\}$
- Similar to `list(range(n))` in Python 3.X or `range(n)` in Python 2.X

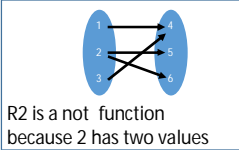
108

### Examples

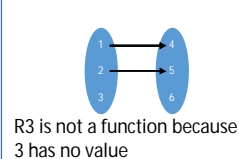
- $A = \{1, 2, 3\}$ ,  $B = \{4, 5, 6\}$
- $R1 = \{(1,4), (2,5), (3,4)\}$
- $R2 = \{(1,4), (2,5), (2,6), (3,4)\}$
- $R3 = \{(1,4), (2,5)\}$



R1 is a function.



R2 is not a function because 2 has two values



R3 is not a function because 3 has no value

109

### Total Functions

- To emphasize that some function is not a partial function, we use the term **total function**
- The terms total function and function are synonyms

112

### Partial Functions

- Definition:** A **partial function** is a binary relation between A and B that associates at most one element of B to every element of A
- Question:** Why bother with partial functions?
- Answer:** They are all around you!

110

### Functions

- A **finite sequence** in X is a function  $f: \underline{n} \rightarrow X$  for some integer n
- Note that  $\underline{-n} = \dots = \underline{-1} = \emptyset$
- An **infinite sequence** in X is a function  $f: \mathbb{N} \rightarrow X$ , where  $\mathbb{N}$  is the Natural Numbers
- We also use the notation  $f(a_1, \dots, a_n) = b$  when  $f: A_1 \times \dots \times A_n \rightarrow B$

113

### Examples

- What is  $\maxint + \maxint$ ?
- What is  $3/0$ ?
- We can only compute a finite # of values in a finite time

111

### Functions

- If  $f(a) = b$ , we say that b is the **image of a** and a is in the **inverse image of b**
- If  $X \subseteq A$ , we write  $f(X) = \{ b \in B \mid \exists a \in X \text{ s.t. } f(a) = b \}$ . We call  $f(X)$  the **image of X**
- If  $Y \subseteq B$ , we write  $f^{-1}(Y) = \{ a \in A \mid f(a) \in Y \}$ . We call  $f^{-1}(Y)$  the **inverse image of Y**
- $f(A)$  is called the **range of f**

114

### Functions

- Let  $f: A \rightarrow B$
- $f$  is **surjective** or **onto** iff  $f(A) = B$
- $f$  is **injective** or **1-1** iff  $a_1 \neq a_2 \rightarrow f(a_1) \neq f(a_2)$
- $f$  is **bijective** iff  $f$  is surjective and injective
- $f: A \rightarrow A$  is a bijection iff  $f$  is a **permutation of A**
- Bijective functions have inverse functions  $f^{-1}$  which are different from inverse images

115

### Examples

5.  $f: X \rightarrow X$  ( $X$  is any set) where  $f(x) = c$  where  $c$  is any point in  $X$ 
  1. Injective?
  2. Surjective?
  3. Bijective?
6.  $f$  is called a **constant function**

118

### Examples

1.  $f: \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(n) = n^2$ 
  1. Injective?
  2. Surjective?
  3. Bijective?
2.  $f: \mathbb{Z} \rightarrow \mathbb{Z}$  ( $\mathbb{Z} = [0, +1, -1, +2, -2, \dots]$ ) where  $f(n) = n^2$ 
  1. Injective?
  2. Surjective?
  3. Bijective?

116

### Function Composition

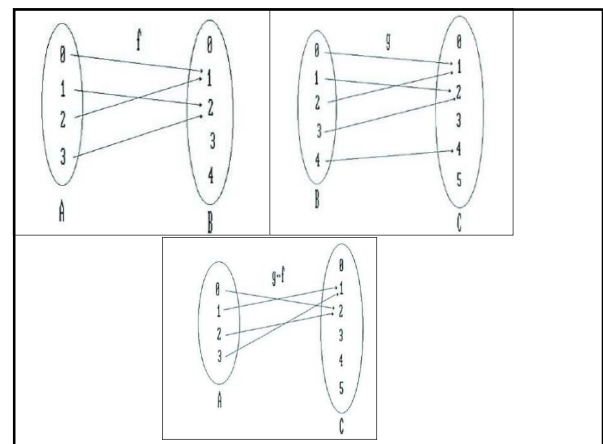
- **Definition:** Let  $f: A \rightarrow B$  and  $g: B \rightarrow C$  be functions. The **composition of  $f$  and  $g$** , written  $g \circ f: A \rightarrow C$ , is the function defined by  $g \circ f(a) = g(f(a))$ , i.e., first compute  $f(a)$  and then plug that value into  $g$  to get the final result.

119

### Examples

3.  $f: \mathbb{R} \rightarrow \mathbb{R}$  ( $\mathbb{R}$  is the real numbers) such that  $f(x) = x^2$ 
  1. Injective?
  2. Surjective?
  3. Bijective?
4.  $f: X \rightarrow X$  ( $X$  is any set) where  $f(x) = x$ 
  1. Injective?
  2. Surjective?
  3. Bijective?

117



120

### The Associative Law of Function Composition

- We want to show that  $h \circ (g \circ f) = (h \circ g) \circ f$  where  $f: A \rightarrow B$ ,  $g: B \rightarrow C$  and  $h: C \rightarrow D$
- We do so by direct computation.
- Pick  $a \in A$ . Then  $(h \circ (g \circ f))(a) = h(g \circ f(a)) = h(g(f(a)))$
- Note that  $((h \circ g) \circ f)(a) = (h \circ g)(f(a)) = h(g(f(a)))$  which is the same result.

121

### Strings

- A **string** over an alphabet  $A$ , is a sequence of elements taken from  $A$
- A **substring** of a string is formed by taking some number of consecutive elements from the string
- There are  $|A|^k$  strings of length  $k$ , where the length of a string is the number of terms in the sequence

124

### Theorem

- **Theorem:** Let  $f: A \rightarrow B$  and  $g: B \rightarrow C$  be functions. Then
  - a. If  $f$  and  $g$  are injections, then  $g \circ f$  is an injection.
  - b. If  $f$  and  $g$  are surjections, then  $g \circ f$  is a surjection.
  - c. If  $f$  and  $g$  are bijections, then  $g \circ f$  is a bijection.
- **Proof:** Exercise

122

### Permutations

- One definition of a **permutation on a set  $S$**  is a bijection from  $S$  into  $S$
- This definition is handy, because such permutations can be composed with each other to form a group

125

### Counting

- Rule of Sums
- $|A \cup B| = |A| + |B|$  if  $A$  and  $B$  are disjoint
- $|A \cup B| = |A| + |B| - |A \cap B|$  in general
- Rule of Products
- $|A \times B| = |A| \times |B|$

123

### An Alternative Approach to Permutations

- Define  $\underline{n} = \{0, 1, 2, \dots, n-1\}$
- Let  $S$  be a set, and define the  $k$ -permutations of  $S$  to be the set of all injections from  $\underline{k}$  into  $S$
- Note that if  $k > |S|$ , the number of such permutations is 0
- If  $|S| = n$ , the number of  $k$ -permutations is denoted by  $P(n, k)$

126

### $P(n,k)$

- To list all  $k$ -permutations of  $S$  proceed as follows
- Think of them as being given by  $f(1), f(2), \dots, f(k)$
- A value for  $f(1)$  can be chosen in  $n$  ways
- Once  $f(1)$  is chosen there are  $n-1$  choices for the value of  $f(2)$ , etc.
- Thus  $P(n,k) = n(n-1)(n-2)\dots(n-k+1)$
- Note that  $P(n,k) = n!/(n-k)!$

127

### Combinations

- What are the equivalence classes of  $R$ ?
- They correspond to the different  $k$ -combinations of  $\underline{n}$
- How many elements are there in each equivalence class?
- It is easy to see that there are  $P(k,k) = k!$  elements in each equivalence class

130

### Combinations

- A  **$k$ -combination of an  $n$ -element set** is just a subset of the  $n$ -element set that contains  $k$ -elements
- It is our goal to count these
- How should we do this?
- We start with the set of all  $k$ -permutations of an  $n$ -element set
- Define  $P(n,k)$  as the set  $\{ f: \underline{k} \rightarrow \underline{n} \mid f \text{ is an injection} \}$

128

### Combinations

- Since the equivalence classes are all the same size and they divide  $P(n,k)$  there are  $P(n,k)/k!$  equivalence classes, each of which corresponds to a unique  $k$ -element subset of  $\underline{n}$
- We call the number  $P(n,k)/k!$ ,  $C(n,k)$
- Note  $C(n,k) = n!/(k!(n-k)!)$
- The  $C(n,k)$  are also called **binomial coefficients**

131

### Combinations

- We know that  $|P(n,k)| = P(n,k)$
- Recall the concept of an image:
- If  $f: X \rightarrow Y$  and  $S \subseteq X$ , then  $f(S) = \{ y \in Y \mid \exists s \in S \text{ s.t. } f(s) = y \}$
- For  $f, g \in P(n,k)$  define  $f R g$  iff  $f(\underline{k}) = g(\underline{k})$
- It is easy to see that  $R$  is an equivalence relation
- Reflexive, Symmetric, Transitive?

129

### Binomial Coefficients

- These may be familiar to you from **Pascal's Triangle**

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
    
```

Note that  $C(n,0) = C(n,n) = 1$   
 $0! = 1$

132

### Binomial Coefficients

- $(a + b)^n = a^n + C(n,1)a^{n-1}b + C(n,2)a^{n-2}b^2 + \dots + C(n,n-1)ab^{n-1} + b^n$
- Note if we let  $a = 1 = b$ , we get
- $2^n = C(n,0) + C(n,1) + C(n,2) + \dots + C(n,n)$
- Can you give me a combinatorial proof of this?
- Note also that  $C(n,k) = C(n,n-k)$
- Can you give me a combinatorial proof of this?

133

### Binomial Coefficients

- If you want to refer to  $C(n,k)$  orally, you should say "n choose k"
- If  $H(x) = -x \lg x - (1-x) \lg(1-x)$  ( $0 < x < 1$ )
- and  $H(0) = H(1) = 0$
- Then  $C(n,k) \leq 2^{nH(k/n)}$
- Note that for **fixed k**,  $C(n,k) \in \Theta((n/k)^k) = \Theta(n^k)$
- Fascinating bounds for  $k = n^5$  and  $k = n/2$

136

### The Binomial Theorem

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

134

### Order Of Growth

- Describing the performance of an algorithm precisely is tricky
  - Memory hierarchies affect performance
  - Different machine architectures affect performance
  - Different inputs affect performance
- For this reason use an order of growth notation:  $O, \Omega, \Theta$

137

### Bounds for Binomial Coefficients

Another common notation for  $C(n, k)$  is

$$\binom{n}{k}$$

We have the following inequalities

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} \leq \frac{n^n}{k^k (n-k)^{(n-k)}}$$

135

### Big Oh

- What does Big Oh notation mean?
- Often written  $f(x) = O(g(x))$
- I prefer that you do not use this notation, because you are setting two very different things equal
- The LHS (what is this?) is a function while the RHS (what is this?) is a family of functions
- Better to use  $f(x) \in O(g(x))$

138

### Big Oh

- What does  $f(x) \in O(g(x))$  mean?
- It means that  $\exists$  (what does this mean?) an integer  $n_0 > 0$  and  $\exists c > 0$  s.t. (such that)  $\forall$  (what does this mean?)  $n \geq n_0$ ,  $|f(n)| \leq c|g(n)|$
- I use  $|f|$  and  $|g|$  to avoid problems with negative numbers
- Generally our functions are all  $\geq 0$  when we deal with running time

139

### How Do You Prove Something is not in Big Oh or Big Omega?

- Suppose you want to show that  $x^2$  not in  $\Omega(x^3)$
- These proofs always go by contradiction
- Assume that  $x^2 \in \Omega(x^3)$
- Then there exist  $n_0 > 0$  and  $c > 0$  such that  $\forall n \geq n_0, n^2 \geq c n^3$
- This implies that  $1/c \geq n$  for all  $n \geq n_0$
- Pick  $n > 1/c$  and you get a contradiction!

142

### How Do You Prove Things About Big Oh?

- Say you want to show  $(x^2 + 5x) \in O(x^3)$
- Pick  $c = 1$  and  $n_0 = 10$
- Note that if  $n \geq 10$  we have
- $n^2 + 5n \leq n^2 + 10n \forall n \geq 10$  Why?
- $n^2 + 10n \leq 2n^2 \forall n \geq 10$  Why?
- $2n^2 \leq n^3 \forall n \geq 10$  Why?
- $(x^2 + 5x) \in O(x^3)$  Why?

140

### What is Big Theta?

- $f(x) \in \Theta(g(x))$  iff  $f(x) \in O(g(x))$  and  $f(x) \in \Omega(g(x))$
- You prove that  $f(x)$  is in big theta by using two proofs
- First, you prove it is in big oh
- Second, you prove it is in big omega
- If it is not in both of them, it is not in big theta

143

### Big Omega ( $\Omega$ )

- What does  $f(x) \in \Omega(g(x))$  mean?
- Similar to big oh, but means we have a lower bound
- It means that  $\exists$  an integer  $n_0 > 0$  and  $\exists c > 0$  s.t.  $\forall n \geq n_0, |f(n)| \geq c|g(n)|$
- Proving that something is in big omega is similar to proving it is in big oh, except that you have  $\geq$  instead of  $\leq$

141

### Progressions

- Certain types of series are called progressions
- In particular, when we state that we have a progression, we imply that there is some simple rule between consecutive members of the associated sequence
- There are two types of progressions that are widely used: arithmetic and geometric

144



### Arithmetic Progressions

- Definition: An arithmetic progression is a series where each term in the associated sequence differs from the preceding term by a constant
- For example,  $2 + 4 + 6 + \dots$  or  $7 + 10 + 13 + \dots$
- In general,  $S_n = \sum_{k=0}^n (F + k * d) = F + (F+d) + \dots + (F+n*d)$  where F means first term
- How do we sum this?

145

### Geometric Progressions

- Definition: An geometric progression is a series where each term in the associated sequence is a constant multiple of the preceding term
- For example,  $2 + 4 + 8 + \dots$  or  $1 + 1/3 + 1/9 + \dots$
- In general,  $S_n = \sum_{k=0}^n (F * r^k) = F + F*r + \dots + F*r^n$  where F means first term
- How do we sum this?

148

### Arithmetic Progressions

- Let  $L = F + n*d$ , i.e., L stands for the last term
- Then we have  $S_n = \sum_{k=0}^n (F + k * d) = F + (F+d) + \dots + L$  and also  $S_n = L + (L - d) + (L - 2d) + \dots + F$
- Adding both expressions for  $S_n$  together yields  $2S_n = (n+1)(F+L)$
- Thus,  $S_n = (n+1)(F+L)/2$

146

### Summing Geometric Progressions

- We use the following suggestive method  
 $S_n = F + Fr + Fr^2 + \dots + Fr^n$   
 $rS_n = Fr + Fr^2 + \dots + Fr^n + Fr^{n+1}$   
 so  
 $S_n - rS_n = F - Fr^{n+1} = F - rL$  where  $L = Fr^n$  is the last term  
 Hence,  $S_n = (F - rL)/(1-r) = (rL - F)/(r-1)$

149

### Arithmetic Progressions

- One way to summarize the previous formula is to say that the sum of an arithmetic progression is *the product of the number of terms by the average value of the first and last terms*
- $S_n = (\text{no. of terms}) * (\text{First} + \text{Last})/2$

147

### Summing Geometric Progressions

Hence,  $S_n = (F - rL)/(1-r) = (rL - F)/(r-1)$   
 We use the first version when  $r < 1$  and the second when  $r > 1$   
 What about when  $r = 1$ ?  
 If we want infinite sums, then  $r < 1$  or the series diverges since the terms go to  $\infty$   
 Note that if  $|r| < 1$ ,  $\lim_{n \rightarrow \infty} Fr^n = 0$ , so  
 $S_\infty = F/(1-r)$

150

### Summing Infinite Geometric Series

- To summarize, if  $|r| < 1$ ,  $S_\infty = F/(1-r)$
- In particular, a popular  $r$  is  $1/2$ , so we have that  $F + F/2 + F/4 + \dots = F/(1-1/2) = 2F$

151

### Tools for Dealing With Series

- I will show you some tools and tricks for dealing with series
- Some are straightforward and others pretty tricky
- Many useful techniques developed by mathematicians and come from calculus and analysis

154

### The Harmonic Series

- A fascinating sequence is  $a_k = 1/k$  which is called the Harmonic sequence
- The Harmonic series is the series associated to this sequence
- In particular, the terms  $H_n = \sum_{k=1}^n \frac{1}{k}$  are called the Harmonic numbers
- Note that  $H_0$  makes no sense so we start at 1

152

### Linear Combinations of Series

- $\sum_{k=0}^n (c * a_k + b_k) = c * (\sum_{k=0}^n a_k) + \sum_{k=0}^n b_k$
- Example:  

$$- \sum_{k=0}^n (2k - 1) = 2 * \sum_{k=0}^n k + \sum_{k=0}^n (-1) =$$

$$- = 2 * n(n+1)/2 + (n+1)(-1) = n(n+1) - (n+1) =$$

$$- = (n-1)(n+1)$$

155

### The Harmonic Series

- We will soon show you that  $H_n = \ln n + O(1)$
- This is a non-trivial result and requires the development of some tools

153

### Telescoping Series

- Suppose you have two sequences  $a_i$  ( $i = 0, 1, \dots$ ) and  $b_j$  ( $j = 1, 2, \dots$ ) related as follows
- $b_i = a_i - a_{i-1}$  for  $i = 1, 2, \dots$
- Suppose that we want to sum the  $b_i$
- Note that  $b_1 + b_2 + \dots + b_n =$   

$$= (a_1 - a_0) + (a_2 - a_1) + (a_3 - a_2) + \dots + (a_n - a_{n-1})$$
- $= a_n - a_0$  for all  $n$

156

### Telescoping Series

- **Example:**
- Suppose you want to sum  $1/(k*(k+1))$  for  $k = 1$  to  $n$
- This would be the sum  $1/2 + 1/6 + 1/12 + \dots + 1/(n*(n+1))$
- First note that  $1/(k*(k+1)) = 1/k - 1/(k+1)$ 
  - Let  $a_i = -1/(i+1)$  so  $a_i - a_{i-1} = -1/(i+1) + 1/i$
  - $= 1/(i*(i+1)) = b_i$

157

### Some Sample Programs

```
s = 0
for i in range(1,100):
    s += 1.0/2**i
print s

>>> 1.0
0.99 (99 terms)
>>> === RESTART ===
>>> 1.0
0.995 (199 terms)

>>> === RESTART ===
>>> 1.0
0.999 (999 terms)

>>> === RESTART ===
>>> 1.0
0.9999 (9,999 terms)
>>>

s = 0
for i in range(1,10000):
    s += 1.0/(i*(i+1))
print s
```

160

### Telescoping Series

- From our previous results we see that
$$\sum_{k=1}^n b_k = a_n - a_0 = \frac{-1}{n+1} - \frac{-1}{1}$$

$$= 1 - \frac{1}{n+1}$$

Thus,  $\sum_{k=1}^{\infty} b_k = 1$

Note that this is very slow convergence

158

### Using Integrals

- **Definition:** A function is called monotonically increasing iff  $x \leq y \rightarrow f(x) \leq f(y)$
- **Definition:** A function is called monotonically decreasing iff  $x \leq y \rightarrow f(x) \geq f(y)$

161

### Telescoping Series

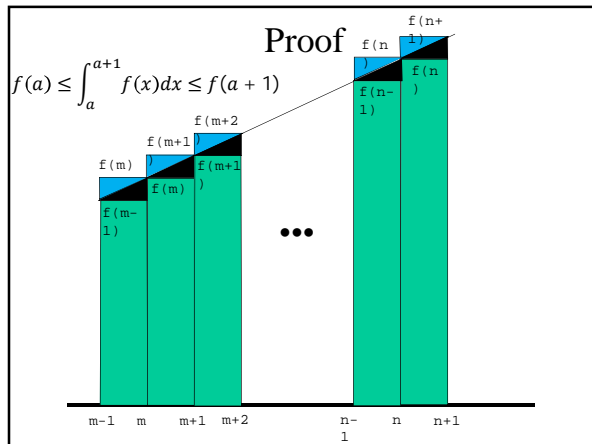
- Another view of this process is
- $S_n = (1-1/2) + (1/2-1/3) + \dots + (1/n-1/(n+1)) =$
- $= 1 - 1/(n+1)$

159

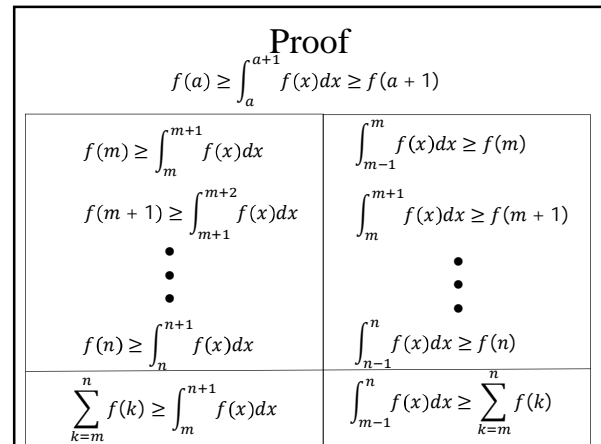
### Using Integrals

- **Theorem (Integral Bound 1):** If  $f$  is monotonically increasing then
$$\int_{m-1}^n f(x) dx \leq \sum_{k=m}^n f(k) \leq \int_m^{n+1} f(x) dx$$

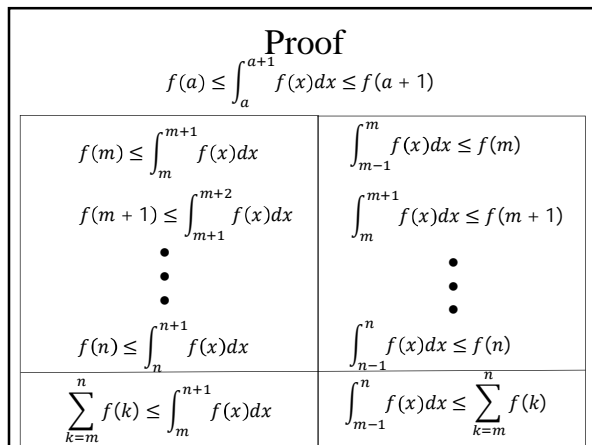
162



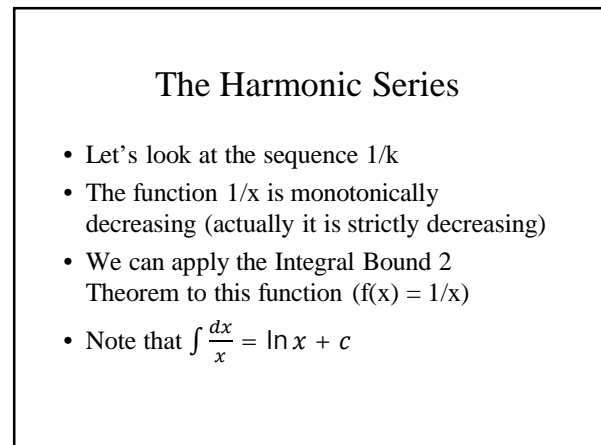
163



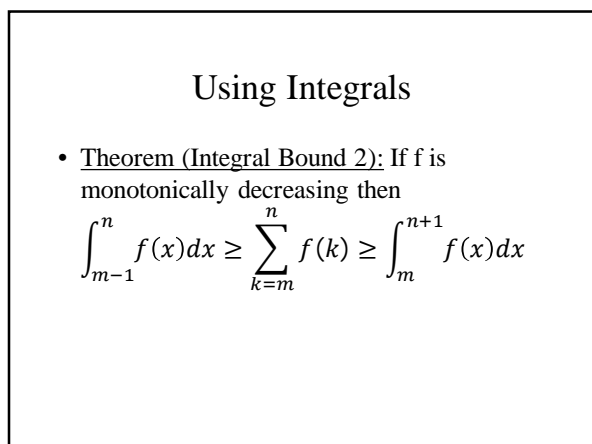
166



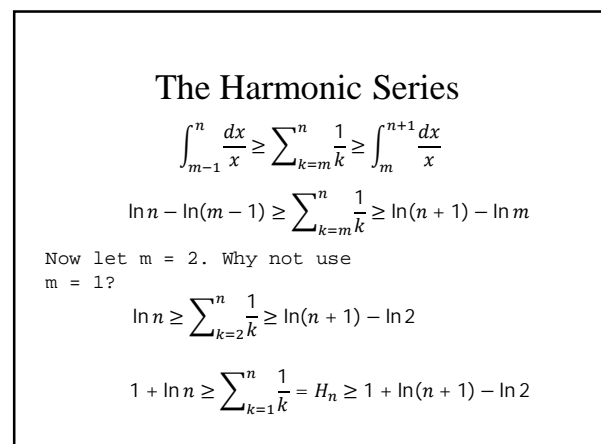
164



167



165



168

### The Harmonic Series

$$1 + \ln n \geq \sum_{k=1}^n \frac{1}{k} = H_n \geq 1 + \ln(n+1) - \ln 2$$

Now let's simplify the right hand side.

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

$$1 + \ln n + \ln \frac{n+1}{n} - \ln 2 \geq 1 + \ln n - \ln 2 > \ln n + .3$$

$$\ln n + 1 \geq \sum_{k=1}^n \frac{1}{k} = H_n \geq \ln n + .3$$

This is a pretty good bound!

169

### More Advanced Techniques

- $xS' = \sum_{k=0}^{\infty} k * x^k = x * \frac{d(1-x)^{-1}}{dx}$
- Now  $\frac{d(1-x)^{-1}}{dx} = (-1) * (1-x)^{-2} * \frac{d(1-x)}{dx}$
- $= (1-x)^{-2}$
- Thus our final result is that
- $\sum_{k=0}^{\infty} k * r^k = \frac{r}{(1-r)^2}$

172

### More Advanced Techniques

- We know that for  $|r| < 1$ ,  $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$
- The question is how to compute  $\sum_{k=0}^{\infty} k * r^k$  if this sum converges
- We start with the Taylor series identity
- $S = \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} = (1-x)^{-1}$

170

### Some Famous Results

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \approx 1.64493$$

$$\sum_{n=1}^{\infty} \frac{1}{n^4} = \frac{\pi^4}{90} \approx 1.08232$$

$$\int_0^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2} \approx 0.886227$$

These results require a variety of other techniques

173

### More Advanced Techniques

- $S = \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} = (1-x)^{-1}$
- As long as  $|x| < 1$ , we can rigorously justify the following step: differentiate one side term by term and differentiate the other side
- This gives the formula
- $S' = \sum_{k=0}^{\infty} k * x^{k-1} = \frac{d(1-x)^{-1}}{dx}$

171

### Products

- Products occasionally play a role in algorithmic analysis, but they do so less frequently than sums
- One reason for this is that most products grow or shrink very rapidly
- Another reason is that logarithms of products turn out to be sums

174

### Products

- $\prod_{k=m}^m a_k = a_m \times a_{m+1} \dots \times a_n$
- Note that  $\log \prod_{k=m}^n a_k = \sum_{k=m}^n \log a_k$
- The most common product is
- $n! = \prod_{k=1}^n k$
- We will have more to say about  $n!$  in the future
- Note that  $\log n! = \sum_{k=1}^n \log k$
- Since  $\log k$  is monotone increasing we can use the earlier theorems to approximate  $\log n!$

175

### Example

- If we are given the following equation
  - $T(n) = \Theta(1)$  if  $n = 1$
  - $T(n) = 2T(n/2) + \Theta(n)$  if  $n > 1$
- We get the solution  $T(n) = \Theta(n \log n)$
- There are 4 techniques for solving these equations
- The textbook only discusses 3 of these techniques

178

### A Final Note on Products

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

176

### Techniques For Solving Recurrences

1. Substitution Method
  2. Iteration Method
  3. Master Theorem Method
  4. Exact Solution
- We will only talk about method 3 briefly in this lecture.

179

### Recurrences

- A very useful tool
- Also called recurrence relations or recurrence equations
- These are basically just recursive functions
- They are of great interest in analyzing **Divide and Conquer Algorithms**

177

### Master Theorem

- Theorem: Let  $a \geq 1$ ,  $b > 1$  be constants and  $T(n) = a T(n/b) + f(n)$  where  $\text{floor}(n/b)$  or  $\text{ceiling}(n/b)$  can be used instead of  $n/b$
- Then let  $\alpha = \log_b a$
- If  $f(n) \in O(n^{a-\epsilon})$  for some  $\epsilon > 0$ , then  $T(n) \in \Theta(n^\alpha)$
- If  $f(n) \in \Theta(n^\alpha)$  then  $T(n) \in \Theta(n^\alpha \log n)$

180

### Master Theorem (cont)

- If  $f(n) \in \Omega(n^{a+\epsilon})$  for some  $\epsilon > 0$ , and if  $a \cdot f(n/b) \leq c \cdot f(n)$  for some  $c < 1$  and  $n$  large enough, then  $T(n) \in \Theta(f(n))$

181

### Example 2

- $T(n) = T(2n/3) + 1$
- $a = 1, b = 3/2$  (why?)
- $\alpha = \log_{3/2} 1 = 0$
- $n^\alpha = n^0 = 1$
- $f(n) = 1$
- $f(n) \in \Theta(n^\alpha)$ , so by the Master Method
- $T(n) \in \Theta(n^0 \log n) = \Theta(\log n)$

184

### Observations about the Master Theorem of Recurrences

- It does not cover all cases
- Won't handle  $T(n) = n^{0.5} * T(n^{0.5}) + 1$
- Won't handle  $T(n) = T(n/3) + T(2n/3) + \dots$
- Very useful for many divide and conquer algorithms
- Does not give an exact solution
- Here are some examples of the Master Method in action

182

### Example 3

- $T(n) = 3T(n/4) + n \log n$
- $a = 3, b = 4$
- $\alpha = \log_4 3 \approx .8$
- $n^\alpha < n$  for all  $n > 1$
- $f(n) = n \log n \in \Omega(n^{a+\epsilon})$  where  $\epsilon = .1 > 0$
- We need to use Part 3 of the Master Method, so we need to check whether  $a f(n/b) \leq c f(n)$  for some  $0 < c < 1$  and  $n$  sufficiently large

185

### Example 1

- $T(n) = 9T(n/3) + n$
- $a = 9, b = 3$
- $\alpha = \log_3 9 = 2$
- $n^\alpha = n^2$
- $f(n) = n$
- $f(n) \in O(n^{1.9}) = O(n^{2-\epsilon})$  with  $\epsilon = .1 > 0$
- By the Master Method
- $T(n) \in \Theta(n^2)$

183

### Example 3 (cont)

- a  $f(n/b)$  here is  $3 (n/4)(\log n/4) = (3n/4)\log(3n/4) \leq (3n/4)\log n = (3/4) f(n)$  so we can set  $c = 3/4$
- By the Master Method
- $T(n) \in \Theta(f(n)) = \Theta(n \log n)$

186