**Course Basics**

- Grading
  - Homework 15% (5 Homeworks total)
  - Midterm 35%
  - Final 50%
- Grades will be available on Gradesource
- Course will be curved so don’t think 89.4% = B+

---

**Course Overview**

**Course Basics**

- Website: [http://cseweb.ucsd.edu/classes/su11/cse21-a/](http://cseweb.ucsd.edu/classes/su11/cse21-a/)
- My Office: EBU3B - 2106
- TA: Wensong “Tony” Xu
- Tutors: Ivan Tham and Susan Liu
- Office Hours: TBD. These will be posted on the webpage by the end of the day
- Section: Wensong is deciding on the final schedule for the discussion

---

**Discussion Board**

![Moodle screenshot]

- 26 July - 1 August
- 2 August - 8 August
- 9 August - 15 August
Course Contents

- This class is all about how to count various structures
- Rough Outline of Where we are Going
  - Counting elements in a set
  - Counting elements in a list
  - Counting functions
  - Using Counting to compute probabilities
  - Using Decision Trees to Count
  - Counting Recursively defined quantities

Why is this class useful?

- This class provides tools that are useful in many areas of computer science

Algorithmic Complexity

![Graph showing time to solve (seconds) vs input size for Algorithm 1 and Algorithm 2](image1)

Algorithmic Complexity

![Graph showing time to solve (seconds) vs input size for Algorithm 1 and Algorithm 2](image2)

Algorithm 1 = e^(c*input)
Algorithm 2 = b*input^2
Probability and Statistics

- Probability and Statistics
  - What are the odds of something happening?
  - Making intelligent decisions

Advice for Doing Well on the Homework

- Check your answers by solving problems multiple ways
- Work with a group (okay as long as you write up your own solutions)
- It really helps to know how to do some mathematical programming (but is not required)
  - Python (http://learnpythonthehardway.org/book/)
  - Octave
  - Matlab

Some Training in Mathematical Proofs

- We will learn some basics of induction
- These techniques will prove very helpful in Algorithms where the goal will be to prove that the algorithm computes what we claim that it does

Lectures

- Notes will be provided every day at the beginning of lecture
- If you miss lecture you can pick the notes up outside my office (they will be in a cardboard box)
- There are several example problems per lecture that I will ask you to solve on your own.
CSE 21
Math for Algorithms and Systems Analysis
Lecture 1
Basic Set Theory and Lists With Repetition

Testing for Equality for Sets and Lists

- Two sets are equal if and only if they have the same members.
- Two lists are equal if and only if they have the same members in the same order.

Definitions

- Set – A collection of unordered items
e.g. \{apple, banana, sandwich\}

- List – A collection of ordered items
e.g. (banana, apple, sandwich)

- The type of braces \{\} or () distinguish which type of collection (set or list) we are referring to.

Two Notations for Defining Sets

- S contains all elements that satisfy “condition” (we’ll call this conditional form)
  \[ S = \{ x \mid \text{condition}(x) \} \]

- S contains the elements 0, 1, and 2 (we’ll call this enumerated form)
  \[ S = \{0, 1, 2\} \]
  \[ S = \{0 \ldots 9\} \]
The Empty Set and Cardinality

• Null set is the set with no elements written as \( \emptyset \)
• The Cardinality of a set \( S \) is written as \( |S| \) and denotes the number of elements in the set \( S \)

Operations on Sets

• Union
  \[ S_1 \cup S_2 = \{ x \mid x \in S_1 \text{ or } x \in S_2 \} \]

• Intersection
  \[ S_1 \cap S_2 = \{ x \mid x \in S_1 \text{ and } x \in S_2 \} \]

• Difference
  \[ S_1 - S_2 = \{ x \mid x \in S_1 \text{ and } x \notin S_2 \} \]

Example Problem – Google Plus

• Let F be the set of all of your close friends
• Let C be the set of all of your coworkers
• Let M be the set of all managers at your work
• Using the binary set operators or sets in enumerated form construct e-mail lists for the following events:
  – A surprise party for your close friend Jeremy (where the guest list includes all of your close friends)
  – An after work get together with all of your coworkers that you are friendly with
  – An e-mail to all your coworkers that are not managers

K-Lists

• K-lists are lists with exactly \( k \) (not necessarily distinct) elements
• We will use the notation \( |L| \) to denote the number of elements in the list \( L \).
• \( L_i \) gives the \( i \)th element of the list \( L \)
• Theorem 1:
  – There are exactly \( n^k \) k-lists that can be made from the elements of an n-set
Example Problems

• How many 10-digit numbers can be made from the digits 1 through 5?

• How many 3 letter words can be made from the letters “a” through “z”?

Rule of Product

• Suppose we want to know the number of lists we can make consider we have $c_1$ choices for the first element, $c_2$ for the second, ... $c_n$ for the nth element

• The number of lists that can be made in this way is

$$c_1 \times c_2 \times \ldots c_n$$

• Note: that the number of choices available at choice i cannot depend on previously selected choices

Example Problems

• A California license plate is made of 1 digit 0-9 followed by three letters A-Z and then 3 digits 0-9. How distinct California license plates are there?

Example Problems

• How many non-negative integers are there less than 1,000 but greater than or equal to 100?
**Cartesian Product**

- Rule of Product counts the number of lists, Cartesian product gives the set of each of these lists.
- Definition:
  Given sets, $S_1, S_2, ... S_n$ the Cartesian product of these sets is written as:

$$S_1 \times S_2 \times ... \times S_n = \{x \mid x \text{ is an n-list with the ith element } x_i \in S_i\}$$

**Example**

- $S_1 = \{a,b,c\}$
- $S_2 = \{d,e,f\}$
- $S_1 \times S_2 = \{(a,d), (a,e), (a,f), (b,d), (b,e), (b,f), (c,d), (c,e), (c,f)\}$
- $S_1 \times S_2 \times S_1 = \{(a,d,a),(a,d,b),(a,d,c),(a,e,a), (a,e,b),(a,e,c),(a,f,a),(a,f,b),(a,f,c),(b,d,a), (b,d,b),(b,d,c),(b,e,a),(b,e,b),(b,e,c),(b,f,a), (b,f,b),(b,f,c),(c,d,a),(c,d,b),(c,d,c),(c,e,a), (c,e,b),(c,e,c),(c,f,a),(c,f,b),(c,f,c)\}$

**Cartesian Product vs. Rule of Product**

- Rule of Product gives the number of lists, Cartesian product constructs the set of each of these lists.

$$|S_1 \times S_2 \times ... S_n| = |S_1| \times |S_2| \times ... \times |S_n|$$

**Cartesian Product Example Problems**

- Construct the set of all 10 digit numbers (leading 0’s are allowed)
A Different View of the Rule of Product

- We will typically use the rule of product to count the number of a particular type of structure
- We can think of each each choice c₁ to cₙ that we make specifying some aspect of the structure
- In order to use the product rule to count the number of structures we require that:
  - Each structure can only arise from 1 sequence of decisions
  - For each structure there is at least 1 sequence of decisions that gives rise to it

Example of Properly Applied Rule of Product

- Suppose the we are trying to count the number of full houses that can be made from the standard 52 card deck
- We can think of trying to count this using the product rule
  - Choice 1: choose two ranks (2 – Ace) one for the 3 of the kind one for the two of a kind
  - Choice 2: Suits of the three of a kind
  - Choice 3: Suits of the pair

Example of Improperly Applied Rule of Product

- Suppose the we are trying to count the number of two pairs that can be made from the standard 52 card deck
- We can think of trying to count this using the product rule
  - Choice 1: rank (2 – Ace) of one of the pairs
  - Choice 2: Suits of the first pair
  - Choice 3: rank (2 – Ace) of the other pair
  - Choice 4: Suits of the other pair

How can we tell that the previous logic is wrong?

- Overcounting:
  - Construct two sequences of decisions that give rise to the same structure
- Undercounting
  - Construct a structure that cannot be created by any sequence of decisions
- Can we do one of these two for the previous example?
Lexicographical Ordering

- Consider lists of distinct elements $P_1 \ldots P_n$
- If we form the Cartesian product $P = P_1 \times \ldots \times P_n$	hen the lexicographical ordering of $P$ is defined as

$$
\text{for } a, b \in P, a <_L b, \text{ iff } \\
\exists i, \ a_i < b_i, a_1 = b_1 \ldots, a_{i-1} = b_{i-1}
$$

Example Problem

- Put the elements of $(a,b,c) \times \text{(apple, pear, banana)}$ in lexicographical order

Example Problem 2

- $(b,\text{apple}), (b,\text{pear}), (c,\text{apple}), (c,\text{pear}), (d,\text{apple}), (d,\text{pear}), (a,\text{apple}),(a,\text{pear})$ are in lexicographical order. Provide a Cartesian Product that gives rise to this lexicographical order

Dictionary Ordering

- Defines an ordering over lists

$$
x <_d y \text{ iff } (\exists i \text{ s.t. } x_i < y_i \text{ and } x_1 = y_1, \ldots x_{i-1} = y_{i-1}) \\
\text{or } (|x| < |y| \text{ and } x_1 = y_1, \ldots x_{|x|} = y_{|y|})
$$