CS395: Analysis of Algorithms

Department of Computer Science

University of Idaho

Julie "Dr. BC" Beeston

bblearn.uidaho.edu

jbeeston@uidaho.edu

JEB 324 – Moscow

Den basement room 6 - CdA

CS 395 – Analysis of Algorithms

Chapter 1 – Introduction

Read 1.1, 1.2

- **BBLearn**
  - **Syllabus**
  - **Schedule**
  - **Notes**
  - **Assignment 0**
- **About the professor**
- **Fundamentals of algorithmic problem solving.**
- **Euclid's algorithm**
- **GCD**
- **Weekly assignment 1**

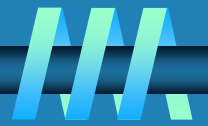
CS 395 – Analysis of Algorithms

Chapter 1 – Introduction

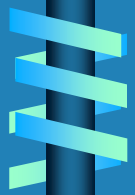
Read 1.1, 1.2

- **BBLearn**
  - **Syllabus**
  - **Schedule**
  - **Notes**
  - **Assignment 0**
- **About the professor**
- **Fundamentals of algorithmic problem solving.**
- **Euclid's algorithm**
- **Weekly assignment 1**

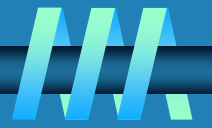
General Course Outline



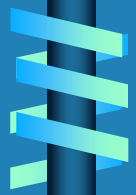
- 1 Introduction
- 2 Fundamentals of the Analysis of Algorithm Efficiency
- 3 Brute Force and Exhaustive Search
  - Assignment 1
  - Midterm 1
- 4 Decrease-and-Conquer
- 5 Divide-and-Conquer
- 6 Transform-and-Conquer
- 7 Space and Time Trade-Offs
  - Assignment 2
  - Midterm 2
- 8 Dynamic Programming
- 9 Greedy Technique
- 10 Iterative Improvement
- 11 Limitations of Algorithm Power
- 12 Coping with the Limitations of Algorithm Power
  - Assignment 3
  - Final Exam



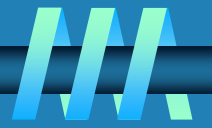
The instructor's Education



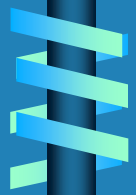
- PhD of Computer Science 2012
  - University of Victoria.
- Master of Computer Science
  - Carleton University 1996
- Bachelor of Science
  - University of Victoria 1994



Previous Teaching Experience



- **University of Idaho / North Idaho College**
  - **CS 241 – Operating Systems**
  - **CS 383 – Software Engineering**
  - **CS 395 – Analysis of algorithms**



Previous Teaching Experience

- **Vancouver Island University**
  - **CSCI 160 – Computing Science I**
  - **CSCI 162 – Topics in Computer Science**
  - **CSCI 260 – Data Structures**
  - **CSCI 261 – Computer Architecture & Assembly Language**
  - **CSCI 265 – Software Engineering**
  - **CSCI 330 – Programming Languages**
  - **CSCI 331 – Object Oriented Programing**
  - **CSCI 370 - Database systems**
  - **CSCI 375 – Systems Analysis**

Previous Teaching Experience

- University of Victoria
  - CSC 230 - Computer Architecture and Assembly Language
  - CSC 355 - Digital Logic and Computer Organization,
  - CSC 370 - Database systems and
  - CSC 360 – Introduction to Operating Systems
- Ambrose University College
  - CSC 100 – Intro to computers
  - CSC 200 – Management Information Systems

Software Designer Work Experience

- **MDS Nordion, Ottawa, Ontario, Canada**
  - **Worked independently or as a senior team member on multiple projects involving the creation and ongoing development of a virtual simulator for cancer radiation therapy.**
- **Ross Video, Ottawa, Ontario, Canada**
  - **Worked as a senior team member developing 3D video drivers for a television switcher.**

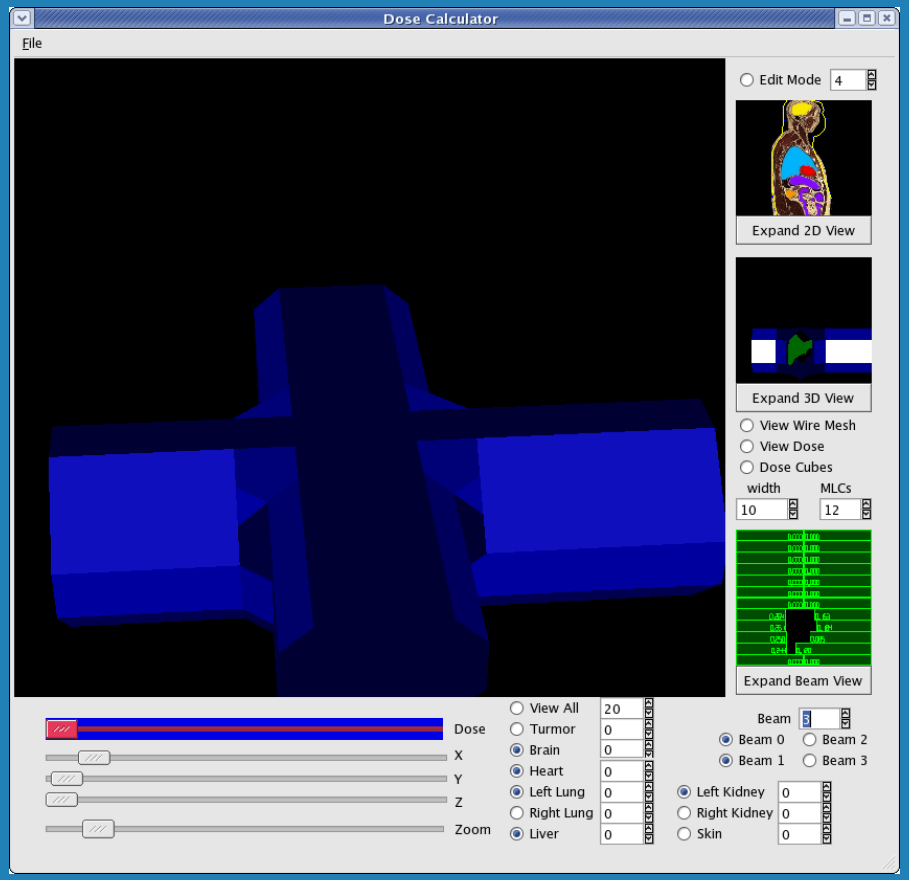
Software Designer Work Experience

- **Lockheed Martin Canada, Victoria, B.C., Canada**
  - **Worked independently or as a senior team member on multiple projects involving the creation and ongoing development of weapons deployment systems for Canadian Naval ships.**
  - **Traveled on site to Canadian naval ship in order to support missile exercises, trouble shoot issues and get ideas for future enhancements for the product.**
- **3xLogic, Victoria, B.C., Canada (Video Camera)**
- **Nortel Networks, Ottawa, Ontario, Canada (Telephone)**
- **Nawitka Resource Consultants, Victoria, B. C. (Maps)**

My Personal Life

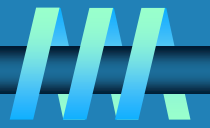
- I moved here from Victoria
- Married 27 years
- 20 year old son
- On Weekends I like:
  - Tubing
  - Boating
  - Dirt Biking
- I have face blindness

My Research: Radio Therapy Simulation

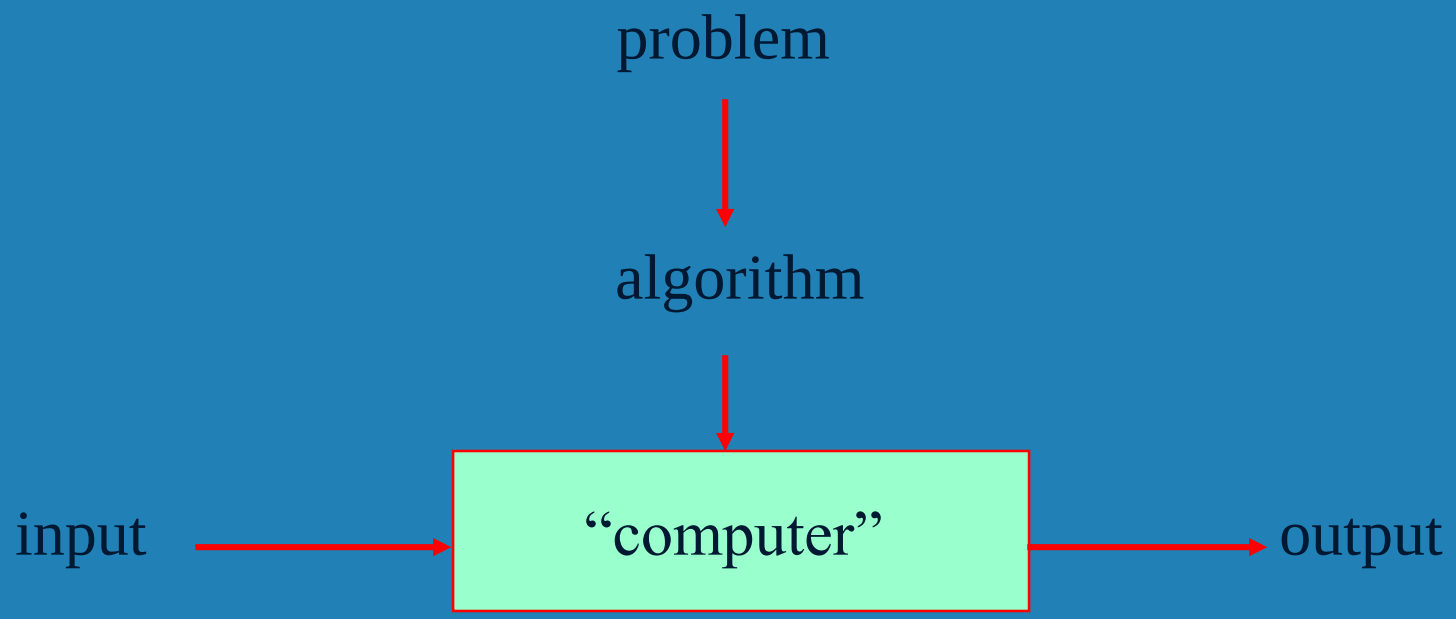


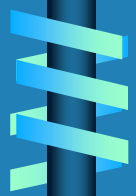
I think I would just cut the wire.

What is an algorithm?

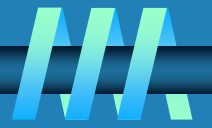


An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

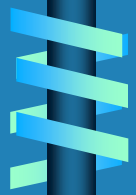




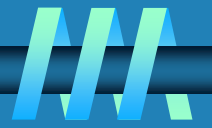
Algorithm Properties



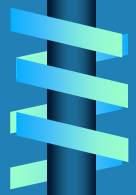
- The requirements at each step of an algorithm are unambiguous and are immutable.
- The range of inputs for which an algorithm works must be specified and followed.
- The same algorithm can be represented in several different ways.
- Several algorithms for solving the same problem may exist.
  - Algorithms for the same problem can be based on very different ideas and can solve the problem with dramatically different speeds (and accuracies).



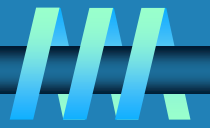
Notion of Algorithm



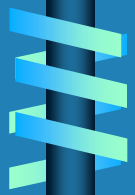
- For a given algorithm there are always three questions to be asked:
  - Is the algorithm correct?
  - How much time does it take, as a function of the input size n, to complete?
  - Is there a faster ("better") algorithm?



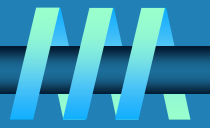
An *algorithm* example



- Computing the *greatest common divisor* of two non-negative integers:
  - **Euclid's algorithm.**
  - Largest integer dividing *m* and *n* evenly.
  - Common prime factorization.



Euclid's Algorithm



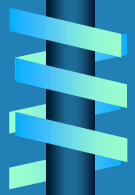
Problem: Find gcd(*m*,*n*), the greatest common divisor of two nonnegative, not both zero integers *m* and *n*

Examples:

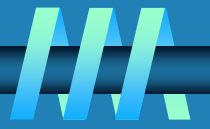
gcd(60,24) = 12

gcd(60,0) = 60

gcd(0,0) = ?



Euclid's Algorithm



Euclid's algorithm is based on repeated application of equality

$$\gcd(m,n) = \gcd(n, m \bmod n)$$

- **where *m* mod *n* is the remainder of the division of *m* by *n***
- **until *m* mod *n* is zero;**
- **gcd(*m*, 0) = *m*, the last value of m is the greatest common divisor of the initial *m* and *n*.**

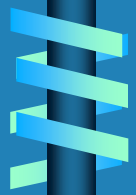
Example:

gcd(60,24) ------ 60 mod 24 = ?

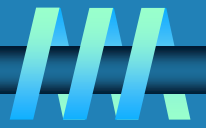
= gcd(24,12) ------ 24 mod 12 = ?

= gcd(12,0) --- Stop

= 12



Two descriptions of Euclid's algorithm



Step 1 If $n = 0$, return m and stop; otherwise go to Step 2

Step 2 Divide m by n and assign the value of the remainder to r

Step 3 Assign the value of n to m and the value of r to n. Go to Step 1.

ALGORITHM *Euclid(m,n)*

//Computes gcd(m, n) by Euclid's algorithm

//Input: Two nonnegative, not-both-zero integers m and n

//Output: Greatest common divisor of m and n

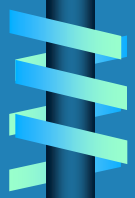
while $n \neq 0$ **do**

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

return m



Other methods for computing gcd(*m*,*n*)

Consecutive integer checking algorithm

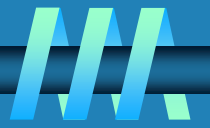
Step 1 Assign the value of min{*m*, *n*} to *t*

Step 2 Divide *m* by *t*. If the remainder is 0, go to Step 3; otherwise, go to Step 4

Step 3 Divide *n* by *t*. If the remainder is 0, return *t* and stop; otherwise, go to Step 4

Step 4 Decrease *t* by 1 and go to Step 2

Other methods for gcd(*m*,*n*) [cont.]



Middle-school procedure

Step 1 Find the prime factorization of *m*

Step 2 Find the prime factorization of *n*

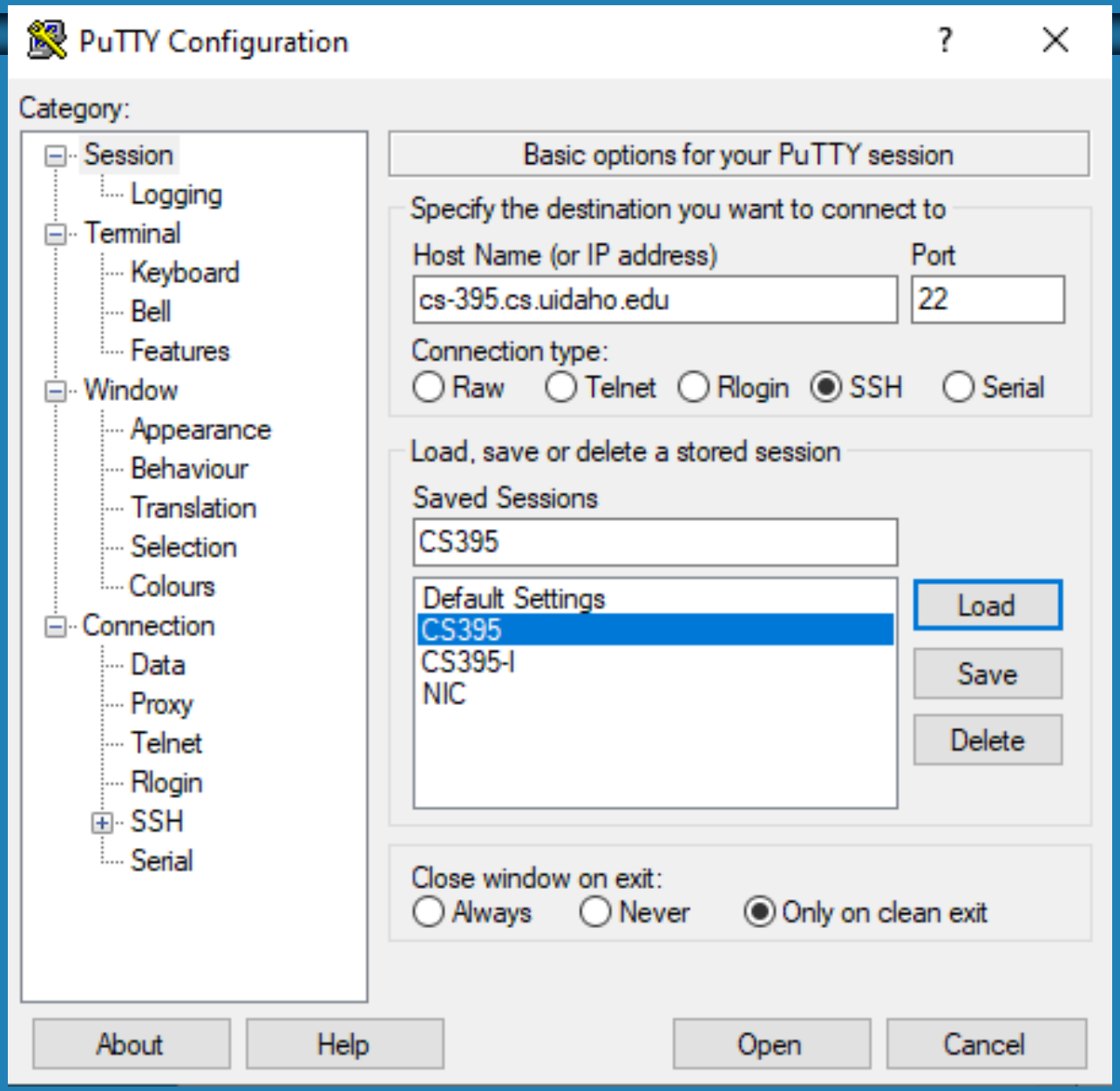
Step 3 Find all the common prime factors

Step 4 Compute the product of all the common prime factors and return it as gcd(*m*,*n*)

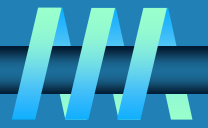
Is this an algorithm?

Remember, for an algorithm:
The requirements at each step of an algorithm are unambiguous and are immutable.
The range of inputs for which an algorithm works must be specified and followed.
Several algorithms for solving the same problem may exist.

Weekly Assignments



W01 Assignment – lets cut the wire



ALGORITHM *MinDistance*$(A[0..n-1])$

//Input: Array $A[0..n-1]$ of numbers

//Output: Minimum distance between two of its elements

$dmin \leftarrow \infty$

for $i \leftarrow 0$ **to** $n-1$ **do**

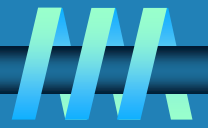
for $j \leftarrow 0$ **to** $n-1$ **do**

if $i \neq j$ **and** $|A[i]-A[j]| < dmin$

$dmin \leftarrow |A[i]-A[j]|$

return $dmin$

W01 Assignment – lets cut the wire



ALGORITHM *MinDistance*$(A[0..n-1]$

//Input: Array $A[0..n-1]$ of numbe

//Output: Minimum distance between two of its elements

$dmin \leftarrow \infty$

for $i \leftarrow 0$ **to** $n-1$ **do**

for $j \leftarrow 0$ **to** $n-1$ **do**

if $i \neq j$ **and** $|A[i]-A[j]| <$ *min*

$dmin \leftarrow |A[i]-A[j]|$

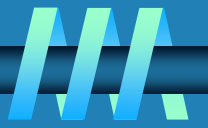
return $dmin$

Hand this in "as is" by Before next class.

Think of ways to make the algorithm better.

How many times does this "if" get called?

Submitting Assignments



/y/shared/Engineering/cs-drbc/submit

Create a .alias file and add this line:

alias drbcsubmit="/y/shared/Engineering/cs-drbc/submit"

(Be careful of the " if you copy and paste)

Add this line to the end of your .bash_login file (Create one if you do not have one.)

source .alias