Computer Science for the Physical Sciences

Week 5

Craig Rasmussen (Research Support Services, University of Oregon)

Computer Science Minor: *Review*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures

- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Functions and pipes
  - Operating Systems
  - Automata Theory
  - Software Methodology I-II
    Revision Control and Make Files
  - Introduction to Compilers
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

Computer Science Minor: *This Week*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Python and nm
  - Operating Systems
  - Automata Theory Regular expressions
  - Software Methodology I-II Debugging
  - Databases netCDF
  - ~~Introduction to Compilers~~
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

Computer Science Minor: *Databases*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Python and nm
  - Operating Systems
  - Automata Theory Regular expressions
  - Software Methodology I-II Debugging

Databases netCDF

  - ~~Introduction to Compilers~~
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

Introduction to Relational Databases

- Think of a relational database as a set of spreadsheet files (called tables)
  - columns are attributes
  - rows are records
- Example I: Constellation table
  - attributes: (ID, name, North/South, location_in_sky)
  - will have 88 rows
- Example II: Star table
  - attributes: (ID, name, galaxy, magnitude, is_variable, constellation)
  - will have LOTS of rows

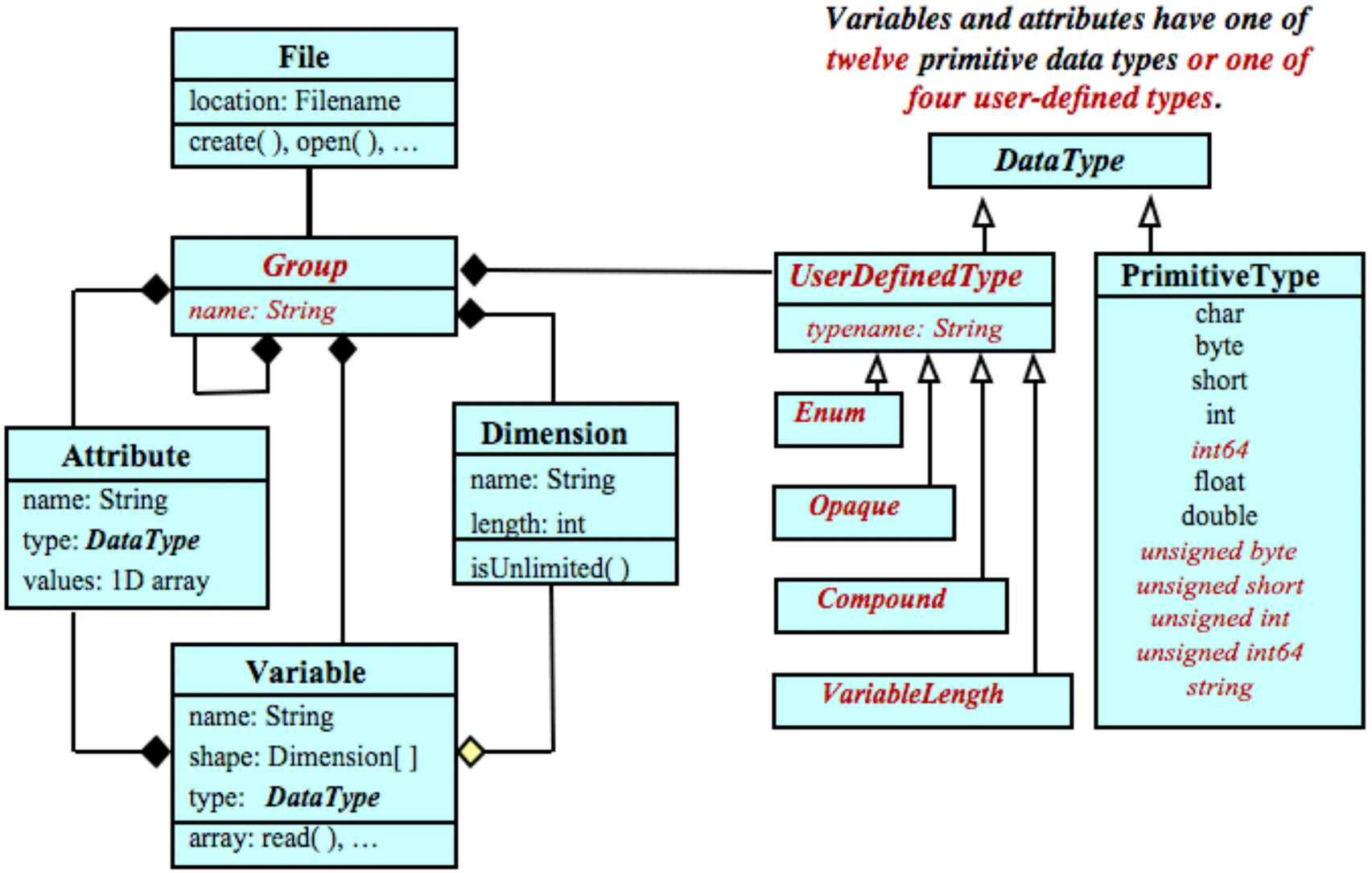
Operations on Relational Databases

- From A and B, join
  - performs outer product of tables A and B
  - this creates a big logical table
- Can reduce size of resulting join to make it more manageable
  - Projection (eliminate columns)
  - Where Filter … (eliminate rows)
- Perform operations on the result
  - sort
  - count
  - ….

Introduction to NetCDF:

- NetCDF is a self describing data format
- Lots of useful information at http://www.unidata.ucar.edu/software/netcdf/workshops/2010/netcdf4/index.html
- Groups provide a scope for names and a scalable way to organize data objects
- Dimensions are like, well dimensions
  - time, lat, long
- Variables can contain multiple dimensions
  - Temperature(time, lat, long)
- Attributes store data about data (ancillary data or metadata)

NetCDF Data Model:
Groups, Dimensions, Variables, and Attributes



A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

NetCDF file example:

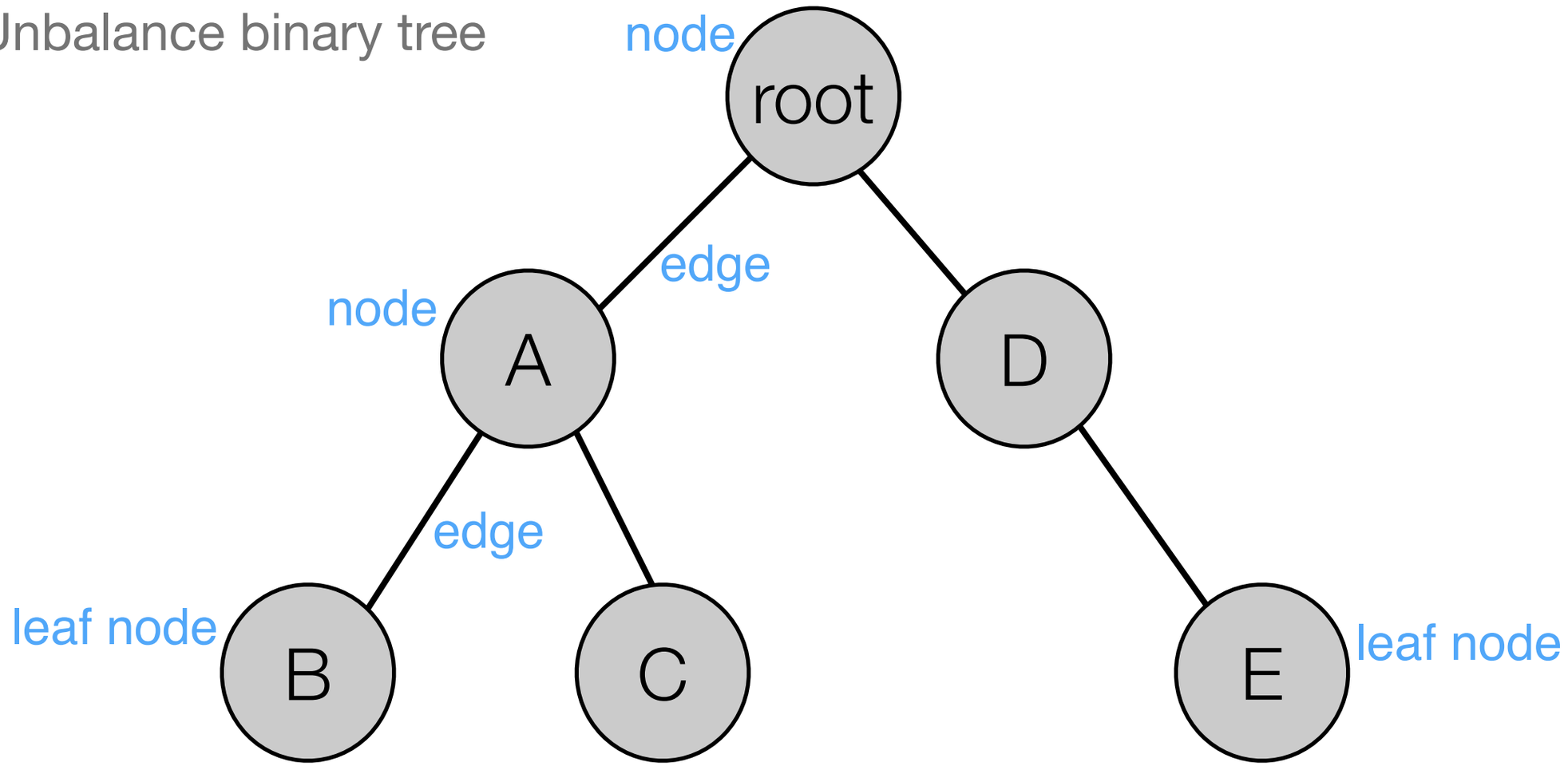
```
File “surfdata_1.9x2.5_simyr1850_c091108.nc"

  dimensions:
     lat = 96;
     long = 144;
     time = UNLIMITED;   // (12 currently)

  variables:
     double LATXY(lat=96, long=144);
        :long_name = “latitude”;
        :units = “degrees north”;
     int time(time=12);
        :long_name = “Calendar month”;
        :units = “month”;

// global attribures
:Conventions = “NCAR-CSM”;
:Source = “Community Land Model: CLM3”;
:Glacier_raw_data_file_name = "mksrf_glacier.060929.nc";
:Revision_Id = "$Id: mkfileMod.F90 18909 2009-10-15 19:12:09 erik $";
```

Computer Science Minor: *Regular Expressions*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Python and nm
  - Operating Systems
  - Automata Theory Regular expressions
  - Software Methodology I-II Debugging
  - Databases netCDF
  - ~~Introduction to Compilers~~
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

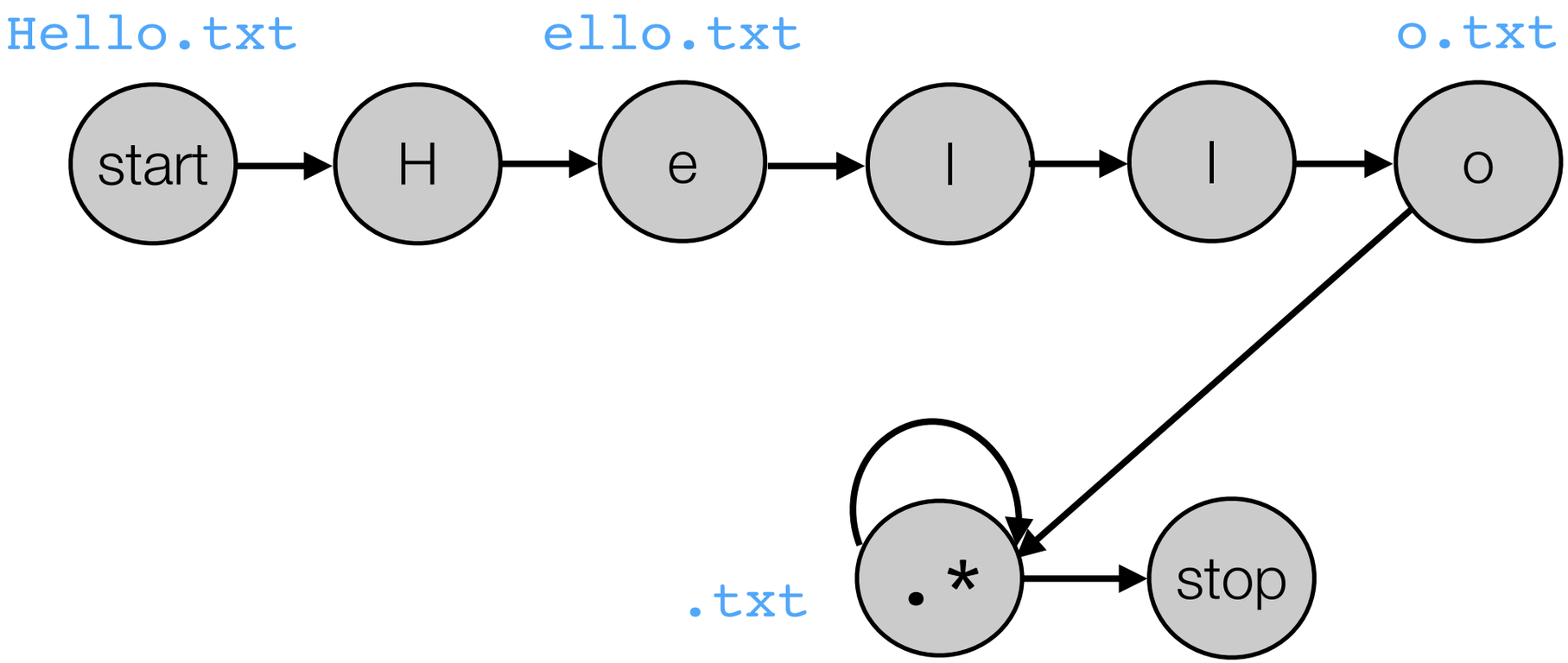
Graphs

- A graph is a representation of a set of objects where some pairs of objects (nodes) are connected by links (edges)
- Unbalance binary tree



Finite Automata

- A finite automata to find all set of strings matching regular expression“`Hello.*`”



Regular Expressions: pattern matching

```
. matches any single character (excluding newlines)

[] matches a single character within the brackets

[^] matches a single character not within the brackets

* matches the preceding element zero or more times

+ matches the preceding element one or more times

? matches the preceding element zero or one times
```

Regular Expressions: by example

.at matches any three-character string ending with "at", including "hat", "cat", and “bat"

[hc]at matches "hat" and “cat"

[^hc]at matches all strings matched by .at other than "hat" and “cat"

^[hc]at matches "hat" and "cat", but only at the beginning of the string or line

s.* matches any number of characters preceded by s, for example: "saw" and "seed"

Computer Science Minor: *Bash command nm*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Python and nm
  - Operating Systems
  - Automata Theory Regular expressions
  - Software Methodology I-II Debugging
  - Databases netCDF
  - ~~Introduction to Compilers~~
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning

Shell Command: *nm*

- nm - display name list (symbol table)
- Steps to build an executable from a source file (*.f90)
  - compile program -> $(FC) -c -I include_path hello.f90
  - link program -> $(FC) -o hello hello.o -L library_path -lsome_library
- hello.o is an object file and contains symbols (e.g., functions) and code
- hello is an executable file (created by linker from *.o and libraries)
- What happens if a symbol (function code) can’t be found by linker?
  - linker can’t create an executable if all dependencies aren’t satisfied
  - use nm to track down missing symbols

Computer Science Minor: *Debugging*

- Required courses (24 credits)
  - Introduction to Computer Science I-II-III
  - Elements of Discrete Mathematics I-II
  - Introduction to Data Structures
- Upper-division courses (8 credits)
  - Computer Architecture
  - Introduction to Algorithms
  - C/C++ and Unix Python and nm
  - Operating Systems
  - Automata Theory Regular expressions
  - Software Methodology I-II Debugging
  - Databases netCDF
  - ~~Introduction to Compilers~~
  - Computational Science
  - Bioinformatics
  - Data Mining
  - Introduction to Artificial Intelligence
  - Machine Learning