ISC-4221

ALGORITHMS for SCIENTIFIC APPLICATIONS II
(Discrete Algorithms in Scientific Computing)

Syllabus

ISC4221-1 ALGORITHMS for SCIENTIFIC APPLICATIONS II

Instructor

Peter Beerli
Office: 150-T DSL
Email: beerli@fsu.edu
Phone: (850) 559-9664

Teaching Assistant

Philip Boehner
Office: _____
Email: pb12c@my.fsu.edu

Lectures (Beerli):

Tuesday, Thursday 12:30am-13:45pm
HCB 217

Lab-session (Boehner):

Thursday 3:30pm - 6:00pm
Dirac Science Library Room 152

Office Hours

Tuesday 2:00-2:45 (Beerli)
Thursday 2:00-2:45 or by appointment (Beerli).
TBA (Boehner)

Textbook

No textbook required

Objectives

This course provides the student with an introduction to algorithms used for solving discrete problems such as sorting or searching an array, scheduling, determining an optimal path (such as the well-known traveling salesman problem), extracting and interpreting data, etc. In addition to introducing the student to common algorithms for various problems, this course also provides the student with tools to analyze algorithms so that algorithms which solve the same problem can be compared.

Content

The course is divided into eight parts:

- Part I - Introduction to Algorithm Design and Analysis
- Part II - Random Processes
- Part III - Graph Theory
- Part IV - Data Mining
- Part V - Clustering
- Part VI - Optimization
- Part VII - Feature Extraction and Pattern Recognition
- Part VIII - Computational Geometry

Assignments

The assignments consist of homeworks, lab-reports and a final project.

- Homework: Each **homework** assignment must be sent as a PDF to beerli@fsu.edu. The subject line MUST consist of **ISC-4221: homework homeworknumber**, the attached PDF MUST have a filename with your lastname and the homework-number, for example beerli1.pdf, berkley2.pdf. I will deduct 10 points (out of 100) for not following these submission guidelines. Contents for each homework will be graded for correctness and being concise, but wordy enough that I can follow your thought-process.
- Lab reports: Each labreport must be sent to Philip Boehner (pb12c@my.fsu.edu). The subject line MUST consist of **ISC-4221: lab labnumber**. Each lab assignment consists usually of a short report that includes instructions how to run the (matlab) program, and the source code. The labreport must be formatted as a PDF and has a filename with your lastname and the lab-number, for example beerli1.pdf or berkley2.pdf. The report **and** the programming source used to generate the results must be packaged into a single zip or tar.gz archive and attached as a **single** attachment to the email. Deviations, such as submitting multiple attachments, submitting the report as word file, or packaging both as a rar file will be penalized by 10 points out of 100.
- Final report: During the semester you will choose a topic for your final report; you will then research this topic and present the results to the class in a short 5 minute presentation during the finals week. For grading of your presentation, I will use your presentation performance and your slides (you need the send them to me the day before the presentation); I may ask you for the programming source code of your final project to refine your grade if I cannot establish what or how you found your results.

We may compare homework, lab reports, and projects using Turnitin and other comparison tools.

Late Assignments

You can turn in one laboratory assignment and one home- work late with no questions asked and no penalty; however, the assignment must be turned in no later than 1 week after its due date. Additional late assignments will be penalized by applying a graded scale which terminates with a 25% reduction at the end of one week; no assignments will be accepted more than a week past the due date. Exceptions to these rules are made only if extenuating circumstances (such as illness, etc.) arise which can be documented.

University Attendance Policy

Excused absences include documented illness, deaths in the family and other documented crises, call to active military duty or jury duty, religious holy days, and official University activities. These absences will be accommodated in a way that does not arbitrarily penalize students who have a valid excuse. Consideration will also be given to students whose dependent children experience serious illness.

Academic Honor Policy

The Florida State University Academic Honor Policy outlines the Universitys expectations for the integrity of students academic work, the procedures for resolving alleged violations of those expectations, and the rights and responsibilities of students and faculty members throughout the process. Students are responsible for reading the Academic Honor Policy and for living up to their pledge to "... *be honest and truthful and .... [to] strive for personal and institutional integrity at Florida State University.* (Florida State University Academic Honor Policy, found at http://dof.fsu.edu/honorpolicy.htm.)

Americans With Disabilities Act

Students with disabilities needing academic accommodation should: (1) register with and provide documentation to the Student Disability Resource Center; and (2) bring a letter to the instructor indicating the need for accommodation and what type. This should be done during the first week of class. This syllabus and other class materials are available in alternative format upon request. For more information about services available to FSU students with disabilities, contact the:

Student Disability Resource Center
874 Traditions Way 108
Student Services Building
Florida State University
Tallahassee, FL 32306-4167

voice: (850) 644-9566
TDD: (850) 644-8504
sdrc@admin.fsu.edu
http://www.disabilitycenter.fsu.edu/

Free Tutoring from FSU

For tutoring and writing help in any course at Florida State University, visit the Academic Center for Excellence (ACE) Tutoring Services comprehensive list of tutoring options - see http://ace.fsu.edu/tutor or contact tutor@fsu.edu for more information. High-quality tutoring is available by appointment and on a walk-in basis. These services are offered by tutors trained to encourage the highest level of individual academic success while upholding personal academic integrity.

Syllabus Change Policy

Except for changes that substantially affect implementation of the evaluation (grading) statement, this syllabus is a guide for the course and is subject to change with advance notice.

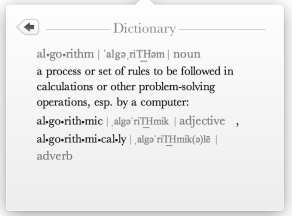
Questions, Concerns?

Websites

Tour of Class

What is an Algorithm

What is an Algorithm



What is an Algorithm

- An algorithm is a description of a procedure which terminates with a result.
- An algorithm is a step-by-step problem-solving procedure, especially an established, recursive computational procedure for solving a problem in a finite number of steps.
- An algorithm is a sequence of unambiguous instructions for solving a problem by obtaining a required output for any legitimate input in a finite amount of time.

What is an Algorithm

- An algorithm is a sequence of unambiguous instructions for solving a problem by obtaining a required output for any legitimate input in a finite amount of time.

The term unambiguous can not be stressed enough - we must be precise! For example, consider the following description of multiplying two n x n matrices, A, B.

Why is it not clear?
What should we add?

Multiply each row of matrix A times each column of matrix B

Multiply each row of matrix A times each column of matrix B

This doesn't tell someone what the input or output is, how to form the multiplication, etc. It would be much clearer to write it as the following

Input: $n \times n$ matrices A, B
Output: an $n \times n$ matrix C which contains the product of A and B

```
for i = 1, n
    for j=1, n
      c(i,j) = sum_{k=1}^{n} a(i,k)*b(k,j)
```


Input: $n \times n$ matrices A, B
Output: an $n \times n$ matrix C which contains the product of A and B

```
for i = 1, n
    for j=1, n
      c(i,j) = sum_{k=1}^{n} a(i,k)*b(k,j)
```

Note that we add the caveat that it must work for any legitimate input. For example, if we are writing a routine for calculating the square root of a real number, we don't expect it to work for a negative number. Usually we code a test to make sure that the input is legitimate so we have a "nice" error message.

Why do we need to study algorithms?

Algorithms are the basis of computer programs and the computations generated by them are now used throughout society. For example, airplane wings are now designed using computers, decisions made concerning global issues such as climate change, groundwater contamination, etc. all rely on computer simulations.

In fact, computations have joined theory and experimental as the pillars of scientific discovery.

What are the goals we are setting for Algorithms I & II?

- To learn a standard set of algorithms from different areas of computational science;
- To see how these algorithms can be used to solve standard problems in scientific computing;
- To be able to analyze algorithms as to efficiency, accuracy, and convergence;
- Be able to compare algorithms as to efficiency and accuracy;
- To begin to see how to design new algorithms.
- To understand the difference between continuous and discrete problems.

What is the main difference between the courses Algorithms I & II?

In Algorithms I we are concerned with numerical problems which typically involve mathematical objects of a continuous nature such as approximating an integral, solving a system of linear equations, finding the roots of a function, solving a differential equation, etc.

In Algorithms II we are mainly interested in problems of a discrete nature such as searching for a text string, sorting a list of objects, finding the optimal path between cities, finding the point from a list which is closest to a given point, simulating a random process, etc.

Discrete vs. Continuous

- The main distinction between Algorithms I and II is that the first deals with continuous problems and the second mainly deals with discrete problems. What do we mean by this?
- Real numbers have the property of varying smoothly so when we integrate a function f(x) from x=a to x=b we expect f to take on all values between a and b.
- In contrast, the objects studied in discrete mathematics (such as integers, graphs, logical statements, etc.) do not vary smoothly in the same way that real numbers do. In fact they have distinct, separated values.
- Classic examples of discrete problems are sorting, searching, optimization using a set of discrete objects, operations dealing with images, etc. Graphs are often used to analyze discrete problems.

Discrete vs. Continuous

- However, with today's technology the distinction between continuous and discrete sometimes becomes blurred. For example display monitors are an array of dots and are therefore discrete. However, the dots are so tiny that the depiction of a continuous function such as sin x looks like a continuous function on the screen. Another example is the number of colors available to view your photos on a screen. The number is so large that for all practical purposes it looks like we are using a continuous color spectrum.
- We will see that the algorithms to solve discrete problems are different from those for solving continuous problems but some of them have aspects in common with algorithms for continuous problems.
- As is the case for continuous problems, there will be several algorithms to solve a particular problem. Usually no one algorithms works for all instance of the problem.

Some important types of problems

- **Sorting**

  Examples include

  - a list of names which we have to sort alphabetically;
  - a vector consisting of numbers which must be rearranged to appear in ascending order;
  - a deck of n cards to be shuffled;
  - sorting a list of students by their GPA;
  - sorting a list of TVs by cost.

Sorting algorithms often require extra memory.

There is no single sorting algorithm that is best in all cases.

Some important types of problems

2. Searching
Examples include

- searching a DNA string to find genes or repeats etc

  TGTAGAACTGTGTGTCACACACACATACACATACCTATATGAG

- searching for a given value x in a numerical array a(1 : n);
- given a set of points X, find the point z in X which is closest to some given point,
- determining if any two elements in an array are equal

If we are searching a list and it is already sorted, then our algorithm should take advantage of this.

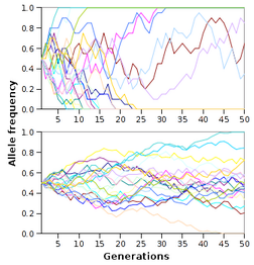
There is no single searching algorithm that is best in all cases.

21

Some important types of problems

3. Randomness
Examples include

- Generating pseudorandom number
- Random walks - stock market, path of a foraging animal, allele frequences through time, ...
- Brownian motion



22

Some important types of problems

4. Graph problems
Examples include

- Königsberg Bridge Problem (1735)

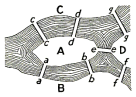


FIGURE 98. *Geographic Map: The Königsberg Bridges.*

The Königsberg bridge problem asks if the seven bridges of the city of Königsberg (left figure; Kraitchik 1942), formerly in Germany but now known as Kaliningrad and part of Russia, over the river Preger can all be traversed in a single trip without doubling back, with the additional requirement that the trip ends in the same place it began. This problem was answered in the negative by Euler (1736), and represented the beginning of graph theory.

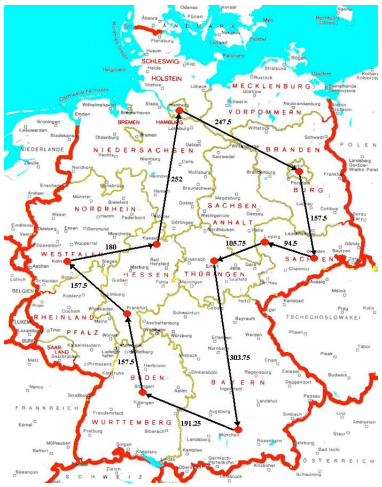
- Minimum spanning tree: An example would be a cable TV company laying cable to a new neighborhood. If it is constrained to bury the cable only along certain paths then there would be a graph representing which points are connected by those paths. Some of those paths might be more expensive, because they are longer, or require the cable to be buried deeper; these paths would be represented by edges with larger weights. A spanning tree for that graph would be a subset of those paths that has no cycles but still connects to every house. There might be several spanning trees possible. A minimum spanning tree would be one with the lowest total cost.

23

Some important types of problems

5. Optimization problems
Examples include

- Traveling salesman problem

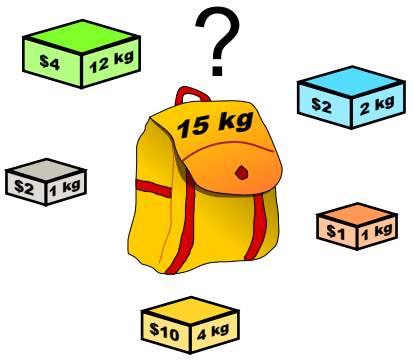


24

Some important types of problems

5. Optimization problems
Examples include

- Knapsack problem



Some important types of problems

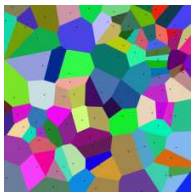
6. Data mining and clustering problems
Examples include

- Fraud/Crime detection (e.g. dentist-girl-friend HIV murder attempt)
- Census data
- Medical statistics
- Stock market

Some important types of problems

7. Computational Geometry
Examples include

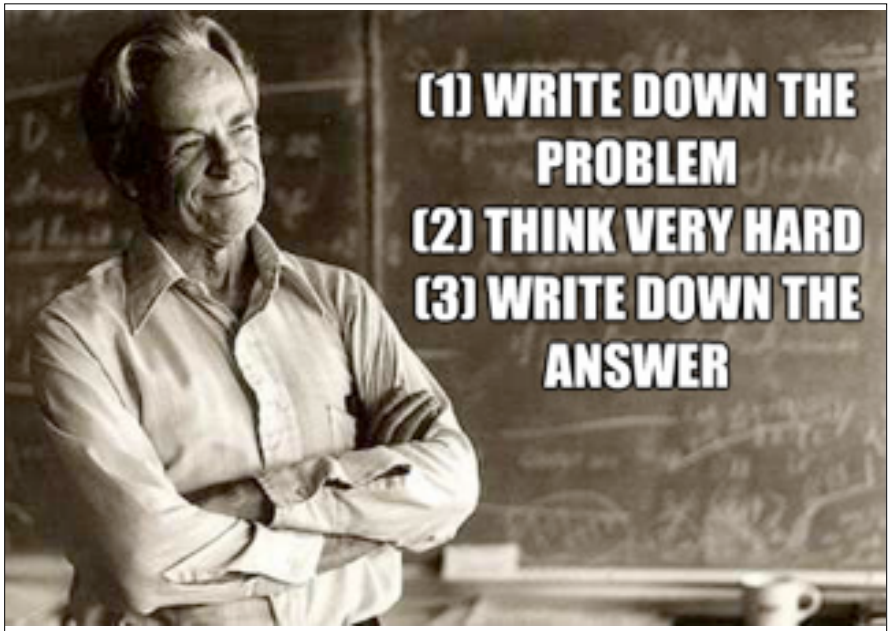
- Grid generation
- Convex hull: given a set of points, find the smallest convex polyhedron/ polygon containing all the points.
- Voronoi diagrams



Some important types of problems

8. Image processing
Examples include

- Feature extraction
- Edge detection
- Image enhancement
- Printing


(1) WRITE DOWN THE PROBLEM
(2) THINK VERY HARD
(3) WRITE DOWN THE ANSWER

29