Lecture 0: Overview of cs2504/CS4092

Dr. Kieran T. Herley

Department of Computer Science
University College Cork

2013/14

cs2504/cs4092 Basics

Who

Me Dr. Kieran Herley
Room G.63, WGB
`k.herley@cs.ucc.ie`
You Single Honours Computer Science and 4Y Web Stream

What

Title Algorithms and Linear Data Structures
Weight Period 2; 5 credits

When and Where

Lectures

Tuesday	12-1	WGB G08
Thursday	11-12	WGB 106

Practicals

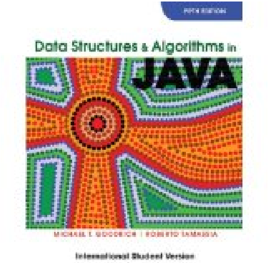
- one session per week per student
- Time, place TBA

Office Hours Fri 2-3, WGB G63

Useful Resources

Useful Text

- No required text but some material taken from following text (copies in library).
- Data Structures and Algorithms in Java
  M. T. Goodrich and
  R. Tamassia.
  Wiley.
- Also your cs2500 materials.


FIFTH EDITION
Data Structures & Algorithms in
JAVA
MICHAEL T. GOODRICH | ROBERTO TAMASSIA
International Student Version

Webpage (soon)

- `www.cs.ucc.ie/~kieran/cs2504`
- Code. Assignments. Handouts. Lecture slides (.ps, .pdf)

Labs and Assignments

- Weekly two-hour sessions
- Assignments (around five)
- Contribute to final mark
- Java based
- Begin shortly

Expectations

Time

	week^{-1}	total
lectures	2	24
practicals	2	20 (approx.)
study	2	24 (min.)
subtotal	6	68
pre-exam study		22
total		90

General

- You are expected to attend every lecture and every lab
- You are expected to attempt and submit every assignment

Assessment (CS2504)

Breakdown

Year's work	20 %
End-of-year exam	80 %

Year's work

- marked assignments

End-of-Year Exam

- Formal, 1.5-hour paper
- Details of format later

Six Ways to Fail cs2504/cs4092

- Don't bother attending lectures
- Don't review lecture materials, read book or study examples covered in lectures
- Ignore practicals and assignments
- Avoid programming
- Rely on memorising material rather than understanding it
- Leave everything until the end and cram like mad

What cs2504/cs4092 Covers

- Data structures
  - Concepts and techniques for representing and interacting with an application's data
  - Key part of every programmer's toolbox
- Algorithms–Techniques for efficient manipulation of data
- Problem solving
- Reasoning about computation

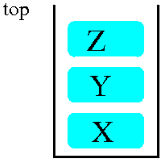
. . . .

- (Some Java programming)
  - Implementing data structures
  - Developing applications employing data structures

Some Abstract Data Types

Stack

- Container concept
- Last-in first-out behaviour

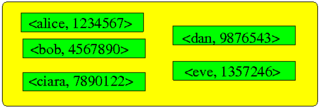


Some Applications

- simple calculator
- simple HTML validator

Map

- container for (key, value) pairs
- searchable by key

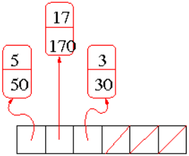


<alice, 1234567>
<bob, 4567890>
<ciara, 7890122>
<dan, 9876543>
<eve, 1357246>

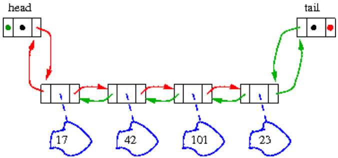
Some Applications

- personal telephone book
- very useful tool in many situations

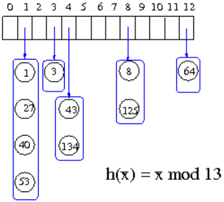
Some Implementation Techniques



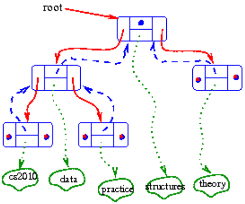
Arrays



Linked Lists



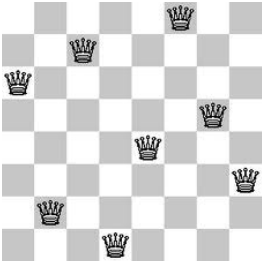
Hashing



Binary Trees

Problem Solving and Reasoning

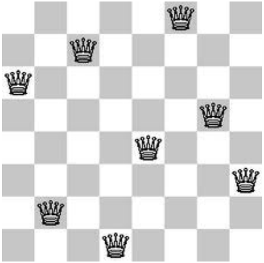
Problem



How position eight queens on a chess board so no two conflict? Generalize to $n \times n$.

Problem Solving and Reasoning

Problem

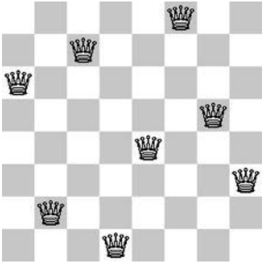


How position eight queens on a chess board so no two conflict? Generalize to $n \times n$.

How create algorithm to find solution(s)? Idea: systematically explore possible placements, skipping over those embodying conflicts

Problem Solving and Reasoning

Problem



How position eight queens on a chess board so no two conflict? Generalize to $n \times n$.

How create algorithm to find solution(s)? Idea: systematically explore possible placements, skipping over those embodying conflicts

How to establish that "algorithm" works?

A Typical Application

Setting

it was the best of times it was the worst of times
1
2 abc
3 def
4 ghi
5 jkl
6 mno
7 pqrs
8 tuv
9 wxy
*
0
#

- Text prediction facility for mobile phones
- Usual encoding: $\boxed{2}$ → a/b/c *etc.*

Goal

- Given a key sequence, *e.g.* $\boxed{2}$ $\boxed{2}$ $\boxed{8}$, present for selection a list of words consistent with that sequence *e.g.*
  [0] cat, [1] act, [2] bat, [3] active, [5] cattle, *etc.*
- User selects word by entering number *e.g.* $\boxed{2}$ for "bat".

Challenges

Conceptual? Implementational?

Implementational Hurdles

- Am I familiar with the Java elements I need?
- How should I design/structure the application?
- How do I express my ideas in Java?
- How do I debug it?

Conceptual Obstacles

- How to represent the information (data structures)?
- How to manipulate same to achieve desired effect (algorithms)?
- How do we tell a “good” data structure/algorithm from a “bad” one? Does it matter?

One Possible Approach

Idea

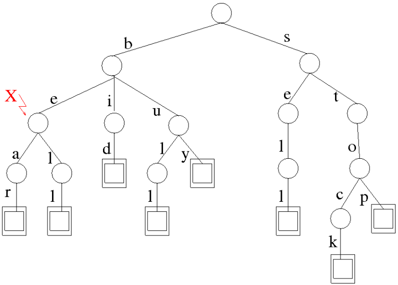
- Maintain “list” of common English words
- Search “list” for words consistent with key sequence

Representing the “list”

- array of words
- linear sweep to find desired words

Observation List needs to be large to be effective. Very bulky to store. Very slow to scan. Is there a better way?

Clever Representation–Trie



- Stores words at bottom
- Each top-to-bottom "path" gives spelling of a word
- "Prefix-similar" words grouped together
- (More details later)

May not see tries in cs2054, but points to benefit of "clever" techniques for data representation and algorithms