Revision Lecture

CS3D5A Data Structures & Algorithms
Trinity College Dublin
3 April 2017

Anton Gerdelan <gerdela@scss.tcd.ie>

Exam Revision

- Dates/places announced
- Sample exam on Blackboard
  - has more questions (pages) than the real exam
  - same question type/style/structure
  - not going to provide answers

Exam Philosophy

- Exams are pretty silly tests
  - high pressure, overly-specific, Google in 5 mins
- but our exam weight is 60%
  - how many points do you need after your internal grade?
    you're welcome
- good excuse for refreshing your Algorithms learning
  - you'll make <u>better</u> software than most
  - you'll be allowed entry to the United States*
  - interviewers often set a screening test of Algorithms exam Qs
    - keep your exam question paper?

Exam Structure

- 4 questions (pages) - choose your favourite 3
- each question is worth 20%
    - 2 easy short-answer questions ~5 marks total
        - e.g. "define these terms relating to trees"
    - 1 working problem ~10 marks
        - e.g. "*write a function to insert a node...*" "*write/draw the steps of this algorithm*"
    - 1 deeper understanding Q ~5 marks
- roughly 1 major topic per question
- supplementary exam has slightly different mix of Qs

Skills You Need

- Can talk about **asymptotic** notation
  - quote a few best/worst in a comparison table
  - know what limitations of this are
- Can write some **short functions in C** code
  - searches, sorts, hash tables, lists, trees, etc.
  - don't worry about semi-colons, exact language rules etc - not going to compile it
  - will pick up on logical errors and edge cases

More Skills You Need

- Can draw **diagrams** of data structures and steps in algorithms
  - e.g. linked lists, trees, graphs, hash tables
  - label data structures with correct terms for features
  - for code questions **consider drawing a diagram anyway**. less likely to make mistakes in code.

Major Topics

- **Arrays** and **Linked Lists** - diagrams, types, simple operations, pointers and loops.
- **Sorts & Searches** - which algorithm with which data structure? comparison of algorithms.
- **Hash Tables** - advantage and problems? can you compare some types?
- **Trees** - diagrams, types of tree, algorithms on trees
- **Graphs** - diagrams, types of graph, algorithms on graphs
- all considering **Complexity** - measuring and approximating

Sorts and Searches

- What are the **elementary sorts**?
- What do are all the **key terms** describing features of e.g. sorting algorithms?
- **Review your assignments** - can you rewrite your simple array sorts and searches?
- Can you list several sort algorithms and **compare** their complexities and other features?
    - bigocheatsheet.com might be handy to double-check

Trees

- what types of tree data structure did we look at?
- why use trees? what is each type for?
- can you draw a diagram of e.g. a binary tree and label the features?
- can you write a simple tree data structure and function e.g. search or insert?
- are your tree functions recursive or iterative?

Graphs

- Key terms used to describe types of graph and graph algorithms?
- What algorithms did we look at? What is the goal of each?
- How do graphs relate to trees?
- Can you **step through** a couple of graph algorithms on paper
  - writing changes to values line-by-line
  - with diagrams - new diagram at each step
- All but v simple coding Q too involved for a 2-hour exam

Walk-through example with Dijkstra

- several vertices/nodes connected with edges
- edges have a fixed cost/weight
- what are the working **variables**?
- how do we know when to **halt**?
- (can do working on a diagram if easier)
- walk-through notation is usually like a table
  - column for each variable - row for each step/iteration

working variables

step	current node	A perm/cost/parent	B perm/cost/parent	C perm/cost/parent	D perm/cost/parent
0	A	yes, 0, none	no, inf, none	no, inf, none	no, inf, none
1			no, **2**, **A**	no, **4**, **A**	
2	**B**		**yes**, 2, A		
...	...	...	...	...	...

- something like that - could have combined steps 1&2
- only need to indicate when variables change value
- remember that current node is chosen from *any* non-permanent nodes (not just neighbours of previous current)

Hash Tables

- key terms to describe hashing algorithms and hash tables?
- can you explain and compare a couple of hash table types?
- can you write a hash function in C?
    - what do you need to be careful of in hash function design?
- can you discuss the variables you can tweak in hash table design?

Complexity

- Explain how Big-O works
  - can you rank Big-Os in order of complexity?
  - why it's useful
  - what it misses
- Can you discuss some additional time or space issues not considered by Big-O?
  - e.g. with respect to computer hardware
  - e.g. other algorithm features.

How to Prep

- **Draw a diagram** of a graph with all the possible features and terms labelled on it (easier to remember visually)
- Same for a tree and linked list
- **Write C data structures** for a linked list/tree/graph node with all the right pointers.
- Find a **simple 5~10 line function** to write in C or pseudo-code using each.
    - longer code problems are more likely to be diagrams or "*for each step in algorithm X, write a line giving change in value of variables.*"
- Go over slides, assignment problems.
    - no specific Qs on case studies like BSP tree.
- We can do some of this shortly...

How to Prep

- Write **tables** of all the data structures and algorithms we studied.
- **Explain the difference** and pros/cons of a couple of algorithms to an **innocent victim** (family member or housemate).
    - They might ask you a question you don't know the answer to...
- If you can answer all the sample paper questions you're going to do pretty well
- Khan Academy has lots of algorithms questions and coding Qs
- If you have a textbook - first few sample Qs at end of every chap.