Exam Review

Anton Gerdelan <gerdela@scss.tcd.ie>

Exam Structure

- 2 hour exam worth 60% of grade total
- 4 pages - one question per page {1., 2., 3., 4.}
    - there are {a), b), c), d)} parts to each question
- roughly 1 major topic per page
- answer any 3 of the 4 questions = answer 3 pages

Major Topics

- Linked lists
- Hash tables
- Trees
- Graphs
- Complexity Analysis
- Search and sort algorithms

Not in Exam

- Debuggers, static analysis
- Writing images
- Makefiles, build systems, libraries
- case studies {Binary Space Partitioning, Genetic Algorithms/road traffic}
- C++ features, <T> templates

Question Break-Down

- **a), b)** short-answers ~5 marks total
  - *define the following terms*…
  - *give two advantages and two disadvantages of*… [algorithms or data structures]
  - *draw a diagram of* [some data structure]
  - *state two* [algorithms or data structures] *that* ….

Question Break-Down

- **c)** problem-solving ~10marks
  - *draw diagrams to show* [some algorithm] *on* [some data structure]
  - *write a function in C that implements* [some algorithm]
  - *consider the following C code*…

Question Break-Down

- **d)** depth of understanding ~5 marks

  - *draw diagrams clearly showing the steps* [more complex]

    - *with reference to time and space complexities, and other features, compare the following*

    - *clearly explain* [in writing] *the steps necessary in* [more complex algorithm]. *draw a diagram of the result*

What to do

- Read all the questions
- Guess your expected grades for each page
- Take the best 3 pages

How grading works

- I write a **grading scheme** for each exam
- For each question
    - "*student gets 1 mark for mentioning **any of the following** concepts - maximum 3 marks*"
        - i put a list of e.g. all 10 possible concepts
    - "*diagram should show the following steps - deduct 1 mark for each missing or incorrect step - maximum 10 marks*"
        - i provide a correct diagram **and note alternative solutions**
    - "*it is possible that the **student interprets** the question to mean XYZ instead - **i would accept this answer***"
    - "*if the student answers XYZ **without a full explanation** - 1 mark of 3*"
- Committee and external examiner review all exams, grading schemes, and grades

Therefore…

- do not leave any parts completely blank
    - chance your arm - guess
    - we are trying to give you grades
        - give us some random assertions
        - start with a few keywords
- don't spend too long on working questions
    - do all the easy/quick questions first
- don't worry about exact formatting of code - semi-colons and so on

Prep Suggestion

- Make a table of all the algorithms and data-structures we looked at
- Advantages and disadvantages of each
- Draw a diagram of each
- Can you compare them against each other?
- Can you talk about Big O for several of them?

Prep Suggestion

- Have another look at your assignments
  - Can you write a few functions on paper?
  - Try explaining the merits of some algorithms to a family member!
  - What types of complexity are there to consider, and how can we analyse that?

My Thoughts on Exams

- Really only test how good you are at doing exams and cramming
- Interview questions are often just Algorithms course exam questions (keep the questions paper?)
- Not a great assessment of practical ability
- If you know you're better but panic during exams
  - don't worry! - nobody cares about my exam grades
  - our CA grade is nice and high

More Important

- Your understanding and ability
    - **keep practising** your programming, reading, writing, and talking
- How you communicate (and how often)
    - Writing quality; reports, papers, teaching, blogs
    - Speaking; talks, interviews, meetings, presenting, teaching
- *Perception* of your
    - professionalism (lots of things)
    - positive impact on team or community

Practice Exam Questions

- I have a pool of easy/working/deeper questions
  - -> main exam in April
  - -> supplementary and ~second sup exams
    - -> i'll upload a sample script for you guys too
- There is a similar exam from the Algorithms CS course - I uploaded this to Blackboard
- Exam questions closely based on those from other colleges' Algorithms courses

2. (a) Arrange the following Big-O approximations in order of complexity from lowest to highest; $O(1)$, $O(n!)$, $O(n^2)$, $O(log(n))$ [2 marks]

   (b) State three different methods for estimating the time complexity of a program. Give an example of each. [3 marks]

   (c) Estimate the best and worst Big-O time complexity of the following C code. Also estimate the auxiliary space complexity. Explain your reasoning.

```
void process_array( int* array, int array_length ) {
        for ( int i = 0; i < array_length; i++ ) {
                int first_val = array[i];
                for ( int j = 0; j < array_length; j++ ) {
                        if ( array[j] > first_val ) {
                                array[i] = array[j];
                                array[j] = first_val;
                                first_val = array[i];
                                break;
                        }
                }
        }
        int k = 0;
        while ( k < 4 ) {
                int tmp = array[0];
                for ( int i = 0; i < array_length - 1; i++ ) {
                        array[i] = array[i + 1];
                }
                array[array_length - 1] = tmp;
                k++;
        }
}
```

[10 marks]

   (d) With reference to time and space complexities, and other points of interest, compare merits and costs of the following algorithms. Use appropriate terms and Big-O notation to classify them. A table may help your explanation: *insertion sort*, *quicksort*, *merge sort*, *bubble sort*. [10 marks]