Module 12: In conclusion

T. Biedl    M. Petrick    O. Veksler
Based on lecture notes by many previous cs240 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2019

References: None
Outline

- Final
- What was the course about
- Comments?
Outline

- Final
  - What was the course about
  - Comments?
Final exam information

- Date, time, rooms → see web page.
- Look up your seat!
- Final help session? → see piazza
- Material covered: everything
  (exceptions, if any, will be posted on web page and piazza)
- Reference-sheet will be provided and published beforehand.
- Strong emphasis on second half
- Types of question: similar to midterm → see blank midterm
Outline

- Final
- What was the course about
- Comments?
Course summary: what was this about?

1. How to re-organize data
   - (Mergesort), Heapsort, Quicksort, count sort, radix sort
   - ADT Priority Queue, finding maximum, Selection
   - Lower Bounds for a problem, decision trees

2. How to manipulate structured data (key-value-pairs)
   - Balanced trees, hashing, tries
   - Special keys: words, integers, points
   - Special situations: biased search-requests, range-queries

3. How to manipulate unstructured data (text)
   - Searching
   - Compression

4. Some general-purpose techniques
   - Randomization: Shift average-requirements from instances to luck
   - Pre-processing: Initial work pays off in faster queries later
   - External-memory: Huge data warrants different thinking
Future courses that are related

**Required:** CS341, Algorithms (FWS)
- Focus on problem solving, especially for graphs
- Lots more lower bounds, especially NP-hardness.

**Optional:**
- (CS341 prereq) CS466, Algorithm Design & Analysis (FS)
  - Amortized analysis, randomized algorithms, online algorithms, ...
- CS348/448: Databases (FWS, F)
  - More complicated queries than search and range
  - Big data → external memory becomes important
- CS482: Biological Sequence Analysis (W)
  - Lots more on pattern matching, especially with suffix trees

Maybe a graduate course?
- CS762: Graph-theoretic Algorithms (taught rarely, S20?, T. Biedl)
  - Algorithms for special types of graphs. CS341 prereq.
  - Grad course; talk to CS advisor if you plan to take it
Future courses that are related

**Required:** CS341, Algorithms (FWS)
- Focus on problem solving, especially for graphs
- Lots more lower bounds, especially NP-hardness.

**Optional:** (CS341 prereq)
- CS466, Algorithm Design & Analysis (FS)
  - Amortized analysis, randomized algorithms, online algorithms, ....
- CS348/448: Databases (FWS,F)
  - More complicated queries than search and range
  - Big data → external memory becomes important
- CS482: Biological Sequence Analysis (W)
  - Lots more on pattern matching, especially with suffix trees
Future courses that are related

**Required:** CS341, Algorithms (FWS)
- Focus on problem solving, especially for graphs
- Lots more lower bounds, especially NP-hardness.

**Optional:** (CS341 prereq)
- CS466, Algorithm Design & Analysis (FS)
  - Amortized analysis, randomized algorithms, online algorithms, ....
- CS348/448: Databases (FWS,F)
  - More complicated queries than search and range
  - Big data → external memory becomes important
- CS482: Biological Sequence Analysis (W)
  - Lots more on pattern matching, especially with suffix trees

**Maybe a graduate course?**
- CS762: Graph-theoretic Algorithms (taught rarely, S20?, T. Biedl)
  - Algorithms for special types of graphs. CS341 prereq.
  - Grad course; talk to CS advisor if you plan to take it
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
- To save on implementation/experimentation-cost, analyze algorithms on paper first to eliminate obviously bad solutions.
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
- To save on implementation/experimentation-cost, analyze algorithms on paper first to eliminate obviously bad solutions.
- There isn’t always one best solution—the answer to “which algo is best?” is almost always “it depends”. Know your input.
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
- To save on implementation/experimentation-cost, analyze algorithms on paper first to eliminate obviously bad solutions.
- There isn’t always one best solution—the answer to “which algo is best?” is almost always “it depends”. Know your input.
- Don’t be content with a hack. Convince yourself that it works and that it’s reasonably fast.

(Based on a quote by J. Malazita)
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
- To save on implementation/experimentation-cost, analyze algorithms on paper first to eliminate obviously bad solutions.
- There isn’t always one best solution—the answer to “which algo is best?” is almost always “it depends”. Know your input.
- Don’t be content with a hack. Convince yourself that it works and that it’s reasonably fast.
- Don’t be content even though it works. Can you be faster and/or more space-efficient?
What to remember after the final...

- Most problems have many possible solutions. Don’t implement the first one you can think of.
- To save on implementation/experimentation-cost, analyze algorithms on paper first to eliminate obviously bad solutions.
- There isn’t always one best solution—the answer to “which algo is best?” is almost always “it depends”. Know your input.
- Don’t be content with a hack. Convince yourself that it works and that it’s reasonably fast.
- Don’t be content even though it works. Can you be faster and/or more space-efficient?

This wasn’t training for your first job (programmer). It was training for your second job (code designer/manager of programmers).

(Based on a quote by J. Malazita)
Outline

- Final
- What was the course about
- Comments?
Any comments?

- If you are registered in this section: evaluate at
  
  https://evaluate.uwaterloo.ca

- For any other comments (e.g. on other sections):
  email the Director of Undergraduate Studies
  
  currently Prof. Brown, dan.brown@uwaterloo.ca
Any comments?

- If you are registered in this section: evaluate at https://evaluate.uwaterloo.ca

- For any other comments (e.g. on other sections): email the Director of Undergraduate Studies currently Prof. Brown, dan.brown@uwaterloo.ca