CS 240 – Data Structures and Data Management

Module 12: In conclusion

A. Storjohann

Based on lecture notes by many previous cs240 instructors

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

Fall 2018

References: None

version 2018-11-29 11:55

Outline

- Final
- What was the course about

Outline

- Final
- What was the course about

Final exam information

- Date, time, rooms $\rightarrow$ see web page.
- Look up your seat!
- Final help session $\rightarrow$ see piazza
- Material covered: everything except external sorting and extendible hashing
  see webpage/piazza
- The exam includes a help-sheet that will be published beforehand.
- Strong emphasis on second half
- Types of question: similar to midterm
- Some short-answer questions
  - Asymptotics: give Θ bound for indicated operation
  - No justification
- $>$ 50% of marks for purely mechanical work

Outline

- Final
- What was the course about

Course summary: what was this about?

1. How to sort data
   - (Mergesort), Heapsort, Quicksort, count sort, radix sort
   - ADT Priority Queue
   - 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
   - Average-case might explain why it's good in practice
   - Randomization used to enforce average-case
   - 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.

Future courses that are related

Required: CS341, Algorithms (FWS)

- Focus on problem solving, especially for graphs
- Lots more lower bounds, especially NP-hardness.

Optional:

- 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 $\rightarrow$ 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:

- 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 $\rightarrow$ external memory becomes important
- CS482: Biological Sequence Analysis (W)
  - Lots more on pattern matching, especially with suffix trees

Maybe a graduate course?

- CS487/687: Introduction to Computer Algebra (W19, Eric Schost)
  - Exact computation with polynomials, matrices, solving linear equations, systems of polynomials

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.

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)