“Notes” slides from before lecture

CSE 21, Winter 2017, Section A00

Lecture 1 Notes

Class URL: http://vlsicad.ucsd.edu/courses/cse21-w17/
• Schedule (with readings) on class website

• HW1 will be posted by tomorrow (due Tuesday, January 17)
  – Groups of up to three permitted (cross-section is okay). Understand Instructions and Academic Integrity rules before you start.

• Handouts (induction, invariants, proof techniques) are posted – please take a look!

• A “pre-test” is online (Google Form, for your own self-calibration, no impact on grade, anonymous if you wish)

• Lecture slides will typically be posted before lecture at the class website and on Piazza (this week’s slides are posted)
  – Annotated slides will typically be posted within a day of lecture

• Sections A00, B00 will keep in synch as best possible – shared TAs and tutors, Piazza, lecture slides, homeworks

• Today: Introduction, Sorting
• Misc questions
  – Does anyone really, really need TA OH’s on Saturday? Current coverage is 4-6-5-5-4-2 hours on each of Sunday-Friday.
  – (Midterms are on Mondays. Five homeworks are due on Tuesdays, and three homeworks are due on Thursdays.)

• You must attend the Section A00 discussion (Thursday evening) unless you have a class conflict. See Piazza @9.

Me: abk @ cs.ucsd.edu

http://vlsicad.ucsd.edu/~abk
Welcome to CSE21!

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Instructor</th>
<th>Days</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>A00</td>
<td>Andrew Kahng</td>
<td>MW</td>
<td>6:30pm – 7:50pm</td>
<td>Center 212</td>
</tr>
<tr>
<td>B00</td>
<td>Miles Jones</td>
<td>MWF</td>
<td>1:00pm – 1:50pm</td>
<td>Center 214</td>
</tr>
</tbody>
</table>

[http://vlsicad.ucsd.edu/courses/cse21-w17](http://vlsicad.ucsd.edu/courses/cse21-w17)

January 9, 2017
About this course

Formulate & solve problems ✓

Describe data ✓

Analyze algorithms ✓

Using math
About this course

Why is math part of the CS curriculum?

Proofs: key to convincing arguments, but also key part of software engineering

Vocabulary: basic language of Computer Science

Quantitative Analysis: are our solutions / programs / algorithms good enough? How much computational resources (time, memory, power) does our solution use?
Introductions
What do we assume you know?

Rosen Chapters 1, 2, some of 5, some of 9.

More precisely: You can describe algorithms and their correctness using precise mathematical terminology and techniques. For example:

- Sets, relations (equivalence relations, orders)
- Logical equivalence, conditionals, hypotheses, conditionals, contrapositives
- Universal and existential quantifiers
- Proof by contradiction (indirect proof)
- Proof by induction
- Algorithm invariants
Logistics, part 1

Textbook: Rosen 7th Edition

Participation: Discussion (quizzes)

Exams:  
First Midterm: Monday, February 8 (Week 5)  
Second Midterm: Monday, February 27 (Week 8)  
Final Exam:  
A00: Wednesday, March 22 (7-10pm)  
B00: Friday, March 24 (7-10pm)  
(11:30-2:30)
Logistics, part 2

Friday 7pm CSB 002 = “problem session”

Class Website: [http://vlsicad.ucsd.edu/courses/cse21-w17](http://vlsicad.ucsd.edu/courses/cse21-w17)
Homework assignments, calendar, announcements, study guides, contact info, lecture slides (avail. Day after lecture.)

Gradescope: [gradescope.com](http://gradescope.com)
Homework submission and exam return. Entry code: MW5R4M

Piazza: Announcements and Q&A. Contact instructors here! No HW questions online.

Office hours: Instructors, TAs, reader-tutors. Weekdays and weekends:
Discuss HW questions here, in person!

(\*Currently, not Saturday)
Logistics, part 3

Exams (60%), HW (35%), Participation (5%)

Details on class website

- **Exams:** Can use handwritten note sheet. Drop lower MT score if do better on final.
- **Homeworks:** 8 HWs; drop lowest HW score.
- **Participation:** Credit if you attend and take a quiz at 7 discussions (your section). (Classroom and Piazza participation can potentially help as well.)

HW and exams answers evaluated not only on the correctness of your answers, but also on your ability to effectively communicate your ideas and convince the reader of your conclusions through proofs and logical reasoning.

Questions?
You’re working on a homework question and run across a definition you don’t understand. You Google the term and the first hit includes a full solution to the homework question. You avoid reading the solution and close the browser. You keep working on the solution and hand in the assignment, without mentioning the Google search since you didn’t use the result. Is this acceptable?

A. Yes  B. No
Academic Integrity Scenarios

You’re not sure if you are interpreting a homework problem correctly. You write a post on Piazza showing what you did to answer it, and asking if this is the correct way of interpreting the question. Is this acceptable?

A. Yes  B. No
You form a study group with two friends and start working on the next homework. Since there are 6 questions you each pick two questions, think about them, and write out your solutions in a shared Google doc. You glance over each other's work before turning in the assignment. Is this acceptable?

A. Yes  B. No
1. **Learn concepts** which computer science relies upon:

- Algorithms
- Asymptotic notation: $O()$, $\Omega()$, $\Theta()$
- Recurrence relations: $T(n) = 2T(n/2) + 1$
- Graphs
- Enumeration and data representation
- Probability
An example of CS vocabulary: Trees

Data structure: Binary search trees

Stay tuned: Chapter 11 in Rosen, Week 6
Algorithm: parsing

```
program
  int id(main) ( ) { code }
  instruction
    id(cout) ...
    instruction
    return 0 ;
```
An example of CS vocabulary: Trees

**Model:** possible paths of computation

- NFA?
- ("Quantum computing"?)

(Pointed out after class: any association to "quantum computing" shouldn't really have been mentioned.)

Automaton

- Accepts

Example Inputs

- Non-determinism
An example of CS vocabulary: Trees

**Model:** Phylogenetic (evolutionary) tree
An example of CS vocabulary: Trees

**State space:** possible configurations of a game

See Lecture 19

(Grad School thoughts?)

CSE 101 W16
Lecture 19

(Will post a link)
An example of CS vocabulary: Trees

Conclusion: Many different applications but same underlying idea.

- How do we define a tree?
- What properties are guaranteed by this definition?
- What algorithms can exploit these properties?
2. Solve problems.

Come up with *new* algorithms

Think of the homework questions as puzzles that you need to unravel: the solution or even the approach won't be clear right away.

You can work on homework in groups of 1-3 students.
Sorting (or Ordering)

* Assume elements of the set to be sorted have some underlying order
Which of the following collections of elements is listed in sorted order?

A. 42, 10, 30, 25
B. 10, 25, 30, 40
C. 40, 30, 25, 10
D. All of the above
E. None of the above

- A. Lexicographic (of written version)
- B. Increasing
- C. Decreasing

D. All of the above
Why sort?

A TA facing a stack of exams needs to input all 400 scores into a spreadsheet where the students are listed in alphabetical order.

OR

You want to find all the duplicate values in a long list.
Why sort?

A TA facing a stack of exams needs to input all 400 scores into a spreadsheet where the students are listed in alphabetical order.

OR

You want to find all the duplicate values in a long list.

It's easier to access data when it is sorted because you know exactly where to find it.
DIY: Sorting Algorithms

1. **Find a group** of about 20 people nearby. Write your first names on separate papers.

2. **Sort** the names of the people in your group alphabetically by first name.

3. **Discuss as a group** the strategy you used to sort the papers, and how you might describe it to someone else.

4. **Write** a clear English description of the strategy your group used (each person should do this).

5. **Select** one representative to describe your group’s strategy on the board. *(turn in one paper with your group’s first names and strategy on it)*
Discussion of Sorting Algorithms

• Strategies that groups used today:
  • Pass sorted stack of papers to the next person; maintain/restore sorted order (basically, insertion sort)
  • Divide according to first letter (or, A-M, N-Z etc.) (basically, bucket sort ~ radix sort-ish)
  • k-way (k = 3) mergesort
  • (Other: min-heap based (?) ; bubble sort (?); one group wrote down code for insertion sort)

Comment: When I passed out the yellow and orange sheets of paper, the holders of these sheets were essentially “cluster centers”. What is a reasonable (heuristic – see Slide 31) algorithm to construct clusters of students that respectively correspond to given “cluster centers”? What are aspects of a “high-quality” clustering solution? [Hints: “diameter”; “balance”; “separation”.] What algorithm would you suggest to find good locations of the cluster centers themselves?]
Discussion of Sorting Algorithms

- Is the strategy clear?
- Will the strategy always work?
- Does the strategy scale well to bigger groups?

\[ \text{e.g., insertion sort?} \]
\[ \text{bucket sort w/2 buckets?} \]
1) **What** problem are we solving?

2) **How** do we solve the problem?

3) **Why** do these steps solve the problem?

4) **When** do we get an answer?
### General questions to ask about algorithms

| 1) **What** problem are we solving? | **PROBLEM SPECIFICATION** |
| 2) **How** do we solve the problem? | **ALGORITHM DESCRIPTION** |
| 3) **Why** do these steps solve the problem? | **CORRECTNESS** |
| 4) **When** do we get an answer? | **RUNNING TIME PERFORMANCE** |

Comment: When discussing heuristic – maybe with Slide 30? – I mentioned the “nearest-neighbor” heuristic for the Traveling Salesperson Problem (“iteratively visit the closest unvisited city”). The term “heuristic” is sometimes used to characterize an algorithm that does not necessarily “work on all inputs”. For example, a heuristic might be optimal on some inputs but achieve poor solution quality (relative to optimal) on other inputs.
Given a list

\[ a_1, a_2, \ldots, a_n \]

rearrange the values so that

\[ a_1 \leq a_2 \leq \ldots \leq a_n \]

Values can be any type (with underlying total order). For simplicity, use integers.
Your approaches: HOW

- (TBD)
- (TBD)
- ...

(see earlier)
Your approaches: HOW

- Selection (min) sort $O(n^2)$
- Bubble sort $O(n^2)$
- Insertion sort $O(n^2)$
- Bucket sort $O(n)$
- Merge sort (recursive)
- Quick sort $\Theta(n \log n)$ to $\Omega(n^2)$
- Binary search tree traversal (heap-based; Heapsort)


Discussion section!
Selection Sort (Min Sort)

"Find the first name alphabetically, move it to the front. Then look for the next one, move it, etc."

Additional informal wording, in case helpful: (1) Linear search to find the smallest element out of the last k elements. (2) Swap this element with the element that is in the first slot out of the last k elements. (3) Repeat this n-1 times for k = n down to 2.
Selection Sort (MinSort) Pseudocode

Rosen page 203, exercises 41-42

\[ \text{procedure selection sort}(a_1, a_2, \ldots, a_n: \text{real numbers with } n \geq 2) \]
\[ \text{for } i := 1 \text{ to } n-1 \]
\[ \quad m := i \]
\[ \quad \text{for } j := i+1 \text{ to } n \]
\[ \quad \quad \text{if } (a_j < a_m) \text{ then } m := j \]
\[ \quad \text{interchange } a_i \text{ and } a_m \]
\{ a_1, \ldots, a_n \text{ is in increasing order} \}

\[ \text{[i}^{\text{th}} \text{ element, and (i+1)}^{\text{st}} \text{ to n}^{\text{th}} \text{ elements, comprise last (n-i+1) elements]} \]
Selection Sort (MinSort) Illustration

Comment: From python execution in class (should be on podcast).
Bubble Sort

"Compare the first two cards, and if the first is bigger, keep comparing it to the next card in the stack until we find one larger than it. Repeat until the stack is sorted."

Additional informal wording, in case helpful: (1) Swap consecutive elements so that the largest element “bubbles” up to the top. (2) Repeat this n – 1 times.
procedure bubble sort(a_1, a_2, ..., a_n: real numbers with n >= 2 )
for i := 1 to n-1
   for j := 1 to n-i
      if ( a_j > a_{j+1} ) then interchange a_j and a_{j+1}

{ a_1, ..., a_n is in increasing order}
Bubble Sort Illustration

Comment: From python execution in class (should be on podcast).

```python
hd1.ucsd.edu % python bsort.py
[3, 6, 1, 4, 2] <-- input list
[3, 6, 1, 4, 2]
[3, 1, 6, 4, 2]
[3, 1, 4, 6, 2]
[3, 1, 4, 2, 6]
[1, 3, 4, 2, 6]
[1, 3, 4, 2, 6]
[1, 3, 2, 4, 6]
[1, 3, 2, 4, 6]
[1, 2, 3, 4, 6]
[1, 2, 3, 4, 6]
hd1.ucsd.edu %
```

- ✓ = comparison, potential swap
- ○ = i\text{th} largest element has bubbled up to its position (in sorted order)
Insertion Sort

"We passed the cards from right to left, each individual inserting their own card in the correct position as they relayed the pile."

Additional informal wording, in case helpful: (1) Take an element of A and find where it belongs relative to the elements before it. (2) Shift other elements (rightward) to make room, and put the element in its proper place. (3) Now that this element has been inserted where it belongs, do the same for the next element of A.
procedure insertion sort(a_1, a_2, ..., a_n: real numbers with n >=2 )
for j := 2 to n ]
  i := 1
  while a_j > a_i 
    i := i+1
  m := a_j
  for k := 0 to j-i-1
    a_{j-k} := a_{j-k-1}
    a_{i} := m

{ a_1, ..., a_n is in increasing order}
Insertion Sort Illustration

Comment: From python execution in class (should be on podcast).

```
hd1.ucsd.edu % python isort.py
[4, 2, 1, 5, 7, 2, 4, 5, 6] <-- input list
[2, 4, 1, 5, 7, 2, 4, 5, 6]
[1, 2, 4, 5, 7, 2, 4, 5, 6]
[1, 2, 4, 5, 7, 2, 4, 5, 6]
[1, 2, 4, 5, 7, 2, 4, 5, 6]
[1, 2, 2, 4, 5, 7, 4, 5, 6]
[1, 2, 2, 4, 4, 5, 7, 5, 6]
[1, 2, 2, 4, 4, 5, 5, 7, 6]
[1, 2, 2, 4, 4, 5, 5, 6, 7]
hd1.ucsd.edu %
```

- ○ = element at position $j$
- --- = positions with elements (guaranteed to be) in sorted order
"Call out from A to Z, collecting cards by first letter. If there are more than one with the same first letter, repeat with the second letter, and so on."
Bucket Sort – Pseudo pseudo code

• Create empty buckets that have an ordering.
• Put each of the elements of the list into the correct bucket.
• Sort within each bucket.
• Concatenate the buckets in order.
"We split into two groups and organized each of the groups, then got back together and figured out how to interleave the groups in order."
Merge Sort – Pseudo pseudo code

- If the list has just one element, return.
- Otherwise,
  - Divide list into two pieces:
    \[ L_1 = a_1 \ldots a_{n/2} \quad \text{and} \quad L_2 = a_{n/2+1} \ldots a_n \]
  - \( M_1 = \text{Merge sort} \left(L_1\right) \)
  - \( M_2 = \text{Merge sort} \left(L_2\right) \)
  - Merge the two (sorted) lists \( M_1 \) and \( M_2 \)

Rosen page 196, 367-370
Others?

Bogo sort

Quick sort

Binary search tree traversal

Why so many algorithms?
Why so many algorithms?

Practice for homework / exam / job interviews.

Some algorithms are better than others. Wait, better?

- model of computation
- per-op costs
- key vs. data size
- range of data

- input “nearly” sorted?
- size of input
- runtime / speed
- storage / memory
- # LOC?
- complexity
End of Day 1 (MWF schedule)
Reminders

- Read syllabus on class website.
- Make sure that you are enrolled in Piazza and Gradescope.
- Make sure to plan to attend your discussion section!!!
- **Note:** Problem Session A02, Fridays 7-8pm CSB002, will be used for extra “problem-solving” and/or review and tutorial sessions (solving recurrences, induction proofs, invariants, etc.)

Homework 1 is due in Gradescope next **Tuesday (January 17) at 11:59pm.**

In general, see Piazza for a list of things you should do right away.
Note #1: Basic Needs Resources

• Are you eating properly? Do you have adequate access to nutritious food? Do you have stable housing? Are you homeless or couch surfing?

• If you or someone you know is suffering from food and/or housing insecurities, please note:
  • The Triton Food Pantry (in the old Student Center), https://www.facebook.com/tritonfoodpantry/, is free and anonymous, and includes produce.
  • Financial aid resources, the possibility of emergency grant funding, and off-campus housing referral resources are available.
  • CAPS and college deans can connect students to the above resources, as well as other community resources and support.
The Office for the Prevention of Harassment & Discrimination (OPHD) provides assistance to students, faculty, and staff regarding reports of bias, harassment, and discrimination. OPHD is the UC San Diego Title IX office. Title IX of the Education Amendments of 1972 is the federal law that prohibits sex discrimination in educational institutions that are recipients of federal funds. Jacobs School students have the right to an educational environment that is free from harassment and discrimination.

Students have options for reporting incidents of sexual violence and sexual harassment. Sexual violence includes sexual assault, dating violence, domestic violence, and stalking. Information about reporting options may be obtained at OPHD at (858) 534-8298, ophd@ucsd.edu or http://ophd.ucsd.edu. Students may receive confidential assistance at CARE at the Sexual Assault Resource Center at (858) 534-5793, sarc@ucsd.edu or http://care.ucsd.edu or Counseling and Psychological Services (CAPS) at (858) 534-3755 or http://caps.ucsd.edu.

Students may feel more comfortable discussing their particular concern with a trusted employee. This may be a Jacobs School student affairs staff member, a department Chair, a faculty member or other University official. These individuals have an obligation to report incidents of sexual violence and sexual harassment to OPHD. This does not necessarily mean that a formal complaint will be filed.

If you find yourself in an uncomfortable situation, ask for help. The Jacobs School is committed to upholding University policies regarding nondiscrimination, sexual violence and sexual harassment.
Start of Day 2 (not repeated from Day 1) (MWF schedule)
From "How" to "Why"

What makes this algorithm work?

How do you know that the resulting list will be sorted?

*For loop-based algorithms:*

What’s the effect of each loop iteration on the list?

Have we made progress?

(We’ll start here on Wednesday. Please review slides through end of Day 3!)