WELCOME TO CSE 100!

Advanced Data Structures in C++
Instructors

• Diba Mirza (dimirza@eng.ucsd.edu)
  • PhD (Computer Engineering, UCSD)
  • Second year @ UCSD as teaching faculty

• Office: 2124 EBU3B

• Office hours:
  • Mon: 3pm – 4pm, Wed: 3pm to 4pm
  • Or by appointment
Instructors

- Debasish Sahoo (dsahoo@ucsd.edu)
  - PhD (Electrical Engineering, Stanford University)
  - Assistant Professor (Department of Pediatrics)
- Office: BRF II – 2119 (Biomedical Research Facility II)
- Office hours:
  - Wed: 2pm – 3pm, Fri: 2pm to 3pm
  - Or by appointment
Information about TAs and tutors

TAs:

Vidya Kirupanidhi
Vineel Konduru

Tutors:
Albert Chang
Alvin See
Avnish Patel
Daniel Lee
Elaine Hwang
Eswar Dhinakaran
Ethan Chan
Jorel Briones
Joshua Tang
Kyle Buzsaki
Kyle Huynh
Kyle Barron-Kraus
Nan Shu
Richard Lin
Jacob Tao
Patrick Chung
Sanjana Agarwal

Office/lab hours available on the course website

http://cseweb.ucsd.edu/classes/sp15/cse100-ab/
CLICKERS OUT

Set your frequency to AA
Have you been in a class that used peer instruction before?

A. Yes
B. No
C. I’m not sure
Peer instruction groups

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
</tbody>
</table>

Take a minute to introduce yourself to your group members
iClickers: You must bring them

- Buy an iClicker at the Bookstore
- Register it on TED by April 4 (Saturday).

Assigned Reading from

- Open Data Structures (in C++). Pat Morin. (Free)

Text on reserve

- C++ for Java Programmers, Mark Allen Weiss
About This Class

You must **attend** class
You must **prepare** for class
You must **participate** in class
Course Logistics

• Less than 75% iClicker response ≡ missing a class
• You may miss up to 2 classes with no penalty
• Two midterms
• Grading
  • Reading quizzes: 5%
  • Participation points: 5% (lowest two dropped)
  • Programming Assignments: 35%
    PA0: 2%, PA1: 5%, PA2: 9%, PA3: 9%, PA4: 10%
  • Midterm exams: 30% (15% each)
  • Final exam: 25%
Assignment 0: Due today @ 9pm
Pair Programming guidelines

Basic rules

- All code written with two programmers at one machine
- You must plan *ahead of time* when you will get together
- You can change partner for each PA
- Don't be a jerk

Selecting partners: Factors to consider

- Schedule compatibility
- Roughly equal “eagerness”
- Roughly equal experience
- Partner from other section

You must choose your partner by MONDAY (tonight)
PA1 due in a week

- Discussion sections for week 1 are mandatory
  - GIT is covered only in discussion sections
  - C++ tutorial

- Get ready for serious C++ programming
  - Read zyBook section 1-8 for C++ introduction
  - Use Piazza to ask burning questions
    - Search posts before asking a new question
    - Google!
    - Refer to course website first
    - Read the PA writeups!
Reading Quizzes, due Wed and Fri before class, administered offline via TED

Discussion sections are mandatory in the first two weeks
Why study data structures?
Discuss with your group

1. Job interviews
2. Fast storage & retrieval of data (Big Data!)
3. Space efficient data representation
4. Good abstractions for operations on data
5. Hone your coding skills → visualize → code
What we will talk about

• Structures for fast storage and retrieval of different types of data (Trees, trees, and more trees! Oh, and Hashtables)

• Structures that rely on randomness for (expected) speed (skip lists)

• Structures to support compact data representation (tries)

• Structures to support topology (graphs)

• Structures that take advantage of memory and disk organization (B-Trees)

All that AND C++ and version control systems!
A lot of data (structures) but what should you take away?

- Ability to choose the correct data structure -> requires understanding data structures from a client perspective:
  - Operations supported
  - Time and space complexity (should be able to analyze)

- Ability to implement some number of non-trivial data structures, correctly, efficiently and elegantly.
  - You may never have to do it (the STL and all…) but if you ever need to, you’ll be glad you have the skill.
  - The ability to write and understand good, clean, well-designed C++ code.
If you only remember one thing today

http://cseweb.ucsd.edu/classes/sp15/cse100-ab/

Read the syllabus. Know what’s required. Know how to get help.
Which of the following is/are a tree?

A.  
B.  
C.  
D. A & B  
E. All of A-C
Which of the following is/are a tree?

A.

B.

C.

D. A & B
E. All of A-C
Binary Search Tree – What is it?

What are the numbers in the nodes?

This node may in general hold more data, only the key is shown. Say if node was an employee record, the data may be multiple fields:

- Employer name
- Age
- SSN
- Salary

Pick one to be the key (depends on approach)

The BST is organized based on the keys.
Binary Search Tree – What is it?

For any node,
- Keys in node’s left subtree \(\leq\) Node key
- Node key \(\leq\) Keys in node’s right subtree

Do the keys have to be integers?
Which of the following is/are a binary search tree?

A. 42
   32
   12

B. 42
   32
   12
   32

C. 42
   32
   65
   12

D. 42
   32
   56
   12
   45

E. More than one of these
Binary Search Trees

- What are the operations supported? If available as part of a standard library (say C++ STL) know how to use it.
- What are the running times of these operations? Later Lectures
- How do you implement the BST i.e. operations supported by it? PA 1
Binary Search Trees

• What is it good for?
  • If it satisfies a special property i.e. Balanced, you can think of it as a dynamic version of the sorted array

Supports all operations like sorted array + Fast Inserts & Deletes
PA1: Implementing BST operations in C++

- You need to implement
  - find() – find an element
  - size() – returns total number of elements
  - clear() – deletes all the elements
  - empty() – checks if the BST is empty
  - successor() – returns the next element in an in-order traversal
- And the iterator pattern (we will talk about it later this week)

Don't cheat!
Under the hood: Searching an element in the BST

To search for element with key k
1. Start at the root
2. If k=key(root), found key, stop.
3. Else If k< key(root), recursively search the left subtree: $T_L$
   Else recursively search the right subtree: $T_R$

Search for 41.
Now search for 43.
Under the hood: Finding the successor of an element in the BST

Which node is the successor of 56? How would you find it?
Traversing the BST in sorted order

Different methods of tree traversal:
- Pre order traversal
- Post order traversal
- In order traversal

successor() – returns the next element in an in-order traversal
In-order traversal of BST

Which of the following results from an in-order traversal of a BST?
A. Nodes are visited in the order in which they were inserted into the BST
B. Nodes are visited in order of the number of children that they have
C. Nodes are visited in sorted order of their keys
D. None of the above
PA0

- Don’t forget to choose a partner and complete PA0 by tonight (Monday, March 30 @9 pm).
- We should have you setup on gitLab by the next day (Tuesday, April 1st @noon). Wait for the announcement.
- You can start working on PA1 soon after.
- PA1 is due in 1 week on Tues (April 7@11:59PM)