Welcome to Programming Practicum

“Putting the C into CS”

You aren’t here

Massey University
Palmerston North, NZ

installing Debian 3.1

writing clinic reports

rebooting knuth (or turing or…)

coding chunky strings

clinic liaison phone call

On the 405, in traffic, being chased by police (and TV) helicopters.

On fire just W of here!!

Traveling through time and space on the Tardis

Teaching Honors English for Janice Barbee at Pomona High School

Mailing something at the Claremont Post Office

University of St. Petersburg

On the 405, in traffic, being chased by police (and TV) helicopters.

On fire just W of here!!

Traveling through time and space on the Tardis

Teaching Honors English for Janice Barbee at Pomona High School

Mailing something at the Claremont Post Office

University of St. Petersburg
Introductions...!

Zach Dodds

Office  Olin B163
Email   dodds@cs.hmc.edu

fan of low-level AI

taker of low-quality photos
Starbucks triumph-er!

not afraid of stuffed animals!

and not good at selfies...
Introductions...

Zach Dodds
Olin B163
dodds@cs.hmc.edu

fan of low-level AI
taker of low-quality pictures
consumer of high-quantity coffee!

and not talented at selfies, either...
Introductions...

Zach Dodds
Olin B163
dodds@cs.hmc.edu

fan of low-level AI
taker of low-quality pictures
consumer of high-quantity coffee!

and not talented at selfies, either...
Winter break...

Zach Dodds
Olin B163
dodds@cs.hmc.edu

fan of low-level AI
taker of low-quality pictures
consumer of high-quantity coffee!
and Ethiopian cuisine...
Winter break...

Zach Dodds
Olin B163
dodds@cs.hmc.edu

fan of low-level AI
taker of low-quality pictures
consumer of high-quantity coffee!
and Ethiopian cuisine...
What is this course about?

practicing algorithmic/programming skills
~ 80-90%

problems part
the balance is up to you...

trying out technologies/projects of interest
~ 80-90%
ditto

projects part
guest talks on cutting-edge research
~ 42%
presentations part
trying out technologies/projects of interest after early November, if you'd like

Alums: What do you feel you didn't get @ HMC CS?
**trying out** technologies/projects of interest

**Alums:** What do you feel you *didn't get* @ HMC CS?

- **Paul Scott:**
  - *Don't try to teach web stuff*

- **Josh Klontz:**
  - *Cmake and build systems*

- **Josh Ehrlich:**
  - *Web things*

- **Karen Gragg:**
  - *Web technologies (just for terminology...)*

- **Moira Tagle:**
  - *Parallelizing/distributing large computations*

- **Will Scott:**
  - *Web frameworks*
Optional open-ended project

• worth up to +10 problems ~ also, an opportunity...

• ... to try out / get familiar with / learn about a technology, domain, library, or project that might be one of your answers to the question, What do you feel you didn't learn @ HMC?
Optional open-ended project

• worth up to +10 problems ~ also, an opportunity...

• ... to try out / get familiar with / learn about a
  technology, domain, library, or project

Plan:

(0) decide what you'd like to learn...

(1) find a reasonable resource for it...

(2) create a project and a write-up...

(3) time expectation: 3 hours per week

that might be one of your answers to the question,
What do you feel you didn't learn @ HMC?
Optional open-ended project

- worth up to +10 problems ~ also, an opportunity...
- ... to try out / get familiar with / learn about a technology, domain, library, or project

- framework, e.g., web framework (Django et al.)
- web basics: HTML/CSS/JavaScript/JQuery "Minecraft" technology?
- OpenCV Qt other UI library game dev. library console library?
- hardware project requiring programming

that might be one of your answers to the question, *What do you feel you didn't learn @ HMC?*
Drawbacks?

• specific technologies should be avoided in the CS curriculum

  I agree. Yet this one-unit course is too small to shift that balance...

• there's not enough support to make it work

  True! I'm no expert at what you're working on, but here the goal's not expertise, but the "working on" ...

• too much time is required...!

  3 good-faith hours per week + write-up == 12 problems
Drawbacks?

• specific technologies should be avoided in the CS curriculum
  I agree. Yet this one-unit course is too small to shift that balance...

• there's not enough support to make it work
  True! I'm no expert at what you're working on, but here the goal’s not expertise, but the "working on" ...

• too much time is required...!
  3 good-faith hours per week + write-up == 12 problems

Benefits?

• it never hurts to have an on-line portfolio of one or more of your projects...
  John Grasel, Cris Cecka

• curricular support vs. expertise support

• helps the limitations of DWIC letters, because it's unique + personalized
  "did well in class"

• sometimes the benefits don't outweigh the drawbacks
  one unit!
Interested? To do by **Feb. 3** ... 

**project proposal due**

**2-3 paragraphs:**

(0) Use the CS wiki (at least as a starting link)
(1) Describe your overall project idea(s)
(2) Describe your plan/resources
   • online tutorial or course?
   • do you have a "Hello, World!" version?
(3) Document your time-spent
(4) Check-in with me on Feb. 3 (or before...)

**First project deliverable/demo:**

Due **Tuesday, March 4.**
Project Grading

Projects are graded in units of "problems"...

You can earn up to **10 problems** for each project.

Here is the breakdown:

- **4 problems** good-faith 3 hrs/week
- **3 problems** weekly progress log, e.g., on CS wiki
- **3 problems** results! ~ a reasonable deliverable

...exceptional results are welcome & recognized!
Examples from last term....
What is this course about?

*practicing* algorithmic/programming skills

~ 80-90%

*problems* part

the balance is up to you...

*trying out* technologies/projects of interest

~ 80-90%

ditto

*projects* part

*guest talks* on cutting-edge research

~ 42%

*presentations* part
Problems!

practicing algorithmic/programming skills

Bessie!

Cows are the *global* theme of CS189's problems.
The **cowqueue** problem

**Input**

ABACB  
AABC

Cow label sequence #1 (s1)  
Cow label sequence #2 (s2)

**Output**

3

The number of the *longest common subsequence* bewteen s1 and s2.

---

In this case, the longest common subsequence is ABC or AAB though the problem doesn’t require knowing these.
LCS problem

\[ \text{LCS}(i_1, i_2) = \text{length of longest common subsequence of } s_1 \text{ up to } i_1 \text{ and } s_2 \text{ up to } i_2 \]

Strategy

1. Write a solution recursively.
2. Then, don't make any call more than once!
"LCS" problem

Length of longest common subsequence of s1 up to i1 and s2 up to i2

\[
\text{LCS}(i_1, i_2): \quad \text{if } s_1[i_1] = s_2[i_2]: \quad \text{return} \quad 1 + \text{LCS}(i_1-1, i_2-1)
\]

\[
\text{else: return} \quad \text{max}(\text{LCS}(i_1-1, i_2), \text{LCS}(i_1, i_2-1))
\]

Input:

\[
s_1 = "ABACB" \quad \text{Input} \quad s_2 = "AABC"
\]

Length of longest common subsequence of s1 up to i1 and s2 up to i2

\[
s_2 = "AABC" \quad s_1 = "ABACB"
\]
LCS code

```python
import sys
sys.setrecursionlimit(100000)

def LCS( i1, i2 ):
    """ classic LCS """
    if i1 < 0 or i2 < 0: return 0
    if s1[i1] == s2[i2]:
        return 1 + LCS(i1 - 1, i2 - 1)
    else:
        return max(LCS(i1 - 1, i2), LCS(i1, i2 - 1))

if __name__ == "__main__":
    s1 = raw_input(); L1 = len(s1)
    s2 = raw_input(); L2 = len(s2)
    result = LCS( L1-1, L2-1 )
    print result
```
practicing algorithmic/programming skills

What
Algorithm analysis and insight
Program design and implementation

Why
ACM programming contest
Hands-on practice with algorithms and techniques
Familiarizing with your choice of language/libraries "reasonable"
Research/prototype programming

Technical interview questions...

Unofficial course name: CS -70
part – but only *part* – of the motivation for CS 189:

ACM programming contest
# Scoreboard SoCal 2013 Contest

## Final Standings

<table>
<thead>
<tr>
<th></th>
<th>TEAM</th>
<th>Score</th>
<th>Penalties</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USC Trojans</td>
<td>6</td>
<td>1</td>
<td>1 (00:27.25)</td>
<td>1 (03:13.40)</td>
<td>0</td>
<td>1 (02:26.11)</td>
<td>1 (01:11.15)</td>
<td>2 (04:59.02 + 00:20.00)</td>
<td>1 (04:38.25)</td>
<td>1 (01:48.00)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>UCI Constructors</td>
<td>7</td>
<td>1</td>
<td>1 (00:43.35)</td>
<td>0</td>
<td>0</td>
<td>1 (01:37.45)</td>
<td>1 (00:58.42)</td>
<td>1 (03:32.50)</td>
<td>1 (03:09.14)</td>
<td>1 (02:44.40)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Caltech A</td>
<td>6</td>
<td>1</td>
<td>1 (00:58.25)</td>
<td>1</td>
<td>0</td>
<td>1 (00:50.10)</td>
<td>1 (01:39.57)</td>
<td>2 (03:51.22 + 00:20:00)</td>
<td>1 (01:11.33)</td>
<td>0</td>
<td>1</td>
<td>01:51:56</td>
</tr>
<tr>
<td>4</td>
<td>Caltech B</td>
<td>6</td>
<td>3</td>
<td>1 (00:25.21)</td>
<td>1</td>
<td>0</td>
<td>1 (02:28.28)</td>
<td>1 (01:34.11)</td>
<td>2 (01:38.15 + 00:20.00)</td>
<td>1</td>
<td>1 (02:09.00)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>CPSLO - The NULL Terminators</td>
<td>6</td>
<td>2</td>
<td>1 (02:11.34)</td>
<td>0</td>
<td>0</td>
<td>1 (00:54.31)</td>
<td>1 (01:02.29)</td>
<td>3 (04:27.51 + 00:40.00)</td>
<td>1 (04:43.26)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>UCLA Bruins</td>
<td>5</td>
<td>0</td>
<td>1 (02:06.22)</td>
<td>0</td>
<td>0</td>
<td>1 (01:08.30)</td>
<td>1 (00:37.04)</td>
<td>1 (02:38.43)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>USC Cardinal</td>
<td>5</td>
<td>2</td>
<td>1 (01:11.44)</td>
<td>0</td>
<td>0</td>
<td>1 (00:52.25)</td>
<td>1 (01:59.48)</td>
<td>3 (04:17.21 + 00:40.00)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>UCSD - Lead, Spin, Pull</td>
<td>5</td>
<td>1</td>
<td>1 (02:06.50)</td>
<td>0</td>
<td>0</td>
<td>1 (00:32.11)</td>
<td>1 (01:11.15)</td>
<td>0</td>
<td>2 (04:54.43 + 00:20:00)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>HMC Ednew</td>
<td>5</td>
<td>0</td>
<td>1 (02:47.54)</td>
<td>0</td>
<td>0</td>
<td>1 (01:07.56)</td>
<td>1 (00:50.05)</td>
<td>0</td>
<td>1</td>
<td>0 (03:21.52)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>HMC Hayden</td>
<td>5</td>
<td>2</td>
<td>1 (02:16.54)</td>
<td>0</td>
<td>0</td>
<td>1 (00:37.02)</td>
<td>1 (01:39.35)</td>
<td>3 (03:46.47 + 00:40.00)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>UCLA Gold</td>
<td>4</td>
<td>1</td>
<td>2 (00:30.31 + 00:20.00)</td>
<td>0</td>
<td>0</td>
<td>1 (01:36.45)</td>
<td>1 (02:06.20)</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Pomona 47</td>
<td>4</td>
<td>2</td>
<td>1 (01:06.45)</td>
<td>0</td>
<td>0</td>
<td>1 (00:52.40)</td>
<td>3 (02:51.40 + 00:40.00)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>UCSD - scissors</td>
<td>4</td>
<td>0</td>
<td>1 (02:41.37)</td>
<td>0</td>
<td>0</td>
<td>1 (01:22.45)</td>
<td>1 (01:44.15)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>SBCC One</td>
<td>4</td>
<td>1</td>
<td>1 (00:56.40)</td>
<td>0</td>
<td>0</td>
<td>1 (01:01.08)</td>
<td>1 (03:25.27)</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>UCSB - Alpha</td>
<td>4</td>
<td>1</td>
<td>1 (00:47.35)</td>
<td>0</td>
<td>0</td>
<td>1 (01:24.05)</td>
<td>1 (02:37.21)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Caltech C</td>
<td>4</td>
<td>2</td>
<td>1 (02:19.00)</td>
<td>2</td>
<td>0</td>
<td>1 (01:25.08)</td>
<td>1 (02:02.30)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>HMC Hammer</td>
<td>4</td>
<td>1</td>
<td>1 (01:38.39)</td>
<td>0</td>
<td>0</td>
<td>1 (02:16.10)</td>
<td>2</td>
<td>0 (03:26.45)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>USC Gold</td>
<td>4</td>
<td>2</td>
<td>1 (00:27.02)</td>
<td>0</td>
<td>0</td>
<td>1 (04:54.31)</td>
<td>1 (01:21.19)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
USC advanced to the finals in 2011, 2012, and 2013...
active watching!
active watching!
| Game Code | Minutes | Score  
|-----------|---------|--------|
| 01        | 0:01:05 | Remaining  

The contest will end on Sat Nov 7 2009 19:20:00.
Course webpage

A few references

Reference Links
- HMC ACM Page
- C++ & STL
- Java 1.6 API

Congratulations! to the HMC teams in the 2018 Southern California regionals. The standings out of 78 participating teams:

- 4th place -- HMC Hammer -- Ryan Brewster, Richard Porczak, and Jackson Newhouse
- 8th place -- HMC Squared -- Andrew Carter, Daniel Lubarov, and Kevin Black
- 10th place -- HMC 42 -- Emily Myers-Stanhope, Eric Aleshire, and Benson Khau
- 21st place -- HMC Escher -- Fiona Tay, Jacob Bandes-Storch, and Tum Chaturapruek

Problems and progress

<table>
<thead>
<tr>
<th>Name</th>
<th>problems</th>
<th>solved</th>
<th>forgot</th>
<th>covqueen</th>
<th>covlphset</th>
<th>covcheck</th>
<th>hfire</th>
<th>total</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dodds</td>
<td>Net</td>
<td>Yes</td>
<td>Net</td>
<td>Net</td>
<td>Net</td>
<td>Net</td>
<td>Net</td>
<td>Net</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Lecture Slides and Starting Code...

- Lecture 1, Fall 2012 materials (zip)

Problems you’ve solved

Total!

Slides, code, administrative info
Details

Problems are worth 150% if

- You solve them during the week they are assigned
- ... which extends to the start of the next ACM class

Language Choice?

Any reasonable language is OK; keep in mind that the ACM competition allows only Java, C, and C++.

Other "standard" languages for CS189 (so far):

C#, Python, Ruby, Perl, PHP, Haskell, Lua, Clojure, Lisp
Grading

CS 189 is graded by default ... (it's possible to take it P/F, too)
though not for CS elective credit...

Coding Guidelines

• problems can be done *any time* during the semester; projects have deadlines...
• discussion of algorithms always OK
• coding should be *within teams of 1-3*
• you may use any references *except* others' solutions or partial solutions...
• use `/cs/ACM/acmSubmit <file>` to submit on *knuth*

<table>
<thead>
<tr>
<th># Solved (out of 42)</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>43+</td>
<td>pretty much impossible!</td>
</tr>
<tr>
<td>28-42</td>
<td>A</td>
</tr>
<tr>
<td>23-27</td>
<td>A−</td>
</tr>
<tr>
<td>20-22</td>
<td>B+</td>
</tr>
<tr>
<td>17-19</td>
<td>B</td>
</tr>
<tr>
<td>14-16</td>
<td>B−</td>
</tr>
<tr>
<td>9-13</td>
<td>C range</td>
</tr>
<tr>
<td>≤ 9</td>
<td>&lt; D range or less</td>
</tr>
</tbody>
</table>
Max, Max, and Carl ~ dynamic programmers
Dynamic Programming

Many problems can be solved recursively...

... but with lots of repeated recursive calls!

These problems can be solved quickly with

(1) Memoization, or
(2) Dynamic programming

Idea: just don't repeat the repeated calls!
The **cowqueue** problem

**Input**

- ABACB
- AABC

Cow label sequence #1 (s1)
Cow label sequence #2 (s2)

**Output**

3

The number of the *longest common subsequence* between s1 and s2.

In this case, the longest common subsequence is ABC or AAB though the problem doesn't require knowing these.
**LCS problem**

\[
\text{s1} = \text{"ABACB"} \quad \text{Input} \quad \text{s2} = \text{"AABC"}
\]

\[
\text{LCS( } i_1, i_2 \text{ )} = \text{length of longest common subsequence of s1 up to } i_1 \text{ and s2 up to } i_2
\]

**Strategy**

1. Write a solution recursively.
2. Then, don't make any call more than once!
LCS problem

LCS( i1, i2 ):

if $s1[i1] == s2[i2]$: return 1 + LCS( i1-1, i2-1 )

if the same character, count it!

else: return max( LCS( i1-1, i2 ), LCS( i1, i2-1 ) )

otherwise, lose both ends and take the better result

Input:

$s1 = "ABACB"

$s2 = "AABC"

length of longest common subsequence of $s1$ up to $i1$ and $s2$ up to $i2$
LCS code

\[
s_1 = "ABACB" \quad \text{Input} \quad s_2 = "AABC"
\]

```python
import sys
sys.setrecursionlimit(100000)

def LCS(i1, i2):
    """ classic LCS """
    if i1 < 0 or i2 < 0: return 0
    if s1[i1] == s2[i2]:
        return 1 + LCS(i1 - 1, i2 - 1)
    else:
        return max(LCS(i1 - 1, i2), LCS(i1, i2 - 1))

if __name__ == "__main__":
    s1 = raw_input(); L1 = len(s1)
    s2 = raw_input(); L2 = len(s2)
    result = LCS(L1-1, L2-1)
    print result
```
**LCS idea**

\[
s_1 = "ABACB" \quad \text{Input} \quad s_2 = "AABC"
\]

<table>
<thead>
<tr>
<th>(s_1[i_1])</th>
<th>(\emptyset)</th>
<th>(A)</th>
<th>(AA)</th>
<th>(AAB)</th>
<th>(AABC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABACB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[LCS(4,3)\]
## LCS idea

### Input

<table>
<thead>
<tr>
<th></th>
<th>Φ</th>
<th>A</th>
<th>AA</th>
<th>AAB</th>
<th>AABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Φ</td>
<td><strong>Φ</strong></td>
<td><strong>A</strong></td>
<td><strong>AA</strong></td>
<td><strong>AAB</strong></td>
<td><strong>AABC</strong></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABAC</td>
<td></td>
<td></td>
<td></td>
<td>LCS(3,3)</td>
<td></td>
</tr>
<tr>
<td>ABACB</td>
<td></td>
<td></td>
<td>LCS(4,2)</td>
<td>LCS(4,3)</td>
<td></td>
</tr>
</tbody>
</table>

\[ s_1 = "ABACB" \quad \text{Input} \quad s_2 = "AABC" \]

\[ i_1 \quad i_2 \]
**LCS idea**

<table>
<thead>
<tr>
<th>s1 = &quot;ABACB&quot;</th>
<th>s2 = &quot;AABC&quot;</th>
</tr>
</thead>
</table>

```
\[
\begin{array}{cccccc}
\emptyset & \emptyset & A & AA & AAB & AABC \\
\hline
\emptyset & \emptyset & A & AA & AAB & AABC \\
AB & AB & AB & AAB & AABC \\
ABA & ABA & ABA & AABC \\
ABAC & ABAC & ABAC & AABC \\
ABACB & ABACB & ABACB & AABC \\
\end{array}
\]
```

**Input**

- \text{s1} = "ABACB"
- \text{s2} = "AABC"

- LCS(4,2) ← LCS(4,3)
- LCS(2,2) ← LCS(3,2)
- LCS(3,1) ← LCS(3,2)
- LCS(2,2) ← LCS(3,2)
### LCS idea

**Input**

- `s1 = "ABACB"`
- `s2 = "AABC"`

---

<table>
<thead>
<tr>
<th>string1 <code>s1[:i1]</code></th>
<th>∅</th>
<th>∅</th>
<th>A</th>
<th>AA</th>
<th>AAB</th>
<th>AABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABA</td>
<td></td>
<td></td>
<td></td>
<td>LCS(2,1)</td>
<td>LCS(2,2)</td>
<td></td>
</tr>
<tr>
<td>ABAC</td>
<td>LCS(3,0)</td>
<td>LCS(3,1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABACB</td>
<td>LCS(4,2)</td>
<td>LCS(4,3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

LCS(4,3) → LCS(3,3) → LCS(2,3) → LCS(1,3) → ∅
**LCS idea**

- **Input**
  - $s_1 = \text{"ABACB"}$
  - $s_2 = \text{"AABC"}$

---

**Table of LCS**

<table>
<thead>
<tr>
<th></th>
<th>∅</th>
<th>A</th>
<th>AA</th>
<th>AAB</th>
<th>AABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>LCS(-1,-1)</td>
<td>LCS(-1,0)</td>
<td>LCS(0,0)</td>
<td>LCS(0,1)</td>
<td>LCS(1,2)</td>
</tr>
<tr>
<td>A</td>
<td>LCS(1,-1)</td>
<td>LCS(1,0)</td>
<td>LCS(2,0)</td>
<td>LCS(2,1)</td>
<td>LCS(2,2)</td>
</tr>
<tr>
<td>AB</td>
<td>LCS(3,-1)</td>
<td>LCS(3,0)</td>
<td>LCS(3,1)</td>
<td>LCS(3,2)</td>
<td>LCS(3,3)</td>
</tr>
<tr>
<td>ABA</td>
<td>LCS(4,-1)</td>
<td>LCS(4,0)</td>
<td>LCS(4,1)</td>
<td>LCS(4,2)</td>
<td>LCS(4,3)</td>
</tr>
<tr>
<td>ABAC</td>
<td>LCS(5,-1)</td>
<td>LCS(5,0)</td>
<td>LCS(5,1)</td>
<td>LCS(5,2)</td>
<td>LCS(5,3)</td>
</tr>
<tr>
<td>ABACB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**LCS idea**

\[
\begin{array}{cccccc}
\emptyset & \emptyset & A & AA & AAB & AABC \\
\hline
\emptyset & LCS(-1,-1) & LCS(-1,0) & & & \\
A & LCS(-1,0) & LCS(0,0) & LCS(0,1) & & \\
AB & LCS(1,-1) & LCS(1,0) & LCS(1,1) & LCS(1,2) & \\
ABA & LCS(2,0) & LCS(2,1) & LCS(2,2) & & \\
ABAC & LCS(3,-1) & LCS(3,0) & LCS(3,1) & LCS(3,2) & \\
ABACB & LCS(4,-1) & LCS(4,0) & LCS(4,1) & LCS(4,2) & LCS(4,3) \\
\end{array}
\]

Collisions!
LCS, memoized

Put results in a dictionary. Look up instead of recomputing.

```python
# This is the "memoizing" dictionary of all distinct calls. Each distinct call is made only once and stored here.
D = {}

def LCS( i1, i2 ):
    """ classic LCS """
    if i1 < 0 or i2 < 0: return 0  # base cases
    if (i1, i2) in D: return D[ (i1,i2) ]  # already done!
    if s1[i1] == s2[i2]:
        result = 1 + LCS(i1-1, i2-1)
    else:
        result = max( LCS(i1-1, i2), LCS(i1, i2-1) )
    D[ (i1,i2) ] = result  # memo-ize it!
    return result  # before returning

if __name__ == "__main__":
    s1 = raw_input(); L1 = len(s1)
    s2 = raw_input(); L2 = len(s2)
    result = LCS( L1-1, L2-1 )
    print result
```
import sys; sys.setrecursionlimit(100000)

class memoize:
    def __init__(self, function):
        self.function = function
        self.memoized = {}  

    def __call__(self, *args):
        try:
            return self.memoized[args]
        except KeyError:
            self.memoized[args] = self.function(*args)
        return self.memoized[args]

@memoize
def LCS(i1, i2):
    # slow, recursive f'n here
LCS, DP'ed

Compute the table of results, bottom-up!

\[ s_1 = "ABACB" \quad \text{Input} \quad s_2 = "AABC" \]

<table>
<thead>
<tr>
<th>( \emptyset )</th>
<th>( \emptyset )</th>
<th>A</th>
<th>AA</th>
<th>AAB</th>
<th>AABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABACB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LCS, DP'ed

Compute the table of results, bottom-up!

s1 = "ABACB"  Input  s2 = "AABC"

string2 s2[:i2]

```
if __name__ == "__main__":
    s1 = raw_input(); L1 = len(s1)
    s2 = raw_input(); L2 = len(s2)
    DP = [ [0]*(L2+2) for i1 in range(L1+2) ]

    for i1 in range(L1):
        for i2 in range(L2):
            if s1[i1] == s2[i2]: DP[i1][i2] = 1 + DP[i1-1][i2-1]
            else: DP[i1][i2] = max( DP[i1][i2-1], DP[i1-1][i2] )

    result = DP[L1-1][L2-1]

    #for row in DP:
    #    print row

    print result
```
This week's problems

- Lecture 1, cowqueue code examples (zip)

New to CS189? Start with this problem!

Part of the challenge is deciding **which** problem to tackle...

Some of this week's problems have a "dynamic programming" theme...
What is this course about?

- **practicing** algorithmic/programming skills
  - ~ 80-90%

- **problems** part
  - The balance is up to you...

- **trying out** technologies/projects of interest
  - ~ 80-90%
  - Ditto

- **projects** part

- **guest talks** on cutting-edge research
  - ~ 42%

- **presentations** part
Guest lectures...

count as 1.5 problems

We have a guest lecture next week by Joshua Eckroth of Ohio State. Join in & sign in!
Jotto!

A word-guessing game similar to mastermind...

<table>
<thead>
<tr>
<th>Sophs</th>
<th>JRs</th>
<th>SRs</th>
<th>POM-CMC-SCR-PTZ</th>
<th>other</th>
</tr>
</thead>
</table>

This term's first class to guess another's word earns 1 problem...
This term's last class to have its word guessed earns 1 problem...