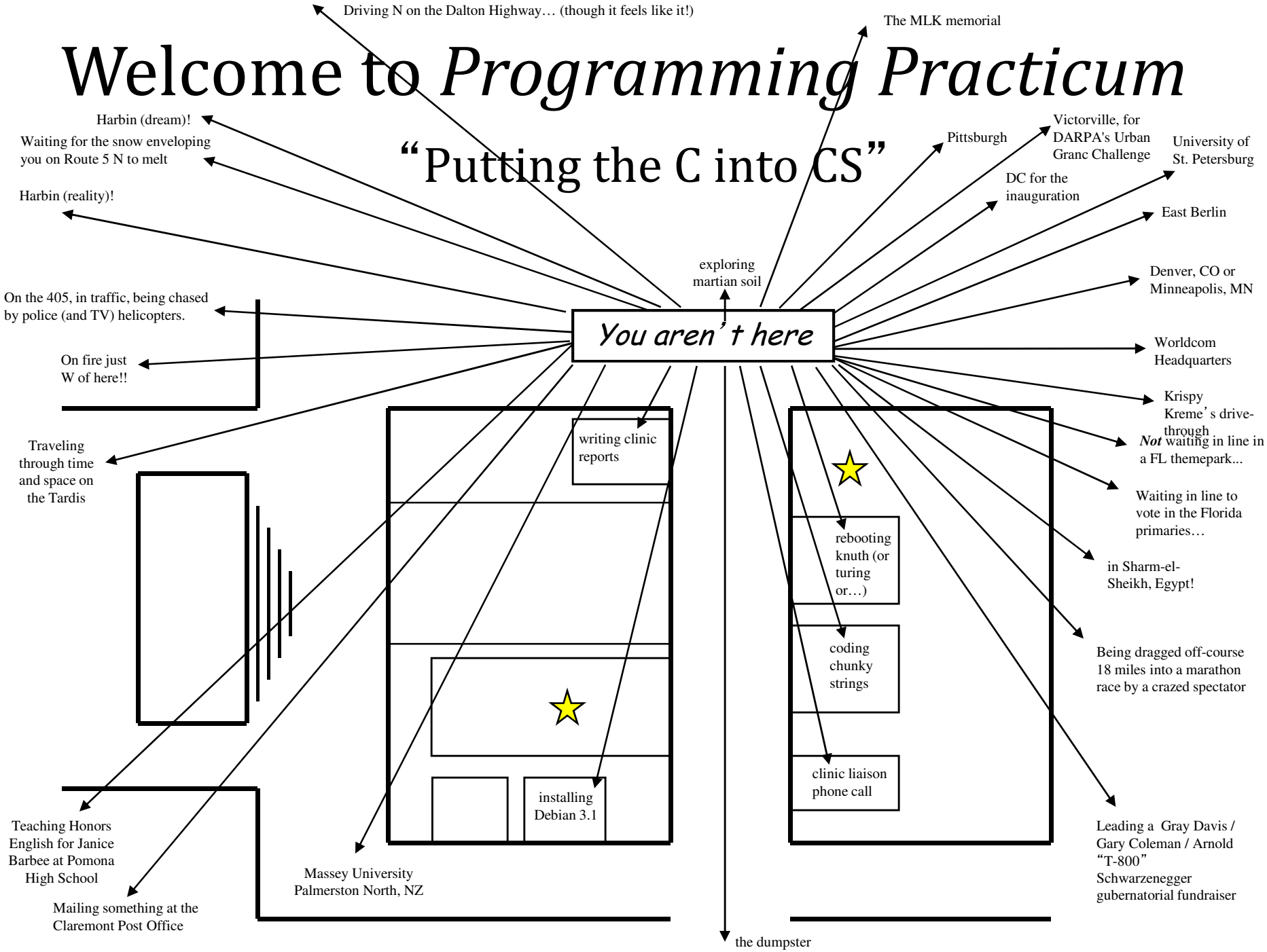
Welcome to *Programming Practicum*

"Putting the C into CS"



Introductions...!

Zach Dodds

Office **Olin B163**

Email **dodds@cs.hmc.edu**

not afraid of stuffed animals!

fan of *low-level* AI

taker of *low-quality* photos

Starbucks triumph-er!

and not good at selfies…

Introductions...

Image size:
246 × 293

No other sizes of this image found.

Visually similar images - Report images

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...

Image size:
246 × 293

No other sizes of this image found.

Visually similar images - Report images

Zach Dodds
Olin B163
dodds@cs.hmc.edu

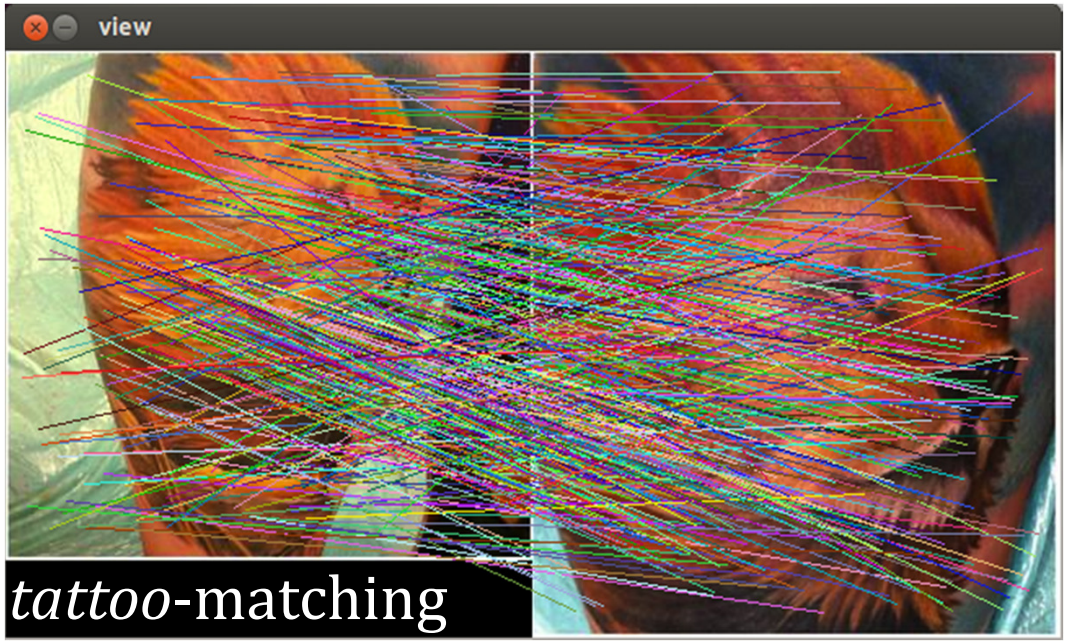
fan of low-level AI

taker of low-quality pictures

consumer of high-quantity coffee!

and not talented at selfies, either…

Winter break...



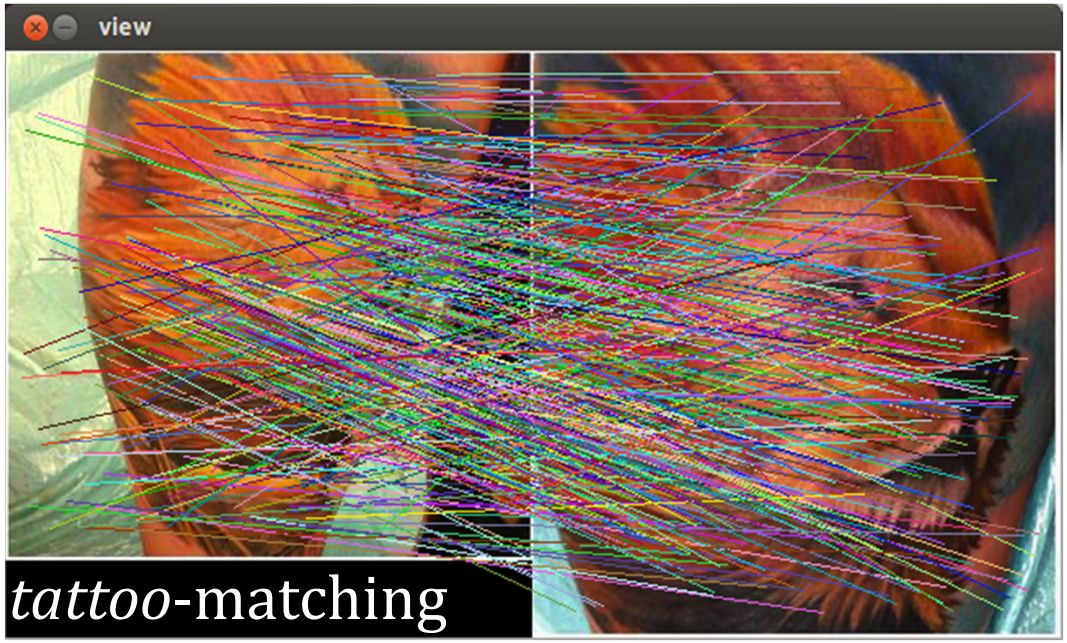
Zach Dodds
Olin B163
dodds@cs.hmc.edu

fan of low-level AI

taker of low-quality pictures

consumer of high-quantity coffee!
and Ethiopian cuisine...

Winter break...

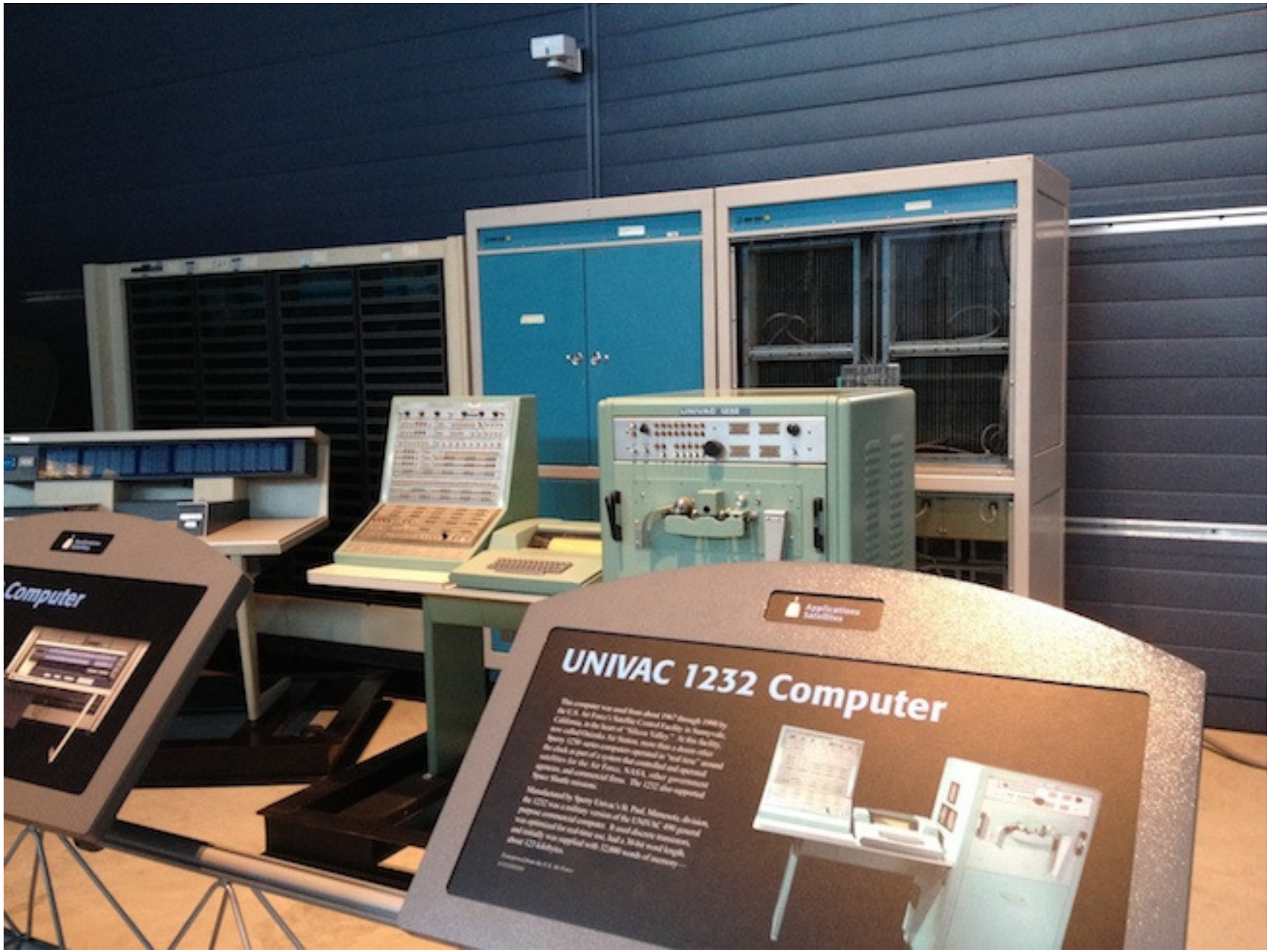


Zach Dodds
Olin B163
dodds@cs.hmc.edu

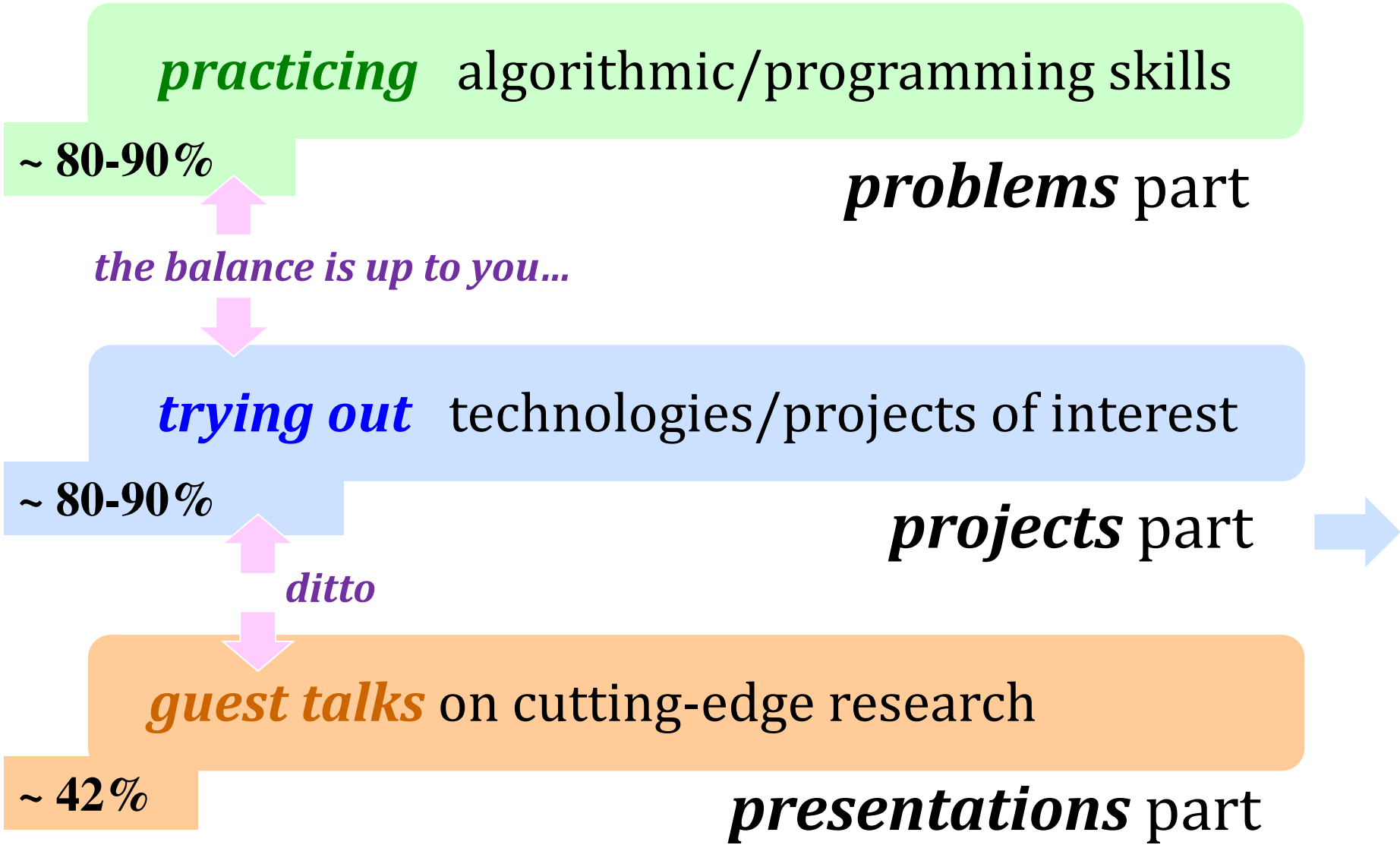
fan of low-level AI

taker of low-quality pictures

consumer of high-quantity coffee!
and Ethiopian cuisine...


Computer
UNIVAC 1232 Computer

What is this course about?

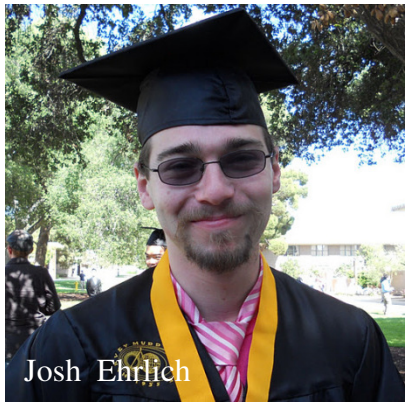


trying out technologies/projects of interest

after early November, *if you'd like*

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


Paul Scott


Josh Ehrlich


Moira Tagle


Josh Klontz


Karen Gragg


Will Scott

trying out technologies/projects of interest

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

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…

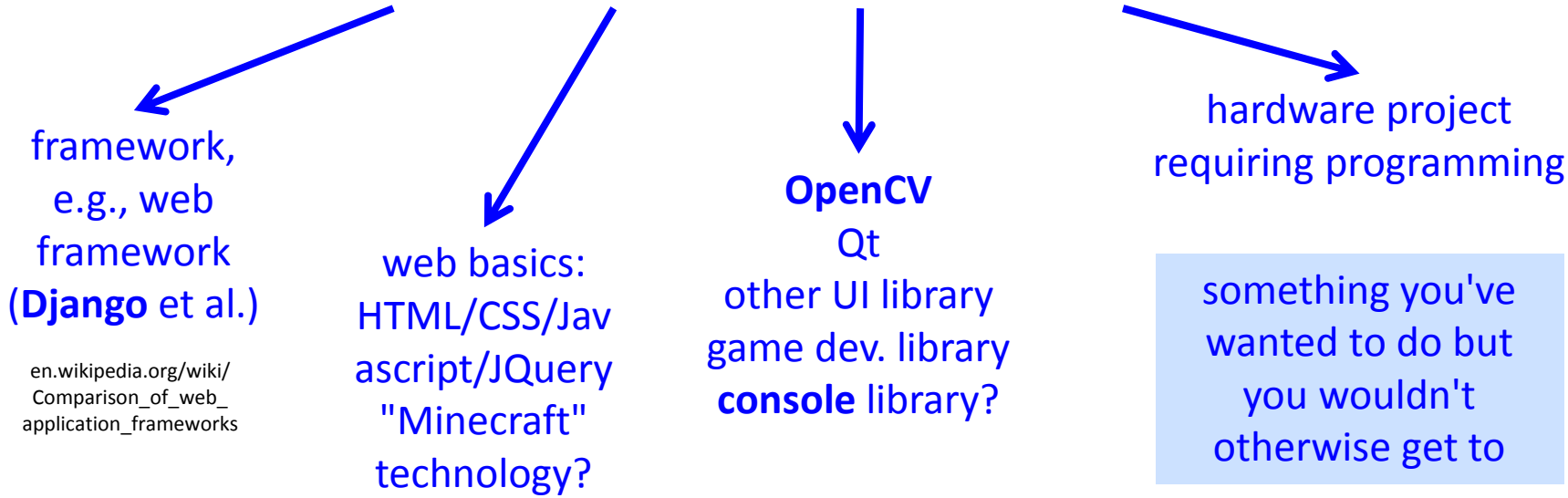
(2) create a project and a write-up…

(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



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.

then we'll repeat the process...

Project Grading

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

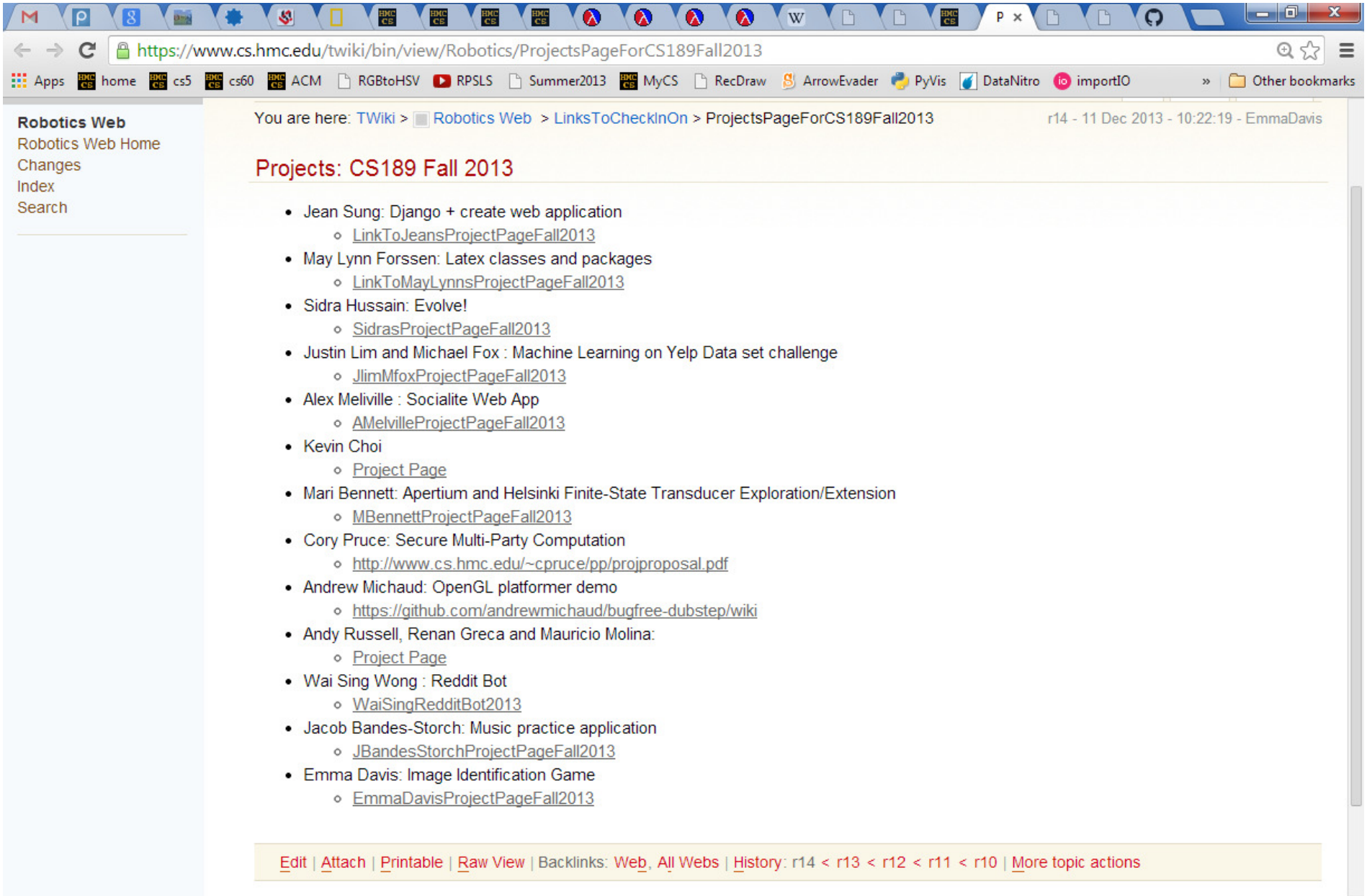
You can earn up to 10 problems for each project

Here is the breakdown:

4 problems	good-faith 3 hrs/week	**1-3** if less time or more intermittent
3 problems	weekly progress log, e.g., on CS wiki	**1-2** if more sporadic
3 problems	results! ~ a reasonable deliverable	**1-2** if it didn't come together

...exceptional results are welcome & recognized!

Examples from last term….



https://www.cs.hmc.edu/twiki/bin/view/Robotics/ProjectsPageForCS189Fall2013

Apps | home | cs5 | cs60 | ACM | RGBtoHSV | RPSLS | Summer2013 | MyCS | RecDraw | ArrowEvader | PyVis | DataNitro | importIO | Other bookmarks

Robotics Web
Robotics Web Home
Changes
Index
Search

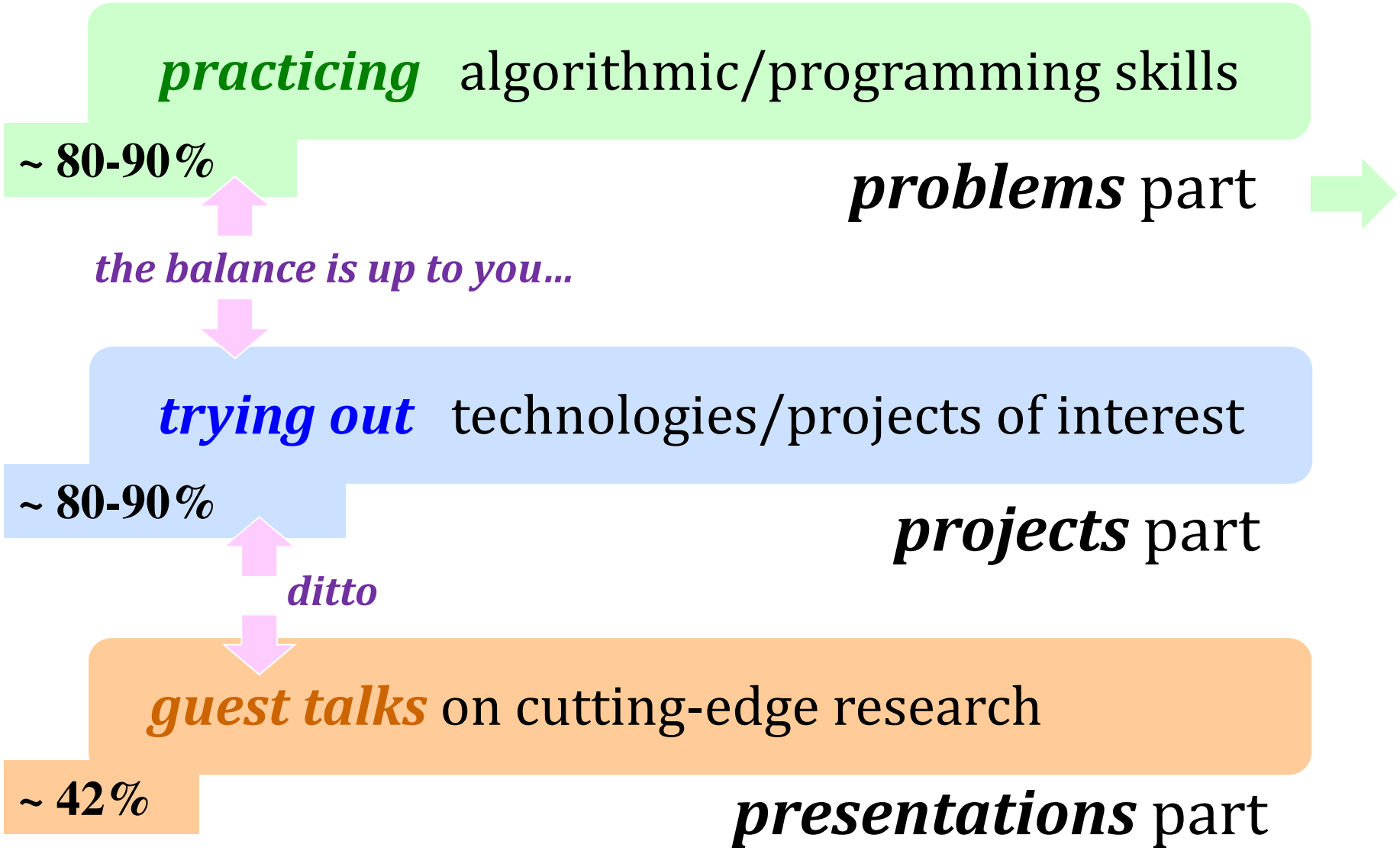
You are here: TWiki > Robotics Web > LinksToCheckInOn > ProjectsPageForCS189Fall2013 r14 - 11 Dec 2013 - 10:22:19 - EmmaDavis

Projects: CS189 Fall 2013

- Jean Sung: Django + create web application
  - LinkToJeansProjectPageFall2013
- May Lynn Forssen: Latex classes and packages
  - LinkToMayLynnsProjectPageFall2013
- Sidra Hussain: Evolve!
  - SidrasProjectPageFall2013
- Justin Lim and Michael Fox : Machine Learning on Yelp Data set challenge
  - JlimMfoxProjectPageFall2013
- Alex Meliville : Socialite Web App
  - AMelvilleProjectPageFall2013
- Kevin Choi
  - Project Page
- Mari Bennett: Apertium and Helsinki Finite-State Transducer Exploration/Extension
  - MBennettProjectPageFall2013
- Cory Pruce: Secure Multi-Party Computation
  - http://www.cs.hmc.edu/~cpruce/pp/projproposal.pdf
- Andrew Michaud: OpenGL platformer demo
  - https://github.com/andrewmichaud/bugfree-dubstep/wiki
- Andy Russell, Renan Greca and Mauricio Molina:
  - Project Page
- Wai Sing Wong : Reddit Bot
  - WaiSingRedditBot2013
- Jacob Bandes-Storch: Music practice application
  - JBandesStorchProjectPageFall2013
- Emma Davis: Image Identification Game
  - EmmaDavisProjectPageFall2013

Edit | Attach | Printable | Raw View | Backlinks: Web, All Webs | History: r14 < r13 < r12 < r11 < r10 | More topic actions

What is this course about?



Problems!

practicing algorithmic/programming skills

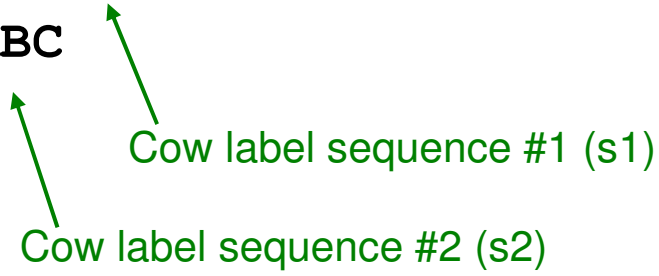
Bessie!

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

The **cowqueue** problem

Input

ABACB
AABC



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

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

Output

LCS(i1, i2) = length of longest common subsequence of s1 up to i1 and s2 up to i2

Strategy

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

LCS problem

```
s1 = "ABACB"          Input          s2 = "AABC"
          ↑                                 ↑
          i1                                i2
```

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

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

LCS code

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

cowqueue_recursive.py – /Users/zdodds/Desktop/cowqueue_recursive.py

```
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

} optimizing ***coding*** time, as well as ***running*** time

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

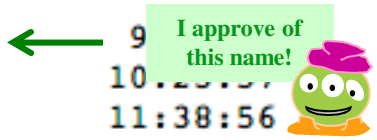
Scoreboard SoCal 2013 Contest

final standings

#	TEAM	SCORE		PENALTIES	1	2	3	4	5	6	7	8	9	10
1	USC Trojans	8	19:40:07	1	1 (00:27:25)	1 (03:13:40)	0	1 (02:26:11)	1 (01:11:15)	2 (04:50:02 + 00:20:00)	1 (04:38:29)	1 (01:48:00)	0	1 (00:45:05)
2	UCI constructors	7	16:22:17	1	1 (00:43:36)	0	0	1 (01:37:46)	1 (00:56:43)	1 (03:32:50)	1 (03:09:14)	1 (02:44:40)	0	2 (03:17:28 + 00:20:00)
3	Caltech A	6	09:54:23	1	1 (00:18:25)	1	0	1 (00:50:10)	1 (01:30:57)	5	2 (03:51:22 + 00:20:00)	1 (01:11:33)	0	1 (01:51:56)
4	Caltech B	6	13:13:25	3	1 (00:25:20)	1	0	1 (02:28:28)	1 (01:34:11)	2 (01:38:16 + 00:20:00)	1	1 (02:09:09)	0	3 (03:58:01 + 00:40:00)
5	CPSLO - The NULL Terminators	6	17:29:06	2	1 (02:11:34)	0	0	1 (00:54:31)	1 (01:02:29)	3 (04:27:51 + 00:40:00)	1 (04:43:29)	0	0	1 (03:29:12)
6	UCLA Bruins	5	07:19:42	0	1 (02:06:22)	0	0	1 (01:08:30)	1 (00:37:04)	1 (02:39:49)	0	1	0	1 (00:47:57)
7	USC Cardinal	5	09:01:56	2	1 (01:11:44)	0	0	1 (00:25:23)	1 (01:09:48)	3 (04:17:21 + 00:40:00)	0	1	0	1 (01:17:40)
8	UCSD - Load, Spin, Pull	5	09:52:08	1	1 (02:06:50)	0	0	1 (00:52:11)	1 (01:11:15)	0	2 (04:54:43 + 00:20:00)	0	0	1 (00:27:09)
9	HMC Escher	5	10:25:49	0	1 (02:47:54)	0	0	1 (01:01:56)	1 (00:50:06)	0	1	1 (03:21:52)	0	1 (02:24:01)
10	HMC Hayden	5	11:06:09	2	1 (02:16:54)	0	0	1 (00:37:02)	1 (01:09:38)	3 (03:46:47 + 00:40:00)	0	1	0	1 (02:35:48)
11	UCLA Gold	4	06:06:13	1	2 (00:50:33 + 00:20:00)	0	0	1 (01:38:46)	1 (02:06:20)	2	0	1	0	1 (01:10:34)
12	Pomona 47	4	07:57:36	2	1 (01:06:45)	0	0	1 (00:52:49)	3 (02:51:40 + 00:40:00)	0	0	0	0	1 (02:26:22)
13	UCSD - scissors	4	08:14:54	0	1 (02:41:37)	0	0	1 (01:22:46)	1 (01:44:18)	0	0	1	0	1 (02:26:13)
14	SBCC One	4	09:17:46	1	1 (00:56:40)	0	0	1 (01:01:08)	1 (03:25:27)	1	0	2	0	2 (03:34:31 + 00:20:00)
15	UCSB - Alpha	4	09:18:05	1	1 (00:47:35)	0	0	1 (01:24:05)	1 (02:07:21)	1	1	2	0	2 (04:39:04 + 00:20:00)
16	Caltech C	4	10:43:46	2	1 (02:19:00)	2	0	1 (01:25:08)	1 (02:02:30)	0	0	0	0	3 (04:17:08 + 00:40:00)
17	HMC Hammer	4	11:13:16	0	1 (01:38:38)	0	0	1 (02:16:10)	2	1 (03:28:49)	0	0	0	1 (03:49:39)
18	USC Gold	4	11:27:02	2	1 (00:27:02)	0	0	1 (04:54:31)	1 (01:21:19)	0	0	0	0	3 (04:04:10 + 00:40:00)

2012-13 Final Standings

Rank	Team ID	Team Name	Solved	Penalties	Score
1	acm170	USC Trojans	9	11	24:59:34
2	acm107	Caltech A	8	3	17:25:48
3	acm151	UCLA Flyaway	6	2	7:23:47
4	acm122	UCSD Load, Spin, Pull	6	0	10:31:29
5	acm124	UCSD kamehb	6	5	11:49:16
6	acm121	HMC Squared	5	0	9
7	acm168	USC Cardinal	5	2	10
8	acm152	UCLA HeroesIII	5	2	11:38:56
9	acm157	UCI constructors	5	3	11:54:00
10	acm123	UCSD bumaga	4	1	5:20:39
11	acm109	Caltech D	4	3	7:43:22
12	acm119	HMC J	4	4	8:02:39
13	acm154	UCSB alpha	4	2	8:47:20
14	acm106	Caltech 1	4	3	9:05:09
15	acm158	UCI instances	4	2	9:27:40
16	acm111	CSUF-B	4	1	9:40:47
17	acm117	CSULB Undeclared Identifiers	4	2	10:22:22
18	acm108	Caltech C	4	3	10:39:47
19	acm118	HMC Escher	4	1	10:46:15
20	acm129	UCR Raphael	4	0	11:37:09



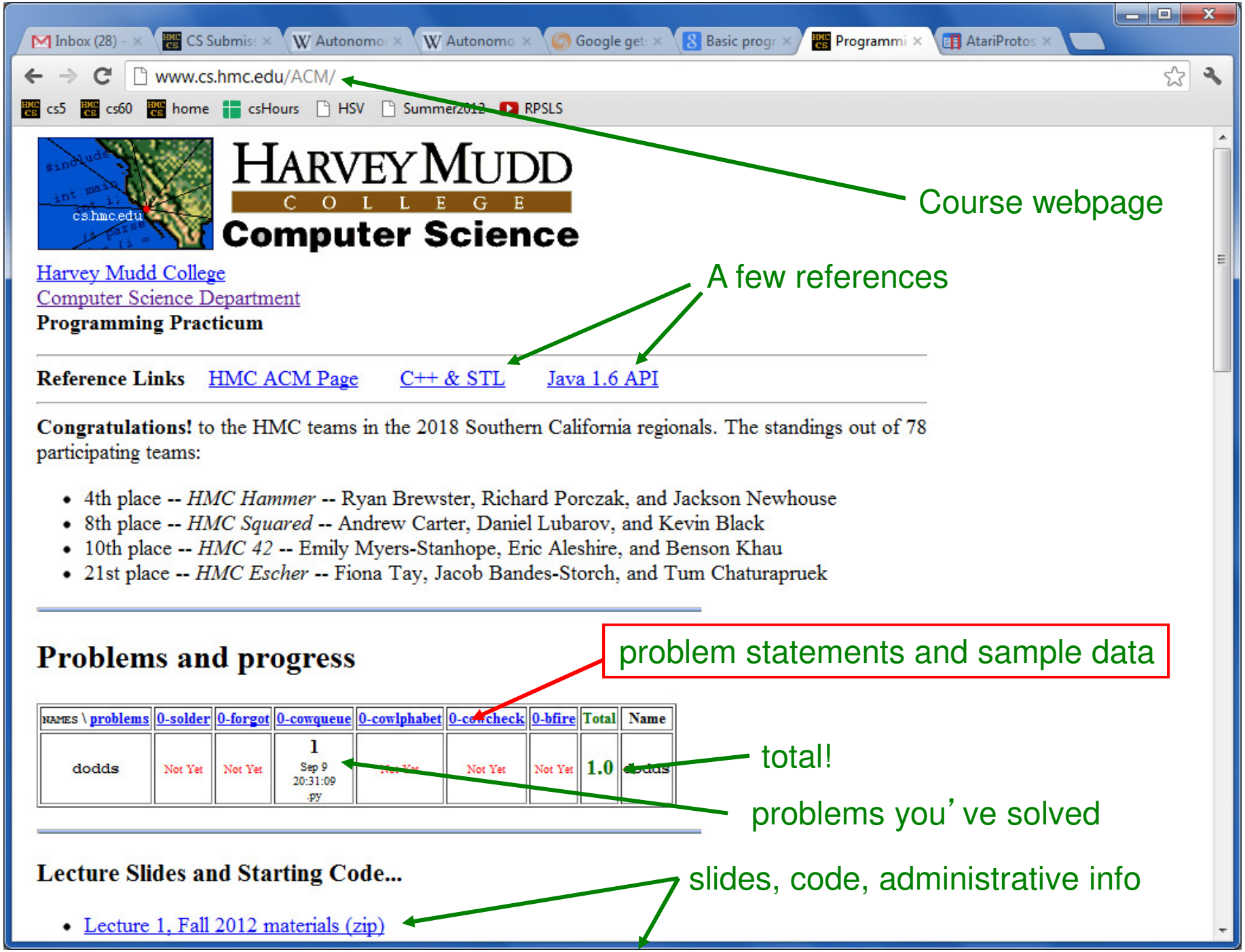
USC advanced to the finals in 2011, 2012, and 2013...

Fluxx…


active
watching!


active
watching!


Rank Team
1 HMC 42
2 Caltech B
3 SLO - Semicolons of Fury
4 Caltech A
5 USC Trojans
6 UC San Diego Jiandao (Scissors)
7 SLO - Blinkenlights
8 UC San Diego Yanshi (Rock)
9 HMC Hammer
10 USC Tirebiters
11 UCLA True Bruins
12 HMC Escher
13 HMC Alien
14 UNLV Ballmer Peak
15 UCLA IDK
16 USC Cardinal
17 Caltech C
18 SLO - Team Ruby Slippers
19 UCSB GoedelEscherBach
20 CPP H1N1++
21 UNLV - Nerd Sniping
22 UCSB Stroustrup's Henchmen
23 UCLA True Blue
24 UC San Diego Wenjian (Paper)
25 CLU NINJAS
26 UCSB Number Crushers
27 UCLA Fight! Fight! Fight!
28 SLO - Bad Horse's Mares
29 UCI CoderAnteaters
30 CPP #define L lambda; Lf*(Lf(x
31 Caltech D
Svd Pen Score
4 2 7:54:15
4 2 9:11:21
4 5 11:07:19
4 3 13:08:38
3 7 8:45:34
3 2 8:46:25
2 0 2:02:45
2 3 3:54:36
2 2 3:56:17
2 0 3:56:32
2 2 4:03:58
2 4 4:50:26
2 3 5:24:14
2 7 6:26:44
2 2 7:16:13
2 5 7:29:32
1 0 0:23:42
1 0 0:23:43
1 0 0:34:28
1 0 0:37:17
1 0 0:40:21
1 0 0:50:23
1 0 0:59:24
1 2 1:24:30
1 0 1:47:47
1 0 2:13:44
1 1 2:33:57
1 0 2:43:08
1 1 2:47:35
1 0 2:49:55
1 0 2:51:32
Rank Team
32 UC San Diego Chui (Hammer)
33 UC Riverside Alpha
34 UC Riverside Omega
35 MC Raiders
APU Army Ants
APU Perpetual Motion Squad
APUters
CLU Killer Penguins
CLU Leaky Capacitors
CPP Furloughs!
CPP Eyes
CSUF
CSUDH Toro1
CSUDH Toro2
CSULA1
CSUCI_1
CSUCI_2
CSUCI_3
MtsacBlue
MtSACgold
MtsacRed
PLNU Sea Lions
RCC::Chinchilla
RCC::Commando
SDSU import awesome;
SDSU Team T10
SDSU Team T100
UndisclosedClosetLemursAssoc
UCSB Soapfoot
UNLV - e to the pi Minus pi
USC Gold
Test Never Scores Either
Test Never Scores
Svd Pen Score
1 2 3:17:32
1 0 4:51:42
1 1 5:01:40
1 1 5:02:52
0 0 0:00:00
0:01:05 remaining in the contest.
Scoreboard Last Updated Sat Nov 7 19:18:55 2009
The contest will end at Sat Nov 7 19:20:00 2009.
active
watching!


www.cs.hmc.edu/ACM/
Course webpage
HARVEY MUDD COLLEGE
Computer Science
Harvey Mudd College
Computer Science Department
Programming Practicum
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
problem statements and sample data
NAMES \ problems
0-solder
0-forgot
0-cowqueue
0-cowlphabet
0-cowcheck
0-bfire
Total
Name
dodds
Not Yet
Not Yet
1
Sep 9 20:31:09 .py
Not Yet
Not Yet
Not Yet
1.0
dodds
total!
problems you've solved
Lecture Slides and Starting Code...
slides, code, administrative info
Lecture 1, Fall 2012 materials (zip)

Details

you can work in teams of up to 3 people ~ must share the work equally

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

additions will also be considered…

Grading

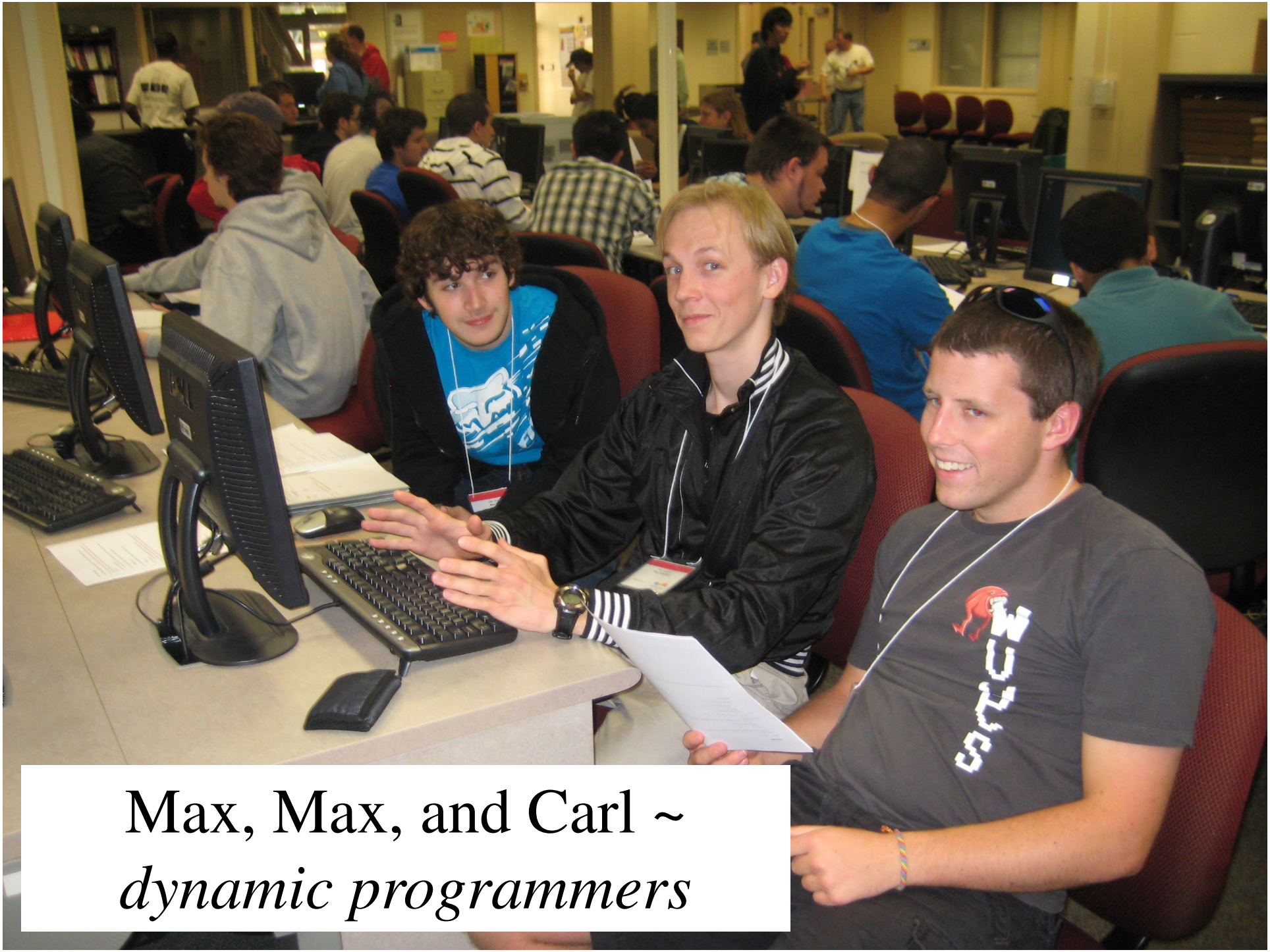
CS 189 is graded by default ... (it's possible to take it P/F, too)

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**

# Solved (out of 42)	Assessment
43+	pretty much impossible!
28-42	A
23-27	A-
20-22	B+
17-19	B
14-16	B-
9-13	C range
≤ 9	< D range or less



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

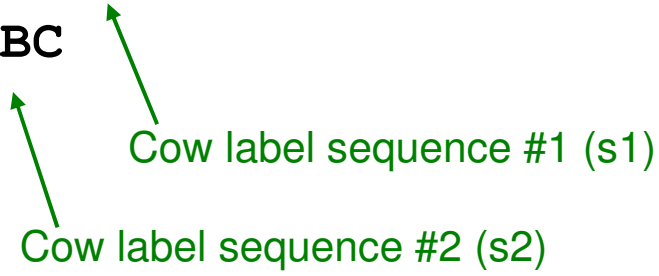
(2) **Dynamic programming**

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

The **cowqueue** problem

Input

ABACB
AABC



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

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

Output

LCS(i1, i2) = length of longest common subsequence of s1 up to i1 and s2 up to i2

Strategy

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

LCS problem

```
s1 = "ABACB"          Input          s2 = "AABC"
          ↑                                 ↑
          i1                                i2
```

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

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

LCS code

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

cowqueue_recursive.py – /Users/zdodds/Desktop/cowqueue_recursive.py

```
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

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

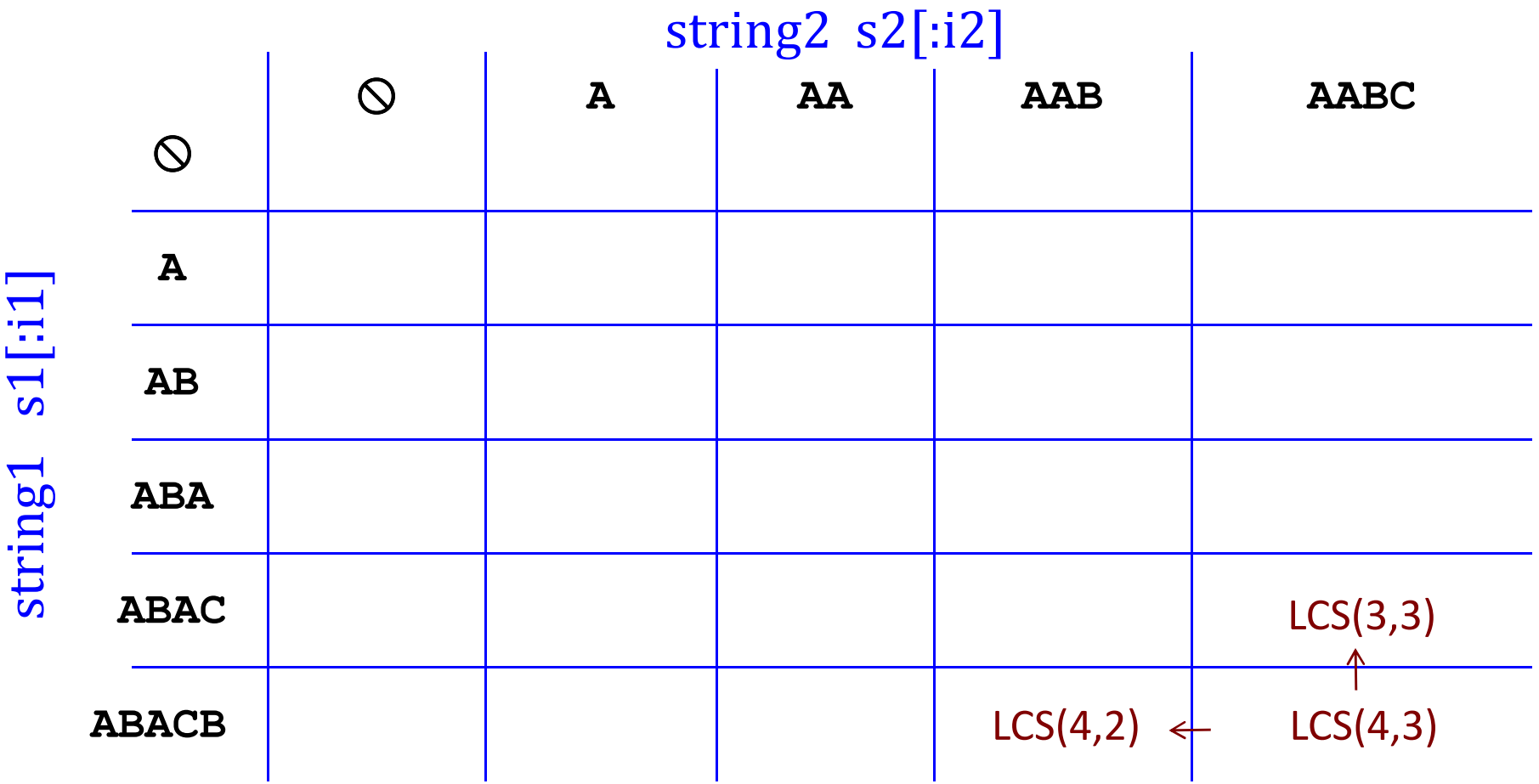
string2 s2[:i2]

string1 s1[:i1]

	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					LCS(4,3)

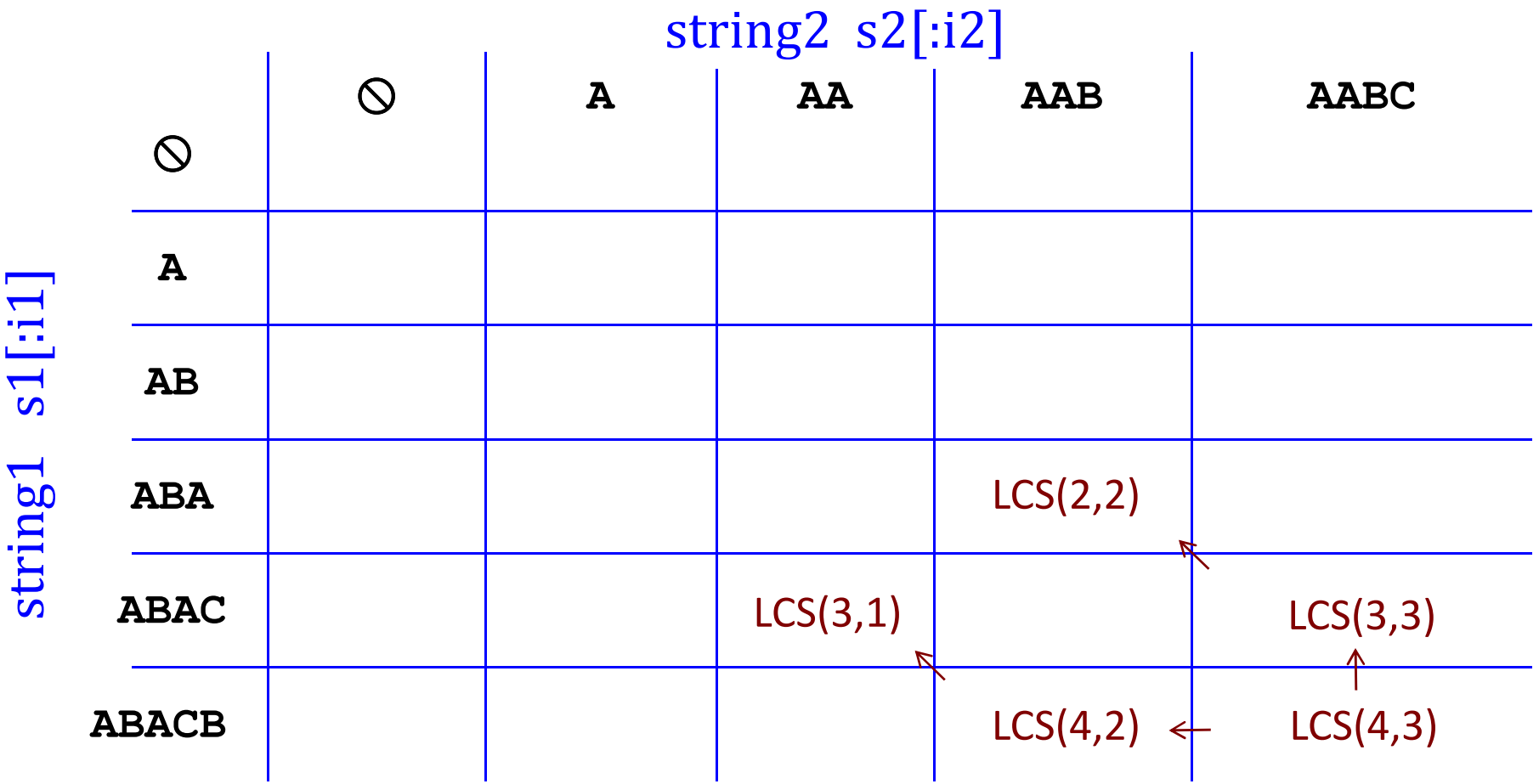
LCS idea

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```



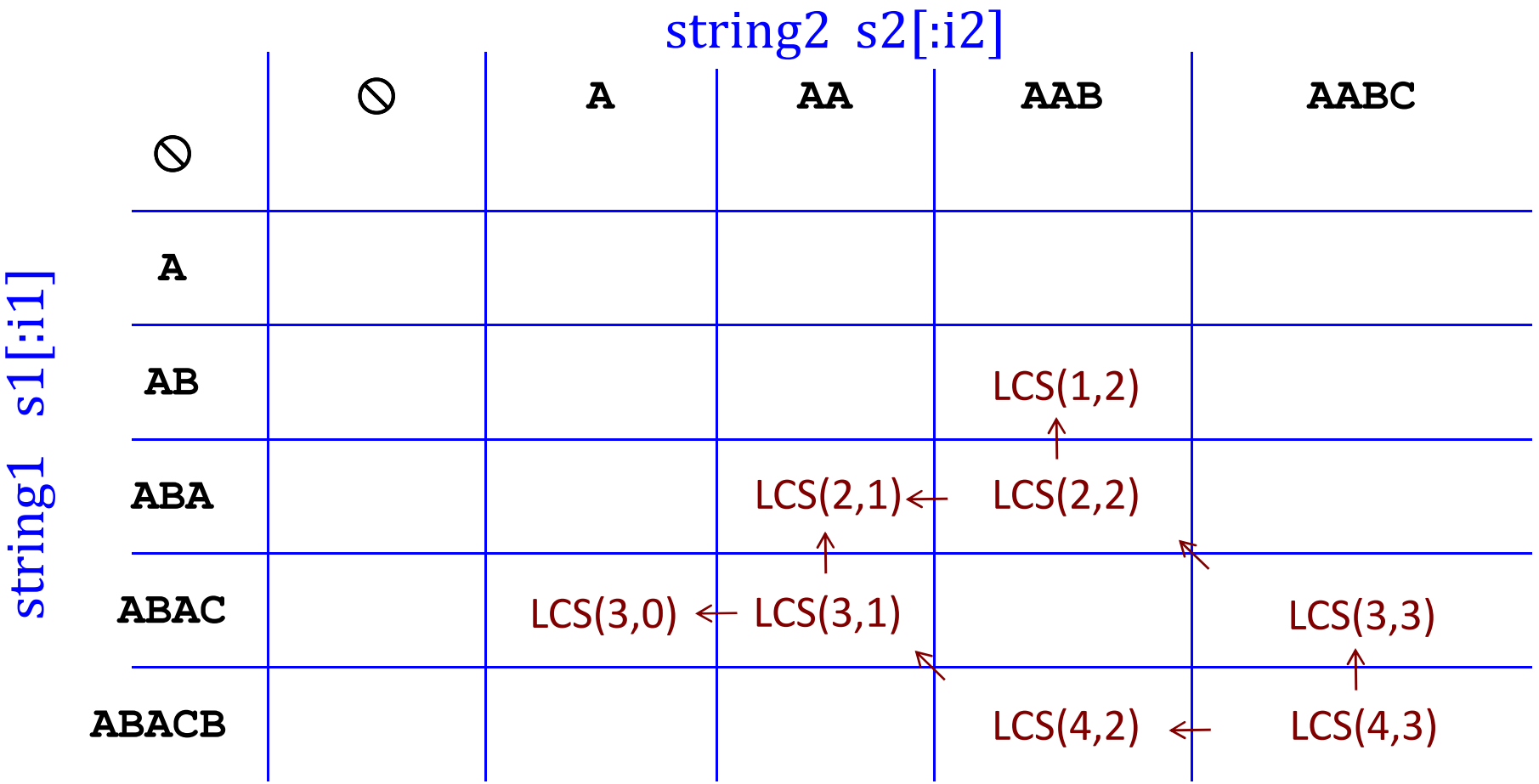
LCS idea

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```



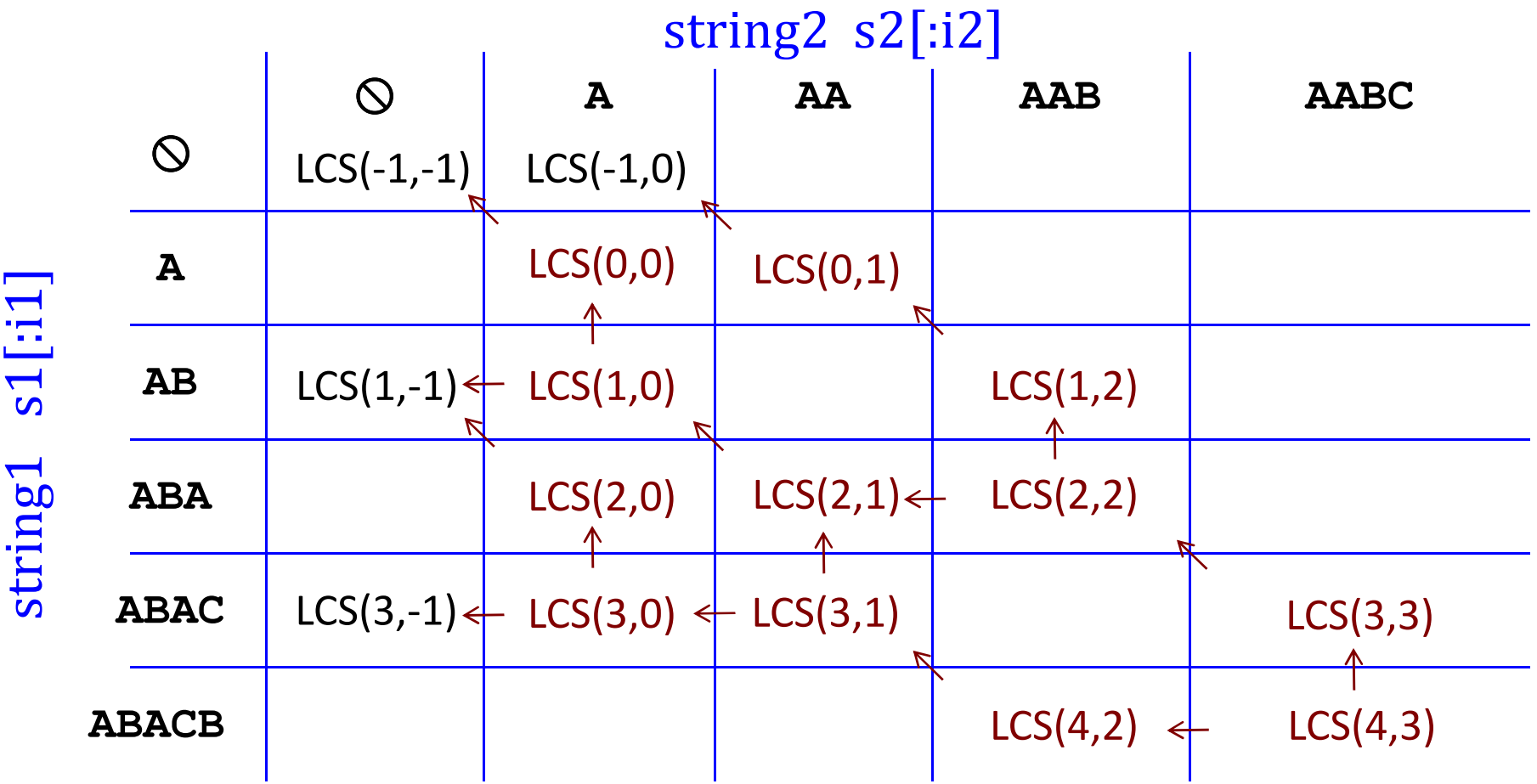
LCS idea

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```



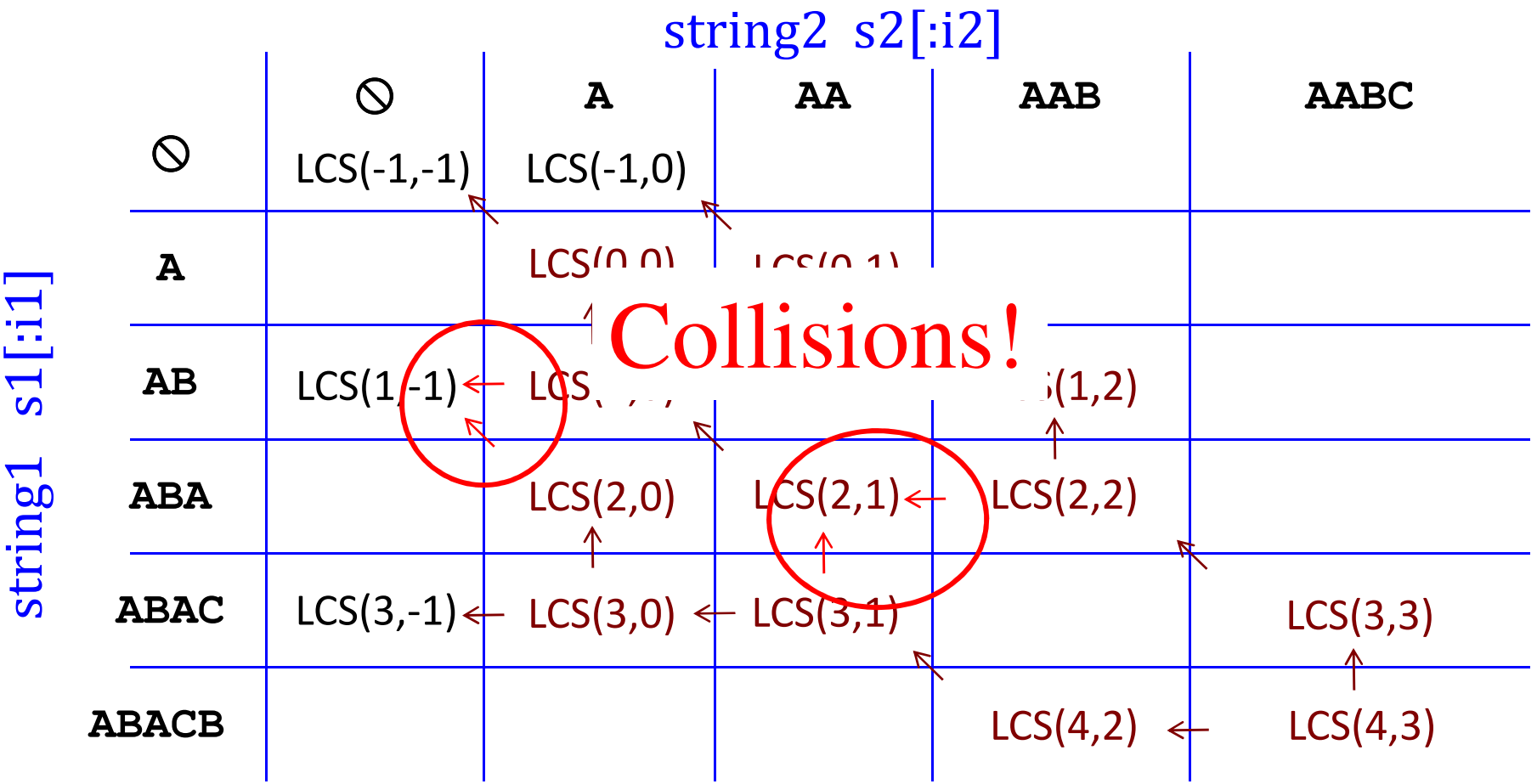
LCS idea

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```



LCS idea

```
s1 = "ABACB"      Input      s2 = "AABC"
      ↑                            ↑
      i1                           i2
```



LCS, memoized

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

```
# 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
```

Python *function decorators*

```
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]
```

Python's "function decorator" syntax!

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

LCS, DP'ed

Compute the table of results, bottom-up!

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

string2 s2[:i2]

string1 s1[:i1]	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					

LCS, DP'ed

Compute the table of results, bottom-up!

```
s1 = "ABACB"          Input          s2 = "AABC"
       ↑                                   ↑
       i1                                  i2
```

string2 s2[:i2]

string1 s1[:i1]

	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					

```
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

Notes, starting code, slides, etc. ...

- Lecture 1, cowqueue code examples (zip)

Problems and progress

NAMES \ problems	0-smount	0-lazy	0-elevator	0-cowqueue	0-cowcash	0-ave	Total	Name
dodds	Not Yet	Not Yet	Not Yet	1.5 Sep 9 16:19:24 .py	Not Yet	Not Yet	1.5	dodds

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% ***projects*** part

ditto

guest talks on cutting-edge research

~ 42% ***presentations*** part

Guest lectures…

count as 1.5 problems

many job candidates…

www.cse.ohio-state.edu/~eckroth/

Apps | home | cs5 | cs60 | ACM | RGBtoHSV | RPSLS | Summer2013 | MyCS | RecDraw | ArrowEvader | PyVis | » | Other bo

Joshua Eckroth

PhD candidate majoring in Artificial Intelligence, minoring in Cognitive Science and Mathematical Logic, at The Ohio State University, Department of Computer Science and Engineering.

My major advisor is Dr. John R. Josephson.

My minor advisors are Prof. Neil Tennant (Cognitive Science) and Prof. Harvey Friedman (Mathematical Logic).

Research Interests

My research addresses the question, how can bounded, situated, autonomous cognitive agents enhance their performance, i.e., think faster and smarter? My approach is to explore how a self-reflective metareasoning component can enable the agent to "think fast and loose" in the usual case, while monitoring its performance, but when confused, focus in on possible mistakes and determine the appropriate corrections.

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

Many weeks we will have a guest speaker …

Jotto!

A word-guessing game similar to mastermind…

Sophs	JRs	SRs	POM-CMC-SCR-PTZ	*other*
diner ?	**diner ?**	**diner ?**	**diner ?**	**diner 2**

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...