Sports: Harvey Mudd professor creates new number never previously seen. Scientists are excited!
Weather: Record new temperature observed yesterday. “It’s not particularly high or low, it’s just different.”

# The CS 5 Green Gazette

## New Largest Infinite Set Discovered by P.I.T. Researcher

(Pasadena, AP): A researcher at the Pasadena Institute of Technology has discovered an infinite set that is larger than any previously known infinite set. “This discovery is absolutely unreal!” said one leading authority in the field. “We thought that we already had the largest infinite set and we weren’t counting on a new one being discovered.” Some critics warned against becoming “irrationally exuberant” about this result, however. “People are galloping to conclusions about how important this is. I’d caution folks to slow down their enthusiasm to a trot, or perhaps even a Cantor,” said one expert. Wow, this headline is probably the cheesiest one of the entire semester. Whoever wrote this (ahem!) clearly needs some vacation.

**INPUTS**

"Science without religion is lame, religion without science is blind."

"Two things are infinite: the universe and human stupidity; and I'm not sure about the universe."

"Duct tape is like the force, it has a light side, a dark side, and it holds the world together."

"If you die in an elevator, be sure to push the Up button."

"All generalizations are false, including this one."

"Clearly you've never been to Singapore!"

"Luke, I am your father."

"To be, or not to be."

"You shall not pass!"

**(… all with authors …)**

Markov-generated wisdom!

**OUTPUTS**

"I have a dream! Duct tape is written on. Luke, I am your thoughts and what lies within us."

**---- Audrey Rooney**

"Your work is lame, religion is nearly the Up button."

**---- Abraham Marx**

"Two things are false, including this one."

**---- Captain_Jack Truman**

# CS 5 Green Finale

***Looking back!*** Evals, Ideas

***Looking ahead?*** Options…

**CS 5G Final lecture**
*now!*

I'll be back…

but that's my line!

CS 5, on the verge
of *termination*

**CS 5G Final Projects**

- Thu 1:15-3:15pm: Open lab to finish projects
- due Fri 5pm (Sat w/ Euro)
- Normal office hours through Friday of this week

**CS 5G Final Exam**

- this room, Tues 12/14, 2-5pm
- comprehensive
- 20-min practice problems on course website
- one 8.5x11 sheet front and back (that you prepare yourself) permitted
- come see us in office hours (or outside office hours)

# Studying for the final

- Review the lecture notes
- Look over the homework problems
- Do practice problems
- Prepare your 8.5x11 sheet

# CS 5 Green: where we've been…

## Concepts

Functions, variables

Conditionals, loops

Data types: lists, dictionaries, strings, tuples

Recursion, use-it-or-lose-it

Assembly language (Hmmm)

Classes and objects

Computability

## Applications

Gene finding

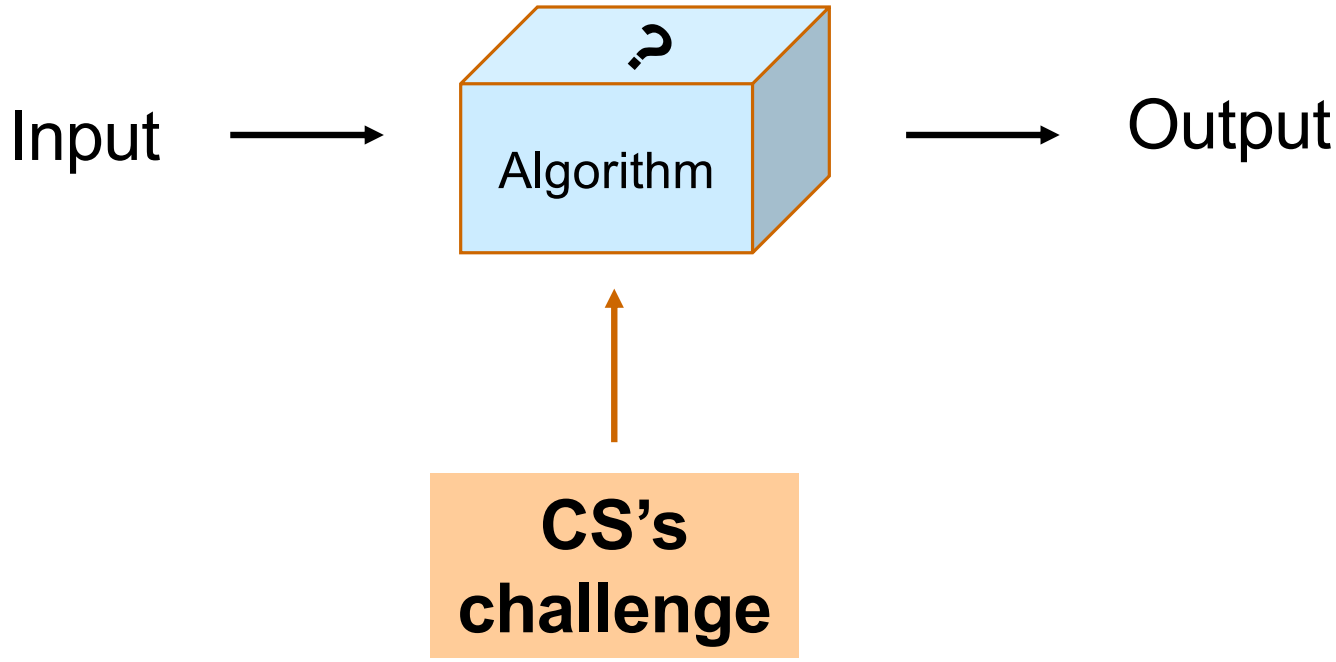
Read mapping

Alignment

RNA Folding

Phylogenetic trees

# The CS view of the world...



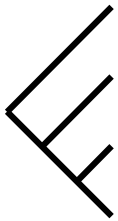
# Input→output in CS5 Green

Salmonella genomic sequence →

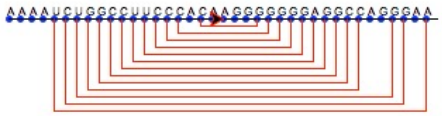
Human and chicken genes →

```
chr11 118415243 h4 --- chr24 5629899 c19
chr11 133938820 h6 --- chr24 2542440 c17
chr14 72399156 h9 --- chr5 28862733 c22
chr3 45016733 h17 --- chr2 43123243 c8
```

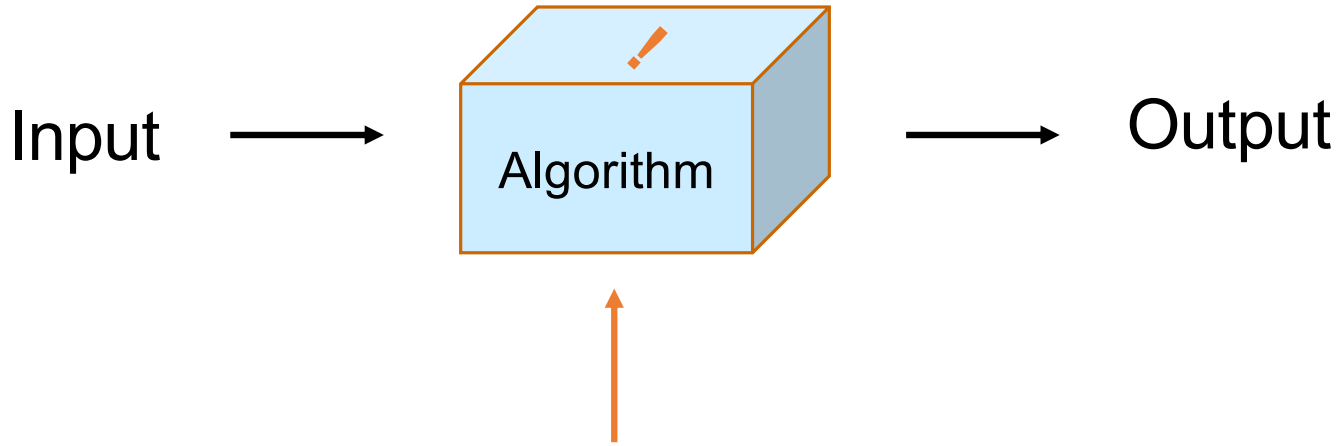
Mitochondrial DNA sequences →



RNA sequences →



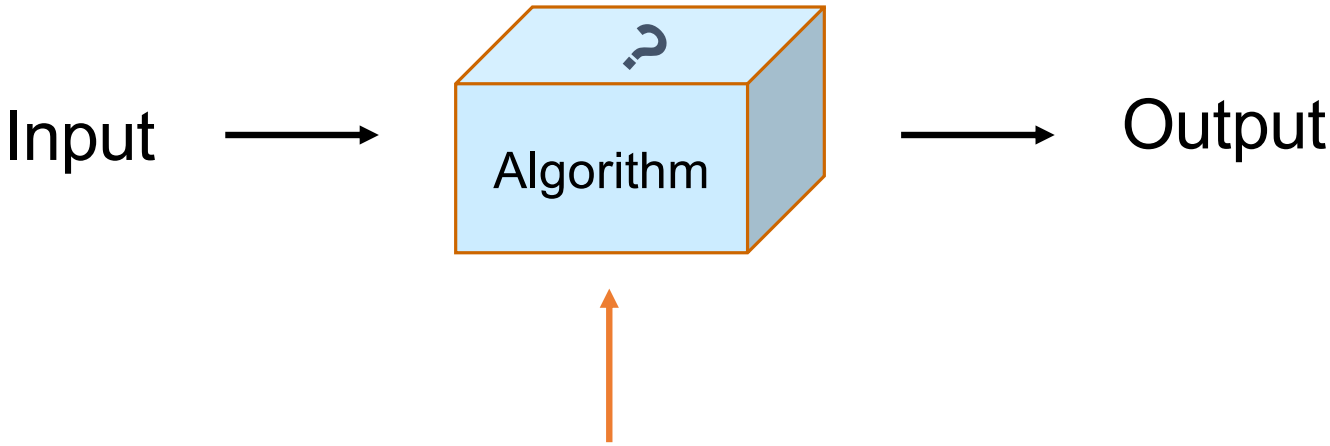
# Uncomputable Functions




**sometimes an algorithm simply does *not* exist…**

***more precisely**:* every possible algorithm contains bugs!

# Meaningful Functions?




Fortunately, nearly all ***meaningful*** functions ***<u>are</u>*** computable…

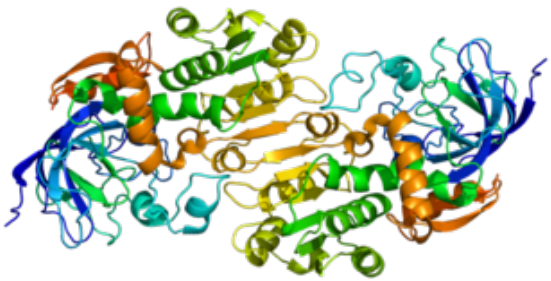
***but this doesn't mean we know how to compute them (yet)!***

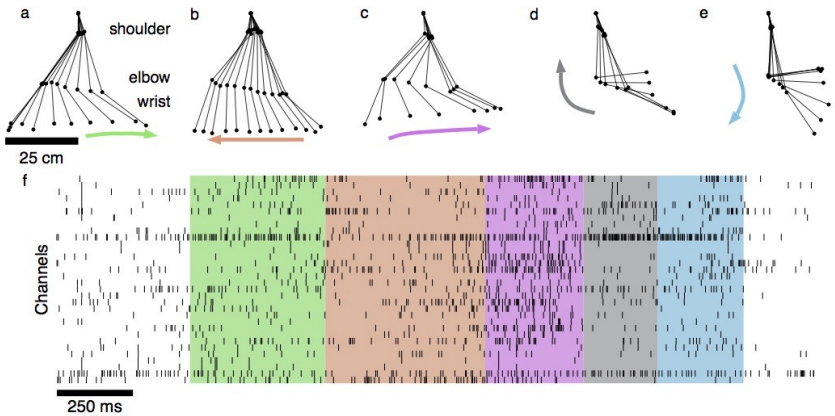
# Tasks

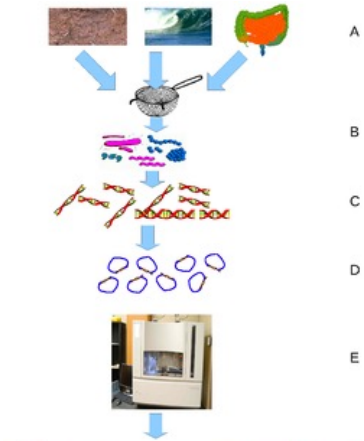

In the 60s, Marvin Minsky assigned a couple of undergrads to spend the summer programming a computer to use a camera to identify objects in a scene. He figured they'd have the problem solved by the end of the summer. Half a century later, we're still working on it.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

# Biology is a rich source of unsolved problems





# Computational methods for studying honey bee foraging behavior


```python
# scale_calibrate.py
# this file takes a bee video, and extracts the size of the checkerboar
# reference card, then calculates the dimensions and area of bee abdome

import cv2
import numpy as np
import argparse
import imutils
import statistics
import pandas as pd
from matplotlib import pyplot as plt


# take inputs
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to the input video")
ap.add_argument("-o", "--outdir", required=True,
                help="path to the out directory")
ap.add_argument("-l", "--leftside", type=bool, default=False, required=
                help="only use this boolean argument if card is the lef
ap.add_argument("-s", "--size", type=int, default=1.5, required=False,
                help="length of bee in cm")
ap.add_argument("-v", "--visualize", default=False, required=False,
                help="show visualizations")
args = vars(ap.parse_args())

LEFTSIDE = args['leftside']  # is the reference card on the left side
BEE_SIZE = args['size']  # length of bee in cm
VISUALIZE = args['visualize']  # show visualizations
OUT_DIR = args['outdir']  # directory for output
label = args['image'].split('.')[0].split('/')[-1]  # output filename
```

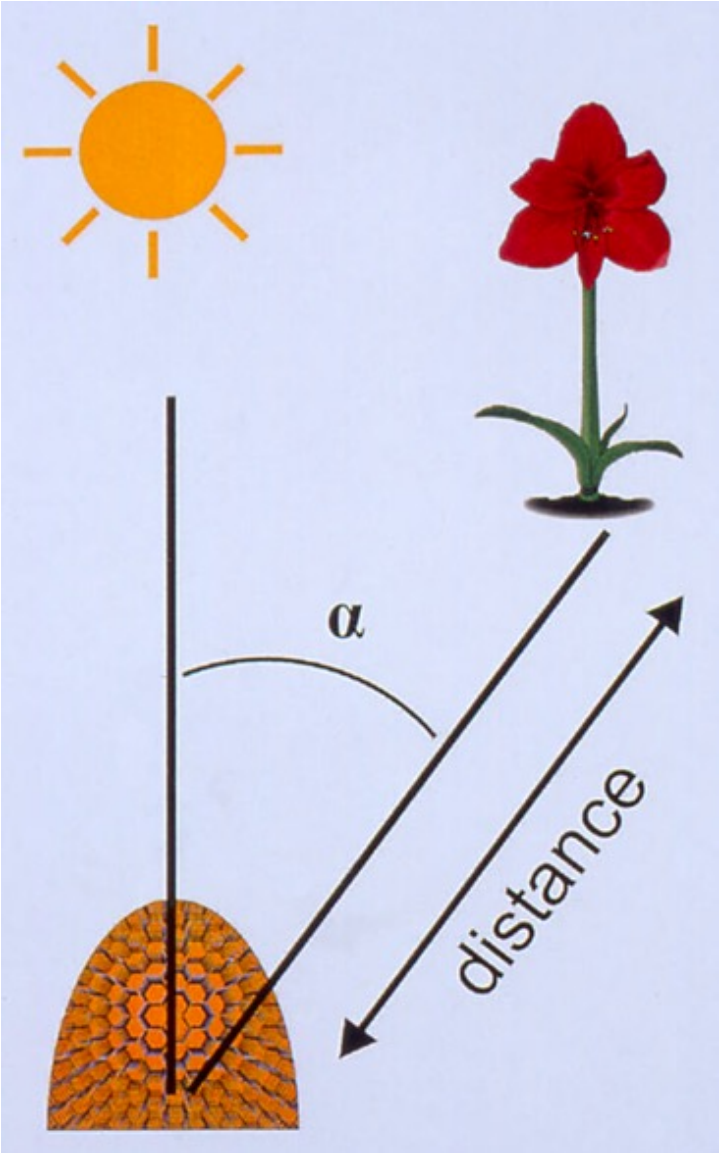
Morgan Carr-Markell

12/9/21

# Big question

Landscape ➡ Food ➡ Colony Health

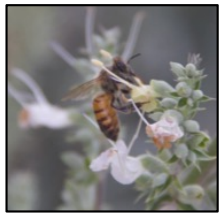
# Waggle dance

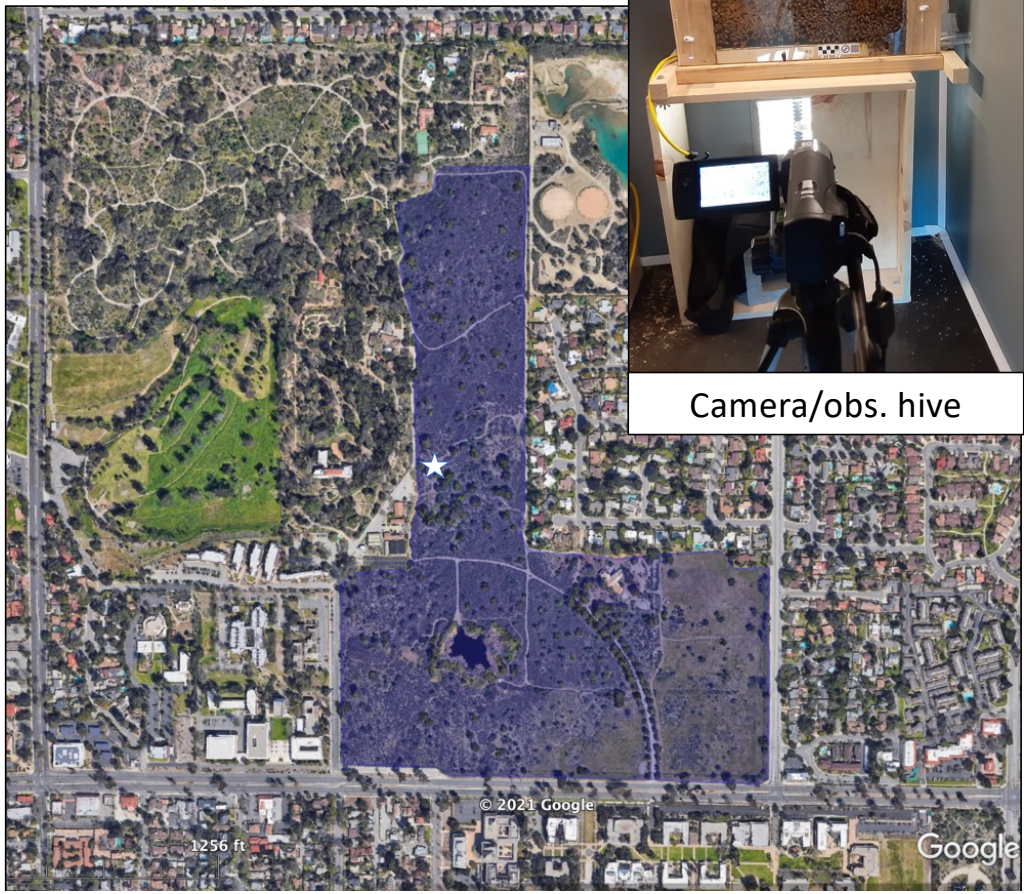

Waggle dance image from: Chittka (2004) *PLoS Biol*

# Research: Field + computational projects

- Bernard Field Station:
  - *Salvia apiana* (white sage)
  - *Eriogonum fasciculatum* (California buckwheat)

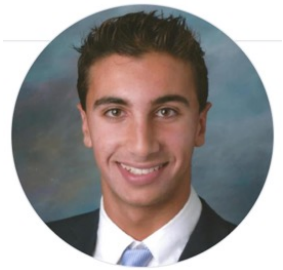


Prof Donaldson-Matasci




Camera/obs. hive

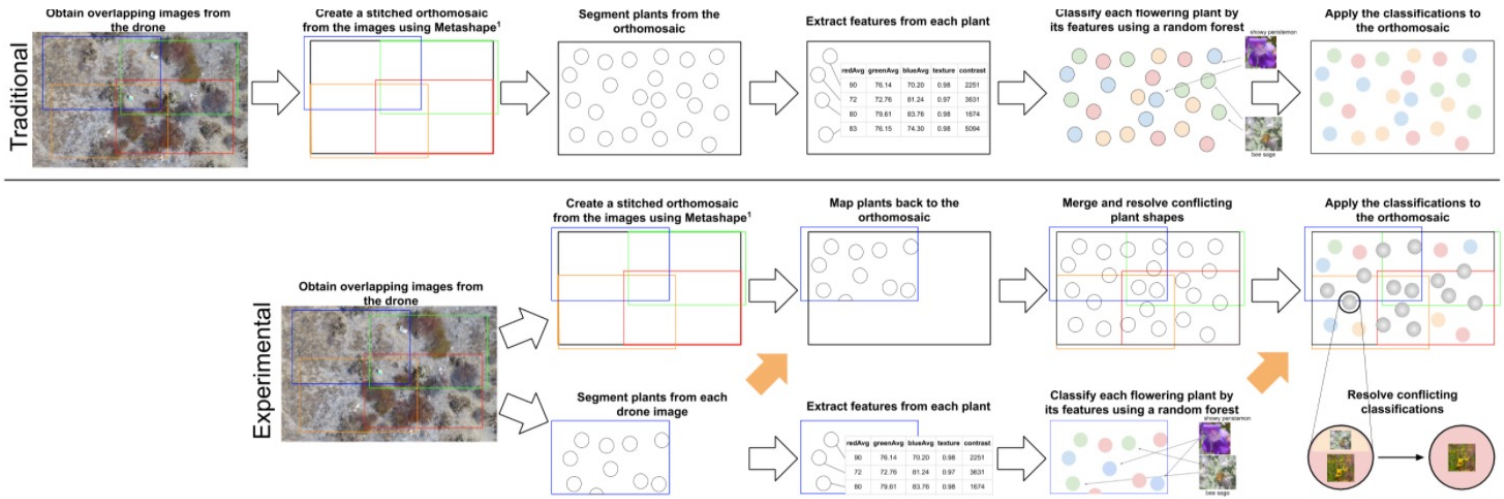
# Machine learning: IDing flowers in aerial drone images



**Arya Massarat**
aryarm

UCSD Bioinformatics and System Biology PhD Student

Matt Crane + DJI Phantom quadcopter



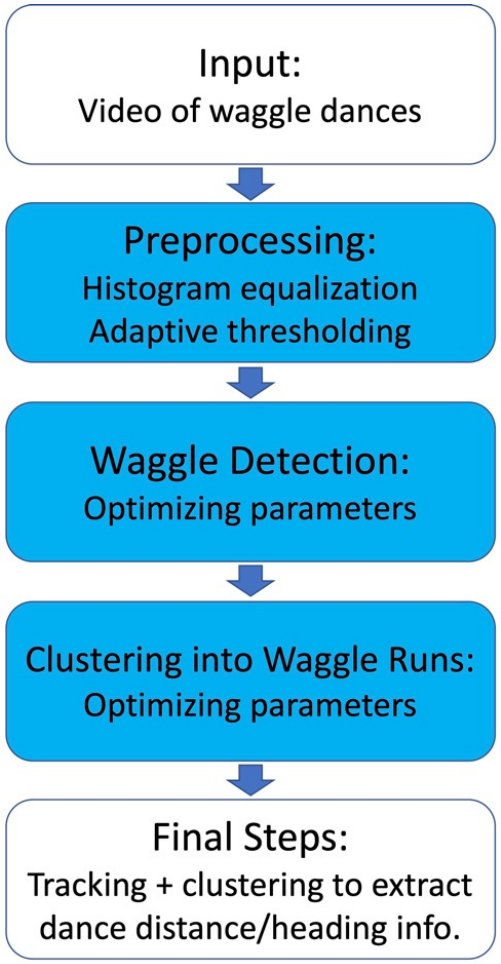

Many smaller steps of the pipeline were left out of this diagram for the sake of brevity. You can fully visualize every step in the pipeline using Snakemake.

Massarat, A. (2020) Mapping floral resources for bees using drone imagery. Senior Thesis, Harvey Mudd College

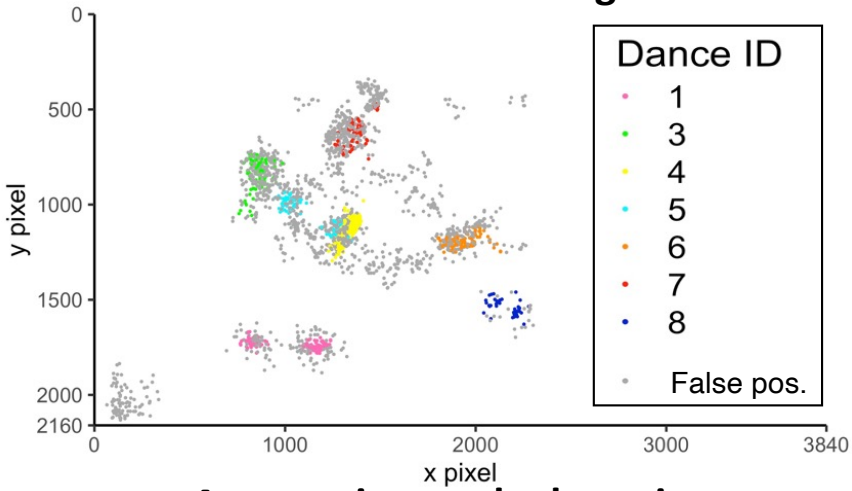
# Computer vision: Automatic waggle dance decoding

- Modified the new method (Reece et al. 2020) to better process videos with different:
  - Lighting
  - Resolution
  - Contrast
  - Frame rate
- Working to optimize parameters to achieve:

1) reliable waggle run detection
2) clustering of detections into individual waggle runs

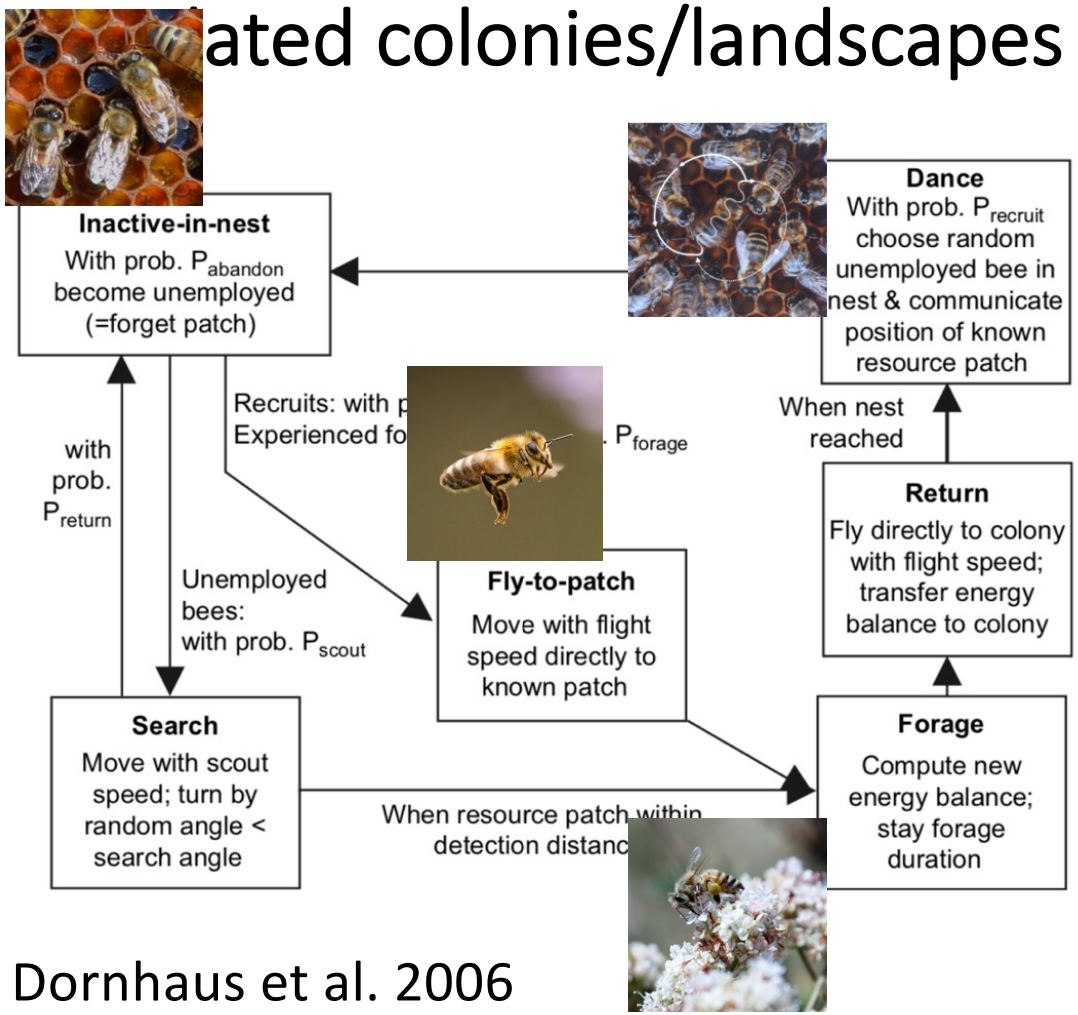


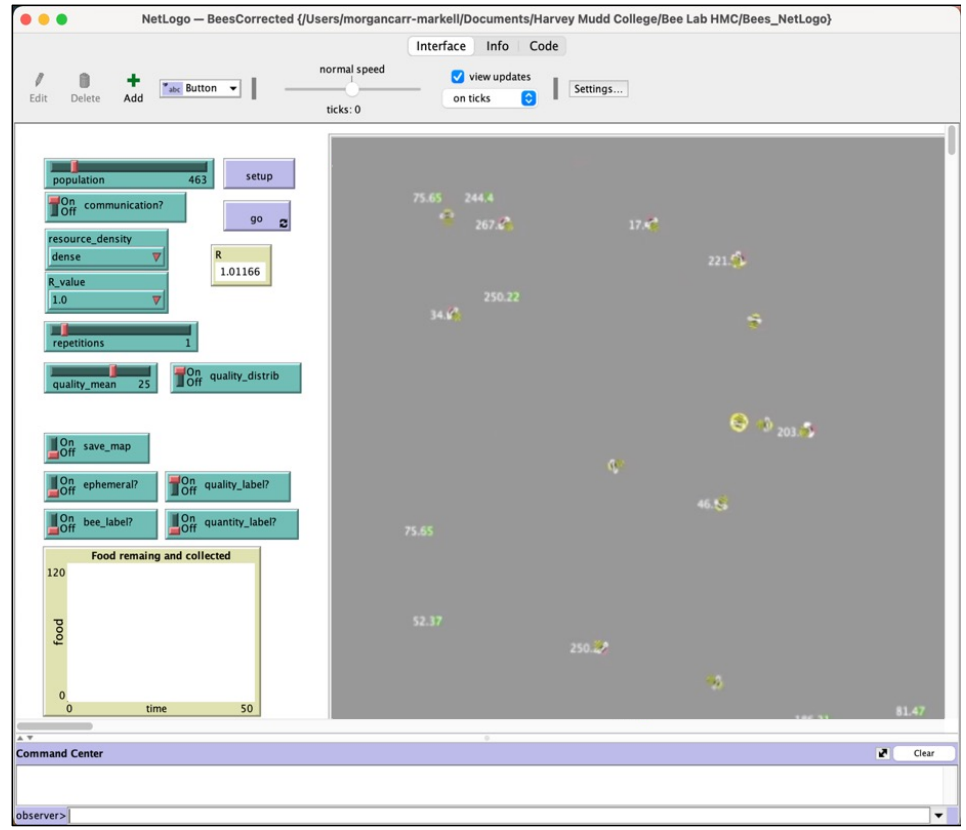

**Manual dance decoding results**




**Automatic waggle detections**

# Agent-based modeling: In-silico experiments with ated colonies/landscapes




Dornhaus et al. 2006

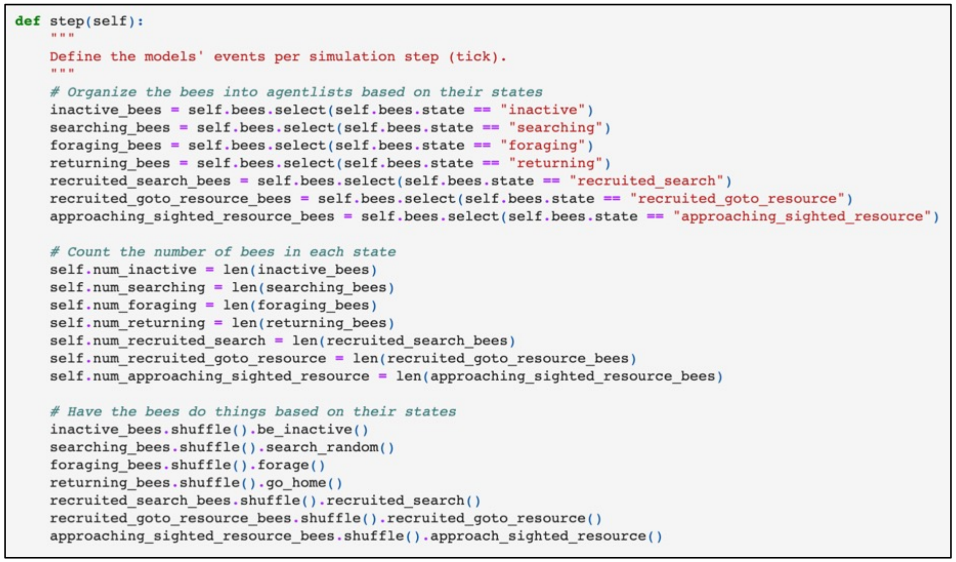
Woodman 2016

```
;;;;;;;;;;;;;;;;;;;;;
;;; State Machine ;;;
;;;;;;;;;;;;;;;;;;;;;

to go
  ; turtle stuff
  ask turtles
  [
    let dist-hive distancexy 0 0
    if (dist-hive > dist-hive-max) [ set dist-hive-max dist-hive ]

    ;; Actions based on states
    if state = "inactive-unemp"
    [ inactive-unemp ]
    if state = "inactive-emp"
    [ inactive-emp ]
    if state = "goto-resource"
    [ goto-resource ]
    if state = "random-search"
    [ random-search ]
    if state = "forage"
    [ forage ]
    if state = "return-to-hive"
    [ return-to-hive ]
    if state = "dance"
    [ dance ]

    ; Update states-transition
    if next-state != ""
    [
      set state next-state
      set state-list lput next-state state-list
      set next-state ""
    ]
  ]
```

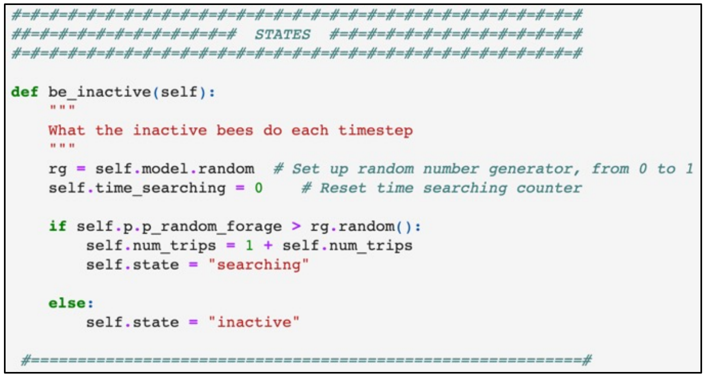


```python
def step(self):
    """
    Define the models' events per simulation step (tick).
    """
    # Organize the bees into agentlists based on their states
    inactive_bees = self.bees.select(self.bees.state == "inactive")
    searching_bees = self.bees.select(self.bees.state == "searching")
    foraging_bees = self.bees.select(self.bees.state == "foraging")
    returning_bees = self.bees.select(self.bees.state == "returning")
    recruited_search_bees = self.bees.select(self.bees.state == "recruited_search")
    recruited_goto_resource_bees = self.bees.select(self.bees.state == "recruited_goto_resource")
    approaching_sighted_resource_bees = self.bees.select(self.bees.state == "approaching_sighted_resource")

    # Count the number of bees in each state
    self.num_inactive = len(inactive_bees)
    self.num_searching = len(searching_bees)
    self.num_foraging = len(foraging_bees)
    self.num_returning = len(returning_bees)
    self.num_recruited_search = len(recruited_search_bees)
    self.num_recruited_goto_resource = len(recruited_goto_resource_bees)
    self.num_approaching_sighted_resource = len(approaching_sighted_resource_bees)

    # Have the bees do things based on their states
    inactive_bees.shuffle().be_inactive()
    searching_bees.shuffle().search_random()
    foraging_bees.shuffle().forage()
    returning_bees.shuffle().go_home()
    recruited_search_bees.shuffle().recruited_search()
    recruited_goto_resource_bees.shuffle().recruited_goto_resource()
    approaching_sighted_resource_bees.shuffle().approach_sighted_resource()
```

```
to inactive-unemp
  if (collected != 0) [ user-message "inactive-unemp: collected != 0" ]
  if (mem-goto = "mem") [ user-message "unemployed bee has mem-goto=mem" ]
  if (mem-goto = "" and resource-in-mem != "") [ user-message "unemployed bee with mem-goto='' has resource in memory" ]
  ifelse (mem-goto = "goto")
  [
    if (random (1 / 0.00125) < 1)
    [ set next-state "goto-resource" ]
  ]
  [
    if (random 1000000 <= 165) ; actual map: 1000000 -> 0.000165/tick ;; ADJUST to actual values
    [ set next-state "random-search" ]
  ]
end
```



```python
#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#
##=#=#=#=#=#=#=#=#=#=#=#  STATES  #=#=#=#=#=#=#=#=#=#=#=#=#=#
#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#

def be_inactive(self):
    """
    What the inactive bees do each timestep
    """
    rg = self.model.random  # Set up random number generator, from 0 to 1
    self.time_searching = 0    # Reset time searching counter

    if self.p.p_random_forage > rg.random():
        self.num_trips = 1 + self.num_trips
        self.state = "searching"

    else:
        self.state = "inactive"

 #===========================================================#
```

**Netlogo**: Sam Woodman, 2016

**Python**: Fletcher Nickerson, 2021

# Acknowledgements


Matina Donaldson-Matasci

**Students:**

- Maya Abo Dominguez
- Fletcher Nickerson
- Giovanni Solis
- Annabelle Teng
- Tom Fu
- Arya Massarat
- Kenneth Mitchell
- Berlin Paez
- Sam Woodman

**Advice/help:**

- Jessica Wu
- Alberto Soto
- Calden Wloka
- Drew Price
- Wallace Meyer
- Margaret Couvillon
- Roger Schürch

**Funding:**

- Postdoctoral Program in Interdisciplinary Computation (PIC)



Any
questions?

# ProTaxa: software to easily perform phylogenomic analyses on prokaryotic taxa

Joseph S. Wirth

Eliot Bush

HARVEY MUDD COLLEGE

# Problems with 16s rRNA gene sequences and NCBI Taxonomy

INTERNATIONAL JOURNAL OF SYSTEMATIC BACTERIOLOGY, Jan. 1992, p. 166–170 Vol. 42, No. 1
0020-7713/92/010166-05$02.00/0

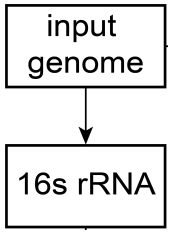

How Close Is Close: 16S rRNA Sequence Identity May Not Be Sufficient To Guarantee Species Identity

GEORGE E. FOX,[1]* JEFFREY D. WISOTZKEY,[2] AND PETER JURTSHUK, JR.[2]
Department of Biochemistry and Biophysical Sciences, University of Houston, Houston, Texas 77204-5934

"An additional, persistent problem remains: dealing with splitting one species into two or more. GenBank records attached to the name will not adjust automatically and will consist of a mixture of the species before and after the split. These can be manually updated, but this is too time consuming to be practical and it rarely happens."

Schoch *et al.*, 2020

# ProTaxa maximizes taxonomic breadth to identify a suitable phylogenetic marker

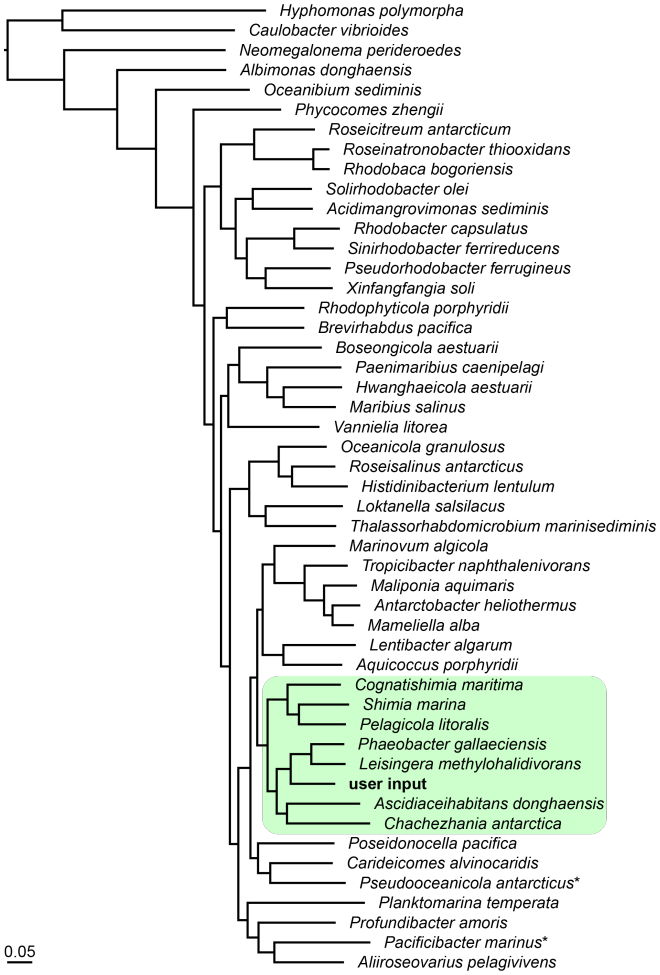



LPSN - List of Prokaryotic names with Standing in Nomenclature

**Founded in 1997 by Jean P. Euzéby.**

https://lpsn.dsmz.de
Wirth and Bush, *in prep*

# Using *Ruegeria pomeroyi* as a test case




| Correlation Coefficient | Gene | Annotation |
|---|---|---|
| 0.987 | *rpoC* | RNA polymerase β' subunit |
| 0.982 | *mfd* | transcription-repair coupling factor |
| 0.982 | *clpA* | Clp protease subunit |
| 0.980 | *dnaE* | DNA polymerase III α subunit |
| 0.980 | *uvrA* | excinuclease ABC subunit |
| 0.979 | *clpB* | ATP-dependent chaperone |
| 0.975 | *recG* | ATP-dependent DNA helicase |
| 0.971 | *gltB* | glutamate synthase large subunit |
| 0.970 | *hrcA* | transcriptional repressor |
| 0.967 | *dnaK* | molecular chaperone |
| 0.967 | *gatB* | amidotransferase subunit |
| 0.951 | *gyrB* | DNA topoisomerase |
| 0.263 | 16S | 16S ribosomal RNA |

Wirth and Bush, *in prep*

# ProTaxa uses the phylogenetic marker to maximize depth of relevant taxonomic space before performing phylogenomic analyses

input genome

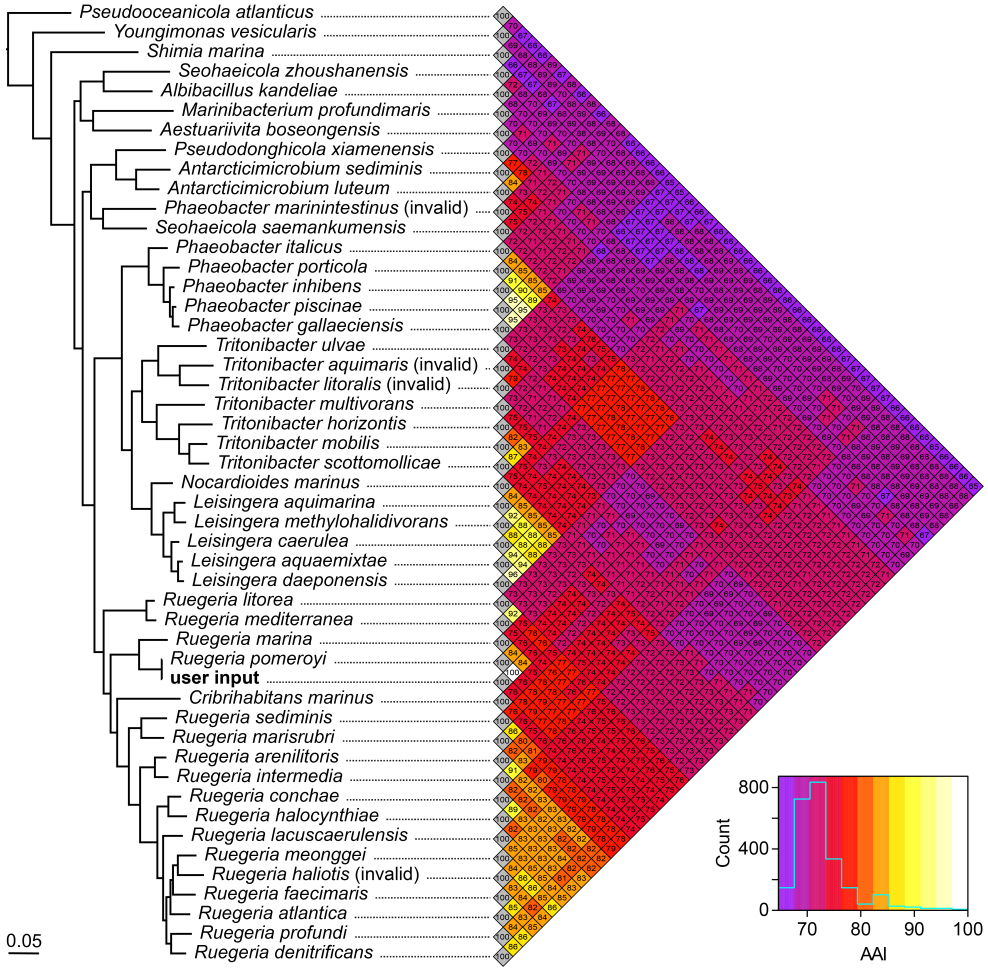
phylogenetic marker

Used to delimit taxonomic ranks

Phylogenomic context

Wirth and Bush, *in prep*

# Using *Ruegeria pomeroyi* as a test case




All whole-genomes are type material

Invalid names are indicated

Genome selection is taxonomically relevant

All data necessary to classify the input have been generated

Wirth and Bush, *in prep*

# Practice problems

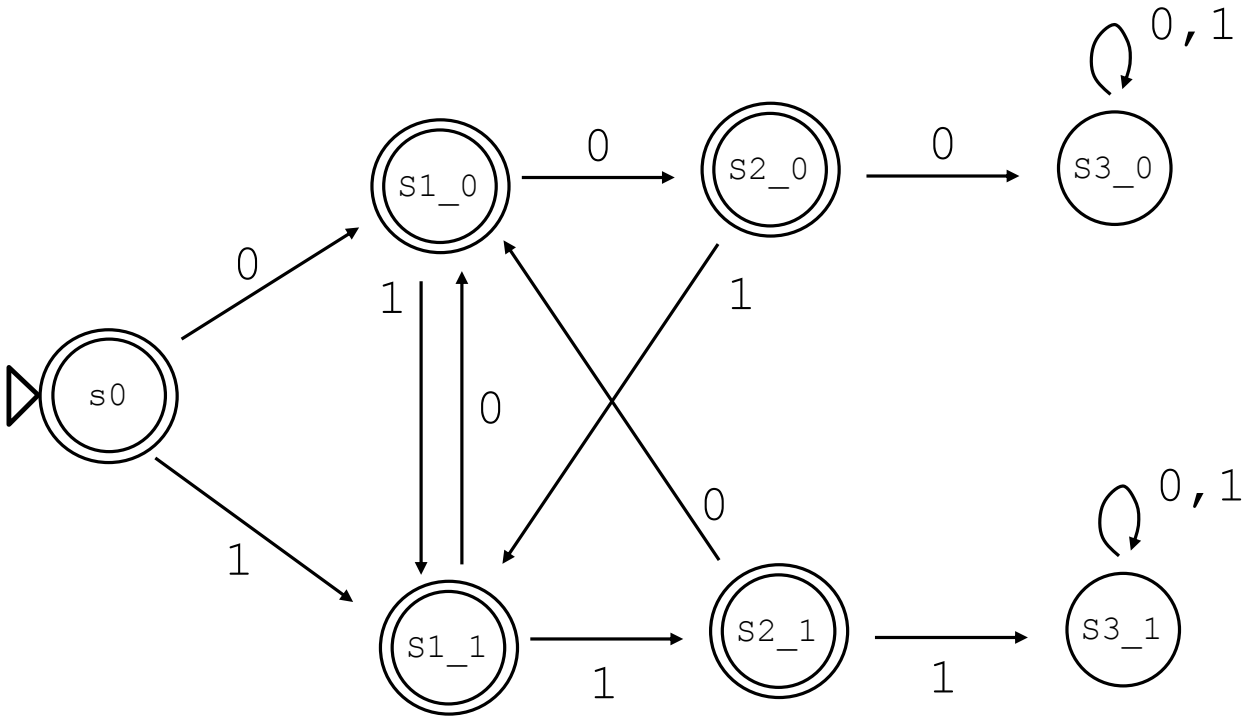
- FSM
- Loops and files
- Strike! and care packages

# FSM

Draw a finite state machine (FSM) that accepts all inputs in which the there are at most two identical consecutive digits. For example, 01010101 should be accepted as should 00110100. But, 10001 and 010111 should be rejected.

# FSM

Draw a finite state machine (FSM) that accepts all inputs in which the there are at most two identical consecutive digits. For example, 01010101 should be accepted as should 00110100. But, 10001 and 010111 should be rejected.



# Loops and files

Contents of testFile.txt:

```
7
Goodbye
```

Reading it into python:

```
>>> f=open("testFile.txt","r")
>>> f.readline()
'7\n'
>>> f.readline()
'Goodbye\n'
>>> f.readline()
''
```

When we get to the end of a file, `f.readline()` returns empty string.

Problem: given a file with an unknown number of lines, write a function to load those lines into a list.

```
def readLines(fileName):
    """Read all the lines in fileName and return as
        a list. (using only the readline() method).
    """
    L = []
    f = open(fileName,'r’)

    f.close()
    return L
```

```
def readLines(fileName):
    """Read all the lines in fileName and return as
        a list. (using only the readline() method).
    """
    L = []
    f = open(fileName,'r’)

    s=f.readline()

    while s != "":

        L.append(s)
        s=f.readline()

    f.close()
    return L
```

# Strike! revisited

“Strike” is a solitaire game played as follows: Given a word (such as “spam”) and a list of words (such as ["ant", "bat", "cat", "dog", "sam"]), find the least number of letters that need to be deleted from our word to get some word in the list (for example, deleting the “p” in “spam” gives us “sam” which is in our list, so the answer is 1 in this case). Here are some examples:

```
>>> strike("dog",  ["ant", "bat", "cat", "dog", "sam"])
0  ← Notice that “dog” is in the list, so no need to strike any letters!

>>> strike("blatz",  ["ant", "bat", "cat", "dog", "sam"])
2 ← We can delete the “l” and “z” from “blatz” to get “bat” which is in our list.

>>> strike("",  ["ant", "bat", "cat", "dog", "sam"])
inf ← Nothing we can do with the empty string to get a word in our list, so we get inf.

>>> strike("blah",  ["ant", "bat", "cat", "dog", "sam"])
inf ← Nothing we can remove from “blah” to get a word in the list so we get inf.
```

Assume that you are *given* a helper function called `helper(letter, wordList)` that works like this:

```
>>> helper('s', ['cat', 'spam', 'dog', 'soup'])
['pam', 'oup']
```

This function throws out the words that don’t start with the letter (“s” in our example) and, for the words that do start with that letter, it chops that first letter off.

```
def helper(letter,L):
    newL=[]
    for word in L:
        if len(word)>0 and word[0]==letter:
            newL.append(word[1:])
    return newL

def strike(word, wordList):
    if word in wordList: return 0
    elif word == "": return float("inf")
    elif wordList == []: return float("inf")
    else:
        loseIt = 1 + strike(word[1:], wordList)
        useIt = strike(word[1:], helper(word[0], wordList))
        return min(useIt, loseIt)
```

Next, write a version called `superStrike(word, wordList)` that returns not just the optimal score, but instead a list of the form `[score, wordsmack]` where `score` is the same score that `strike` would have computed (although `superStrike` should not call `strike`!) and `wordsmack` is the same as `word` but with the letters that should be struck out replaced by "`#`" symbols". Here are some examples:

```
>>> superStrike("dog", ["ant", "bat", "cat", "dog", "sam"])
(0, 'dog')
>>> superStrike("blatz",  ["ant", "bat", "cat", "dog", "sam"])
(2, 'b#at#')
>>> superStrike("",  ["ant", "bat", "cat", "dog", "sam"])
(inf,"")
```

```
def superStrike(word, wordList):
```

```
def superStrike(word, wordList):
    if word in wordList: return (0,word)
    elif word == "": return (float("inf"),"")
    elif wordList == []: return (float("inf"),"")
    else:
        loseIt = superStrike(word[1:], wordList)
        loseIt = (1 + loseIt[0] , "#" + loseIt[1] )

        useIt = superStrike(word[1:], helper(word[0], wordList))
        useIt = (useIt[0] , word[0] + useIt[1])

        if loseIt[0] < useIt[0]:
            return loseIt
        else: return useIt
```

# Looking forward: CS at HMC

*Principles of CS*
CS 60

*SW Engineering & Data structures*
CS 70, 121

*Logic & Computability*
CS 81

*CS for Insight*
CS 35

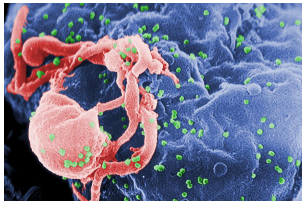
Interfaces — *User Interfaces, Graphics, Animation*
CS 124, 155, 157

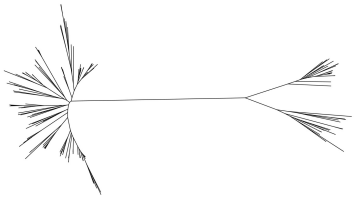
AI — *AI, Neural Networks, Computer Vision, Robotics*
CS 151, 152, 153, 154

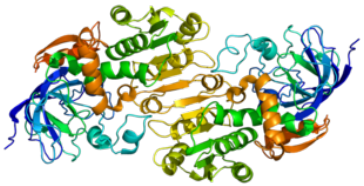
Systems — *Compilers, Programming Languages, Networking, Operating Systems, Computer Architecture*
Eng 85, 155, 158 & CS 125, 131, 132, 134, 136

Theory — *Theory of Computation, Advanced Algorithms*
Math 167, 168 & CS 141, 142

# Looking forward: computational biology at HMC



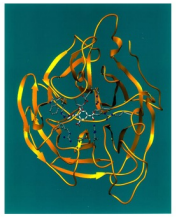


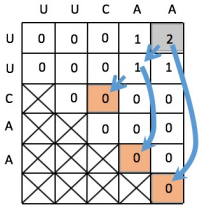


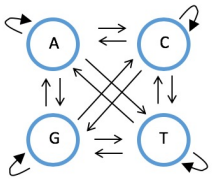
Bio 52

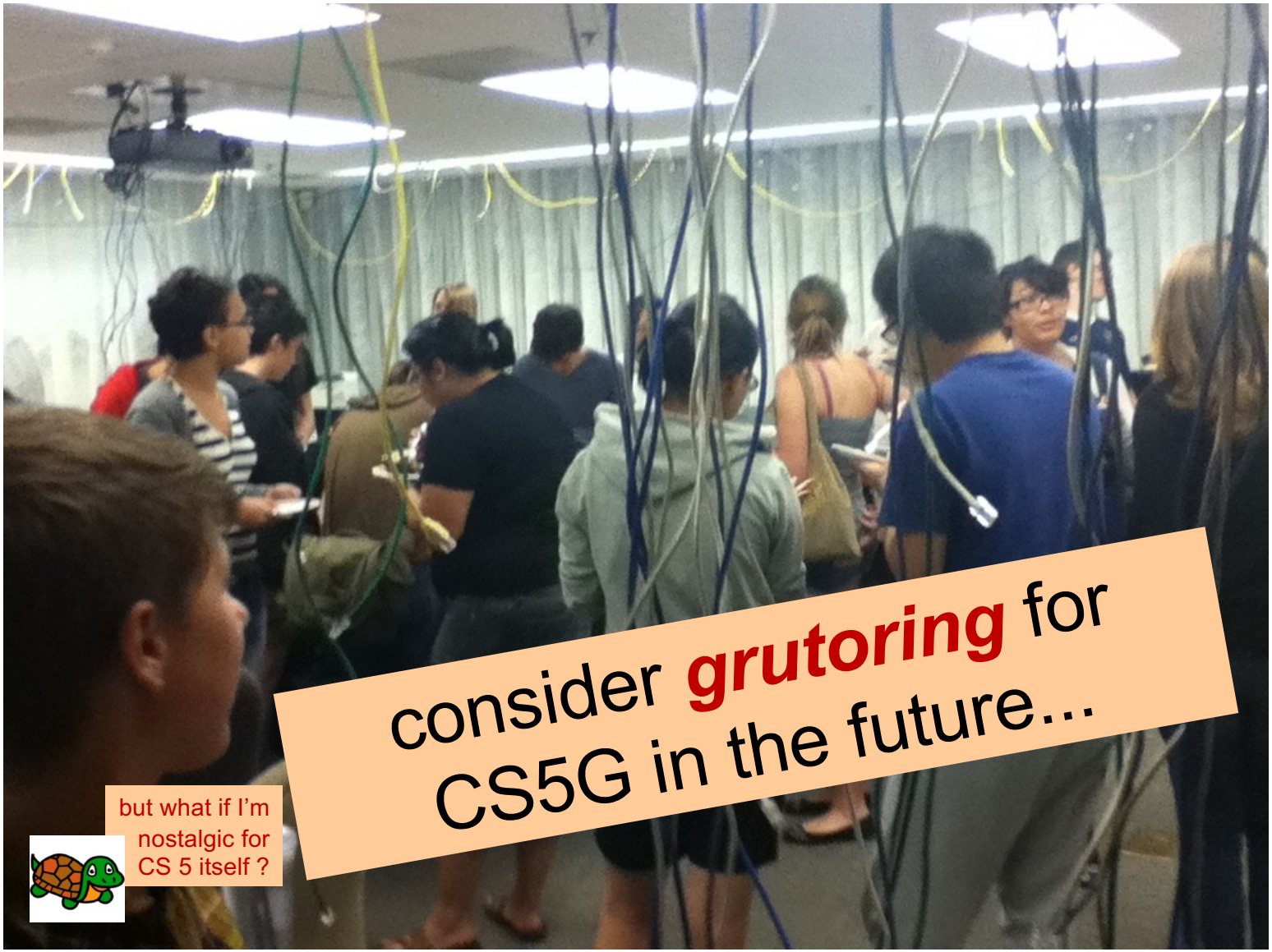
MCBI 118 A and B

Bio 188







consider grutoring for
CS5G in the future…
but what if I'm
nostalgic for
CS 5 itself ?


*Parting thought:*

No matter *what* path you choose,
***it's likely to be in binary***…

Thank you for joining CS5G!

**Good luck on all finals** (projects, exams, papers…)

Final Projects: due ***Friday*** 5pm…

Exam: Tues, **12/14** @ 2pm

Here in Shan 2460