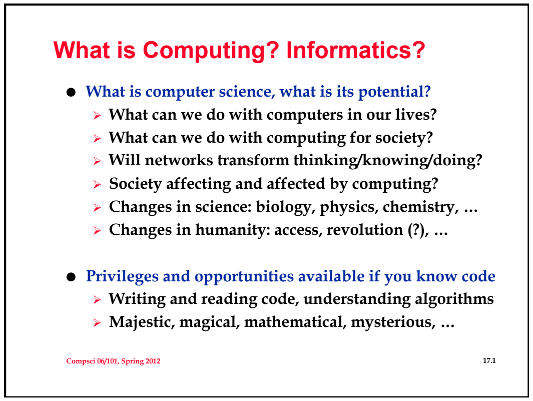
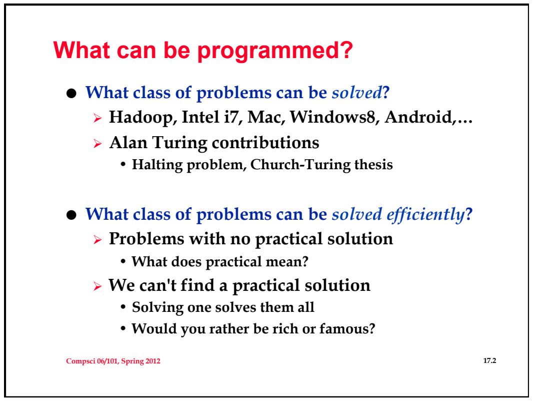


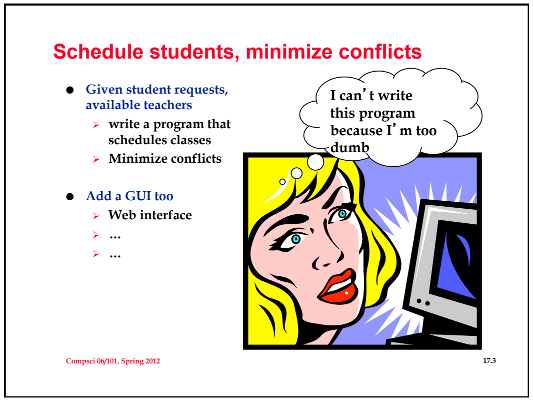
## What is Computing? Informatics?

- What is computer science, what is its potential?
  - What can we do with computers in our lives?
  - What can we do with computing for society?
  - Will networks transform thinking/knowing/doing?
  - Society affecting and affected by computing?
  - Changes in science: biology, physics, chemistry, …
  - Changes in humanity: access, revolution (?), …
- Privileges and opportunities available if you know code
  - Writing and reading code, understanding algorithms
  - Majestic, magical, mathematical, mysterious, …



## What can be programmed?

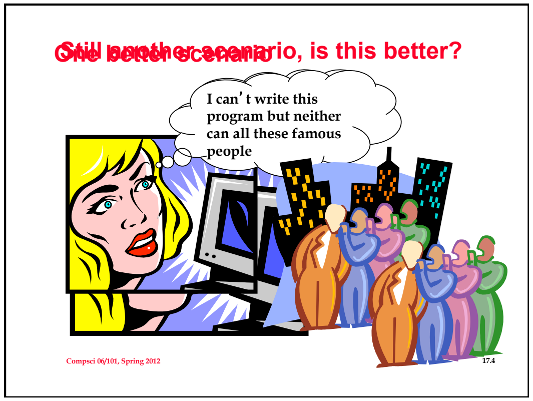
- What class of problems can be *solved*?
  - Hadoop, Intel i7, Mac, Windows8, Android,…
  - Alan Turing contributions
    - Halting problem, Church-Turing thesis
- What class of problems can be *solved efficiently*?
  - Problems with no practical solution
    - What does practical mean?
  - We can't find a practical solution
    - Solving one solves them all
    - Would you rather be rich or famous?



## Schedule students, minimize conflicts

- Given student requests, available teachers
  - write a program that schedules classes
  - Minimize conflicts
- Add a GUI too
  - Web interface
  - …
  - …

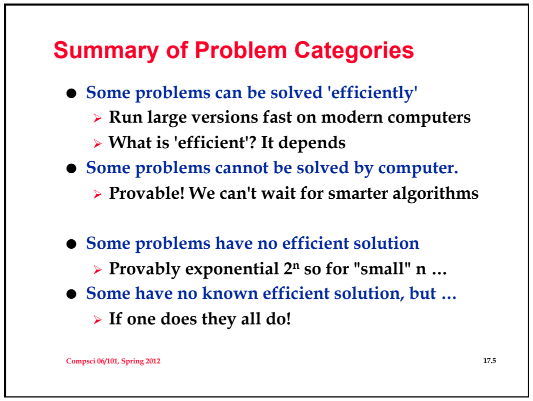
I can't write this program because I'm too dumb



## Still another scenario, is this better?

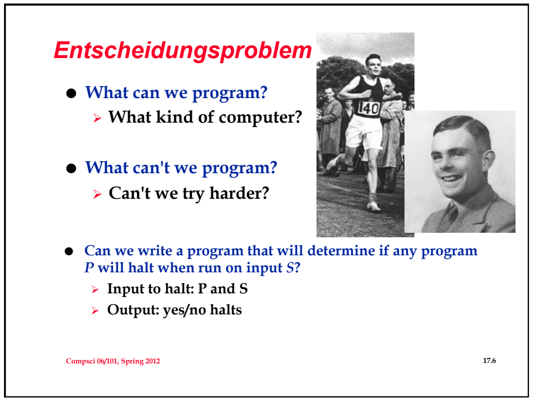
## One better scenario

I can't write this program but neither can all these famous people



## Summary of Problem Categories

- Some problems can be solved 'efficiently'
  - Run large versions fast on modern computers
  - What is 'efficient'? It depends
- Some problems cannot be solved by computer.
  - Provable! We can't wait for smarter algorithms
- Some problems have no efficient solution
  - Provably exponential $2^n$ so for "small" n …
- Some have no known efficient solution, but …
  - If one does they all do!



## *Entscheidungsproblem*

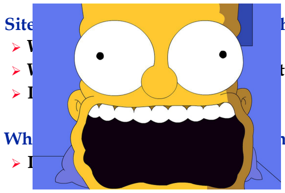
- What can we program?
  - What kind of computer?
- What can't we program?
  - Can't we try harder?
- Can we write a program that will determine if any program *P* will halt when run on input *S*?
  - Input to halt: P and S
  - Output: yes/no halts

## Good sites: http://del.icio.us/

- What is social bookmarking?
  - Why is del.icio.us interesting?
  - Who posts, who visits?
- What about a website of interesting websites?
  - What would you expect to find there?
  - Would the site list itself?
- What about sites that list/link to themselves?
  - What about a site with all sites that list themselves?

## Bad sites: http://haz.ardo.us

- Site hem?)
  - W
  - W tionally?)
  - I
- Wh nselves?
  - I



- Website of all the sites that don’t list themselves?
  - Is `notlisted.com` listed on `notlisted.com`?

## halting module/problem: writing `doesHalt`

```
"""
    function doesHalt returns True if progname
    halts when run on input, and False if progname
    doesn't halt (infinite loop)
"""
    def doesHalt(progname,input):
        #code here

    name = "SpreadingNews.py"
    data = "input.txt"
    if doesHalt(name,data): print "program ended!"
```

- We're assuming `doesHalt` exists – how to use it?
  - It works for any program and any data! Not just one, that's important in this context

## How to tell if X stops/halts on Y

```
import halting
def runHalt():
    prog = "SpreadingNews.py";
    input = "["abc", "def", "hij"]"
    if halting.doesHalt(prog,input):
        print prog,"stops"
    else:
        print prog,"loops 4ever"
```

- Can user enter name of program, X? Input, Y?
  - What's the problem with this program?

## Consider this module *`Confuse.py`*

```
import halting
print "enter name of program",
prog = raw_input()
if halting.doesHalt(prog,prog):
    while True:
        pass
print "finished"
```

- We want to show writing `doesHalt` is impossible
  - Proof by contradiction:
  - Assume possible, show impossible situation results
- Can a program read a program? Itself?

## Are hard problems easy? Clay Prize

## How is Python like all other programming languages, how is it different?

## A Rose by any other name…C or Java?

- Why do we use [Python | Java] in courses ?
  - [is | is not] Object oriented
  - Large collection of libraries
  - Safe for advanced programming and beginners
  - Harder to shoot ourselves in the foot
- Why don't we use C++ (or C)?
  - Standard libraries weak or non-existant (comparatively)
  - Easy to make mistakes when beginning
  - No GUIs, complicated compilation model
  - What about other languages?

## Why do we learn other languages?

- Perl, Python, PHP, Ruby, C, C++, Java, Scheme, ML,
  - Can we do something different in one language?
    - In theory: no; in practice: yes
  - What languages do you know? All of them.
  - In what languages are you fluent? None of them

- In later courses why do we use C or C++?
  - Closer to the machine, understand abstractions at many levels
  - Some problems are better suited to one language

## Find all unique/different words in a file

Across different languages: do these languages have the same power?

## Unique Words in Python

```
#! /usr/bin/env python

def main():
    f = open('/data/melville.txt', 'r')
    words = f.read().strip().split()
    allWords = set()
    for w in words:
        allWords.add(w)
    for word in sorted(allWords):
        print word

if __name__ == "__main__":
    main()
```

## Unique words in Java

```
import java.util.*;
import java.io.*;
public class Unique {
  public static void main(String[] args)
                           throws IOException{
    Scanner scan =
            new Scanner(new File("/data/melville.txt"));
    TreeSet<String> set = new TreeSet<String>();
    while (scan.hasNext()){
        String str = scan.next();
        set.add(str);
    }
    for(String s : set){
        System.out.println(s);
    }
  }
}
```

## Unique words in C++

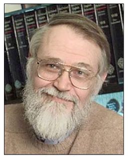
```
#include <iostream>
#include <fstream>
#include <set>
using namespace std;

int main(){
  ifstream input("/data/melville.txt");
  set<string> unique;
  string word;
  while (input >> word){
    unique.insert(word);
  }
  set<string>::iterator it = unique.begin();
  for(; it != unique.end(); it++){
    cout << *it << endl;
  }
  return 0;
}
```

## Unique words in PHP

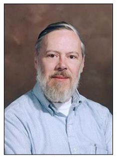
```
<?php

$wholething = file_get_contents("file:///data/melville.txt");
$wholething = trim($wholething);

$array = preg_split("/\s+/",$wholething);
$uni = array_unique($array);
sort($uni);
foreach ($uni as $word){
   echo $word."<br>";
}

?>
```

## Kernighan and Ritchie

- First C book, 1978
- First 'hello world'
- Ritchie: Unix too!
  - Turing award 1983
- Kernighan: tools
  - Strunk and White



Brian Kernighan



Dennis Ritchie

- Everyone knows that debugging is twice as hard as writing a program in the first place. So if you are as clever as you can be when you write it, how will you ever debug it?

Brian Kernighan

## How do we read a file in C?

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int strcompare(const void * a, const void * b){
  char ** stra = (char **) a;
  char ** strb = (char **) b;
  return strcmp(*stra, *strb);
}

int main(){
  FILE * file = fopen("/data/melville.txt","r");
  char buf[1024];
  char ** words = (char **) malloc(5000*sizeof(char **));
  int count = 0;
  int k;
```

## Storing words read when reading in C

```
while (fscanf(file,"%s",buf) != EOF){
  int found = 0;   // look for word just read
  for(k=0; k < count; k++){
    if (strcmp(buf,words[k]) == 0){
      found = 1;
      break;
    }
  }
  if (!found){     // not found, add to list
    words[count] = (char *) malloc(strlen(buf)+1);
    strcpy(words[count],buf);
    count++;
  }
}
```

- Complexity of reading/storing? Allocation of memory?

## Sorting, Printing, Freeing in C

```
  qsort(words,count,sizeof(char *), strcompare);
  for(k=0; k < count; k++) {
    printf("%s\n",words[k]);
  }

  for(k=0; k < count; k++){
    free(words[k]);
  }
  free(words);

}
```

- Sorting, printing, and freeing
  - How to sort? Changing sorting mechanism?
  - Why do we call free? Where required?

```
def is_this_the_end_of_learning_of():
    [x for x in …]
```

## Tim French (Mathemetics)

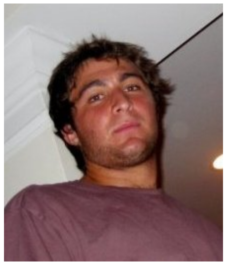
Four FBF in common

## Kristin Oakley (English, Visual/Media)

Three FBF in common

## Graham Oxley (Sociology)

1 FBF in common



## Dmitri Tran (I8N Comparative Studies)

invisible

## Jacquelin Bascetta (Physics)

7 FBF in common

## Chris Kizer (Medieval and Renaissance)

7 FBF in common

## Ubong Akpaninyie

8 FBF in common

## Ryan Magee (Physics)

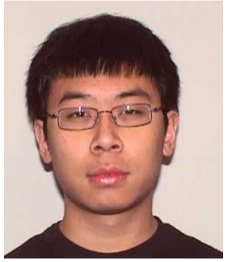
5 FBF in common

## Robby Helms (Physics)

7 FBF in common

## Peter Dong (Chemistry)

6 FBF in common



## Grace Wang (History/Political Science)

invisible