CMSC 337
Algorithms: Design & Practice

CMSC 337
Algorithms: Design & Practice

alt-title

Algorithms: Truth, Beauty & Engineering

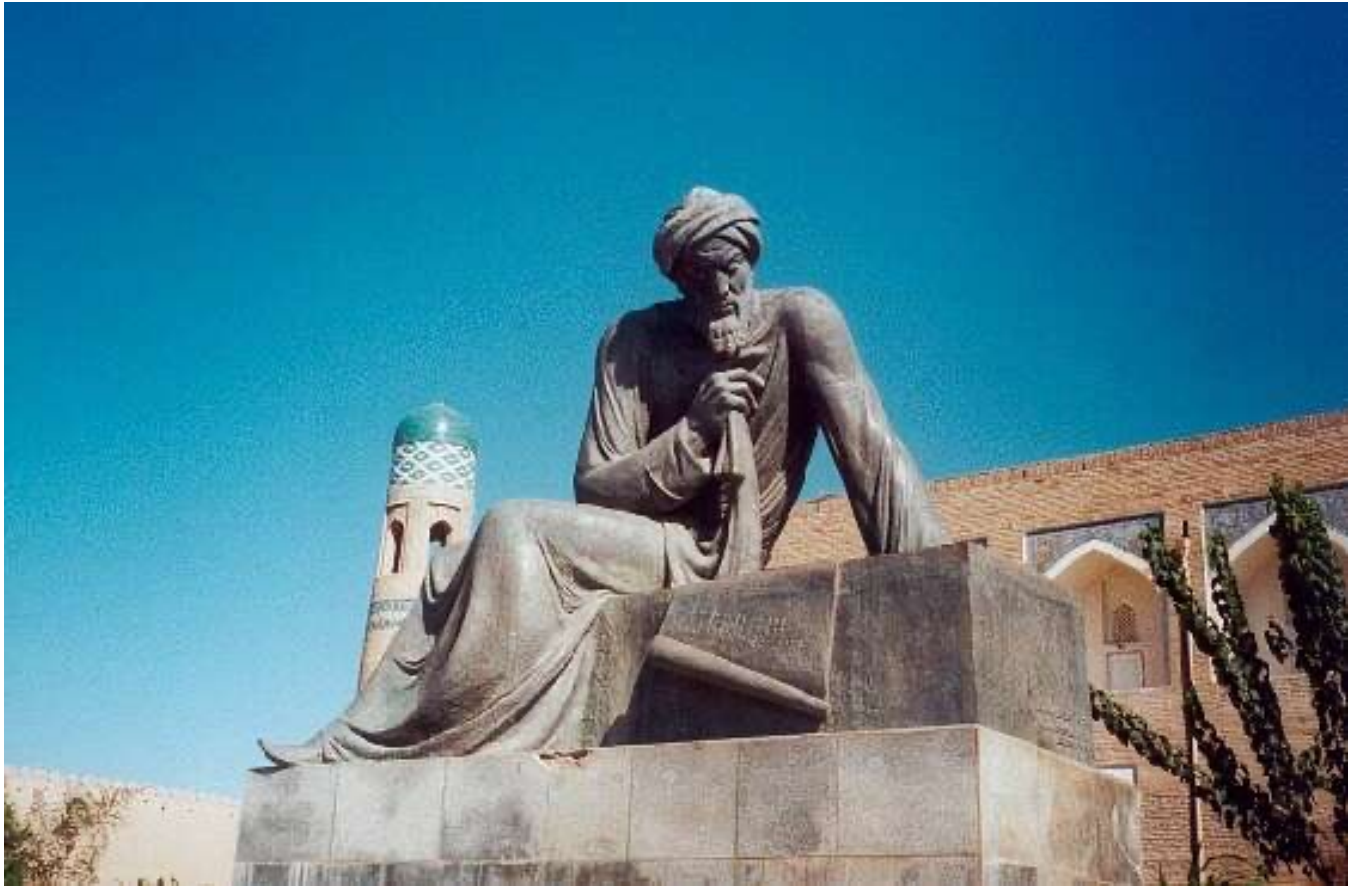
Administrivia

- **Instructor: Geoff Towell**
  259 Park Science Building
  gtowell at brynmawr dot edu

- **Lecture Hours:** M, W 10L10--11:30
  Room: Park 264
  Lab: Wednesday 2:40--4:00 in Park 230
  Office Hours:
  Course Web site **https://cs.brynmawr.edu/cs337**

Algorithms:
Truth, Beauty & Engineering

- **Truth**
  - History
  - Ethics
- **Beauty**
  - Elegance
  - Communication
- **Engineering**
  - Tricks of the trade
  - Eyes open to the world




ПОЧТА СССР
1983
4к
1200
ЛЕТ
Мухаммед
аль-Хорезми

Algorithm

- "A computer algorithm is a set of steps to accomplish a task that is described precisely enough that a computer can run it." (Cormen, pg 1)

Algorithm Desiderata

- "We want two things from a computer algorithm
  - correctness
  - efficiency

(Cormen, pg 2)

Correctness and Efficiency

- Is correctness always required?
  - is it even possible?

- Define "efficiently"

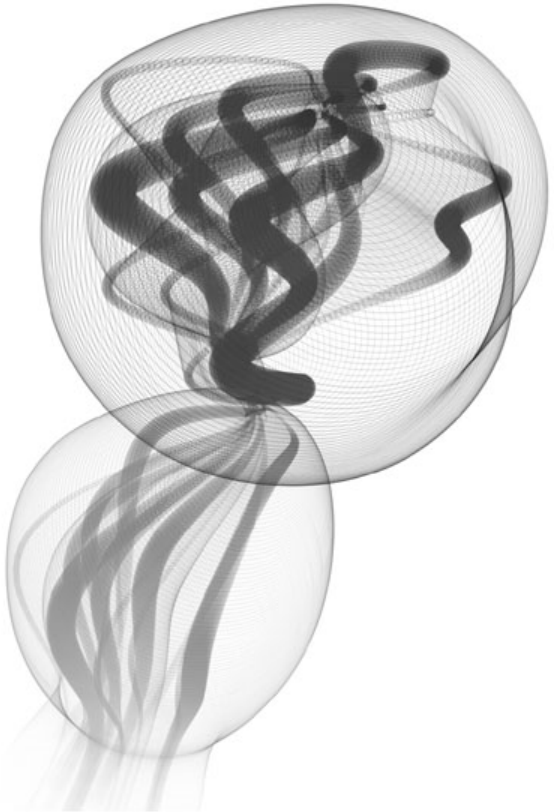
Class Exercise

- Write an algorithm for delivering a piece of paper into a frustrum
  - Constraints:
    - The frustrum is oriented with smaller side down
      - The larger end of the frustrum is open.
    - The deliverer may not get within 2 meters of the frustrum.
    - The approach, when implemented, must have at least a 90% success rate.

Algorithms:
Truth, Beauty & Engineering

- **Truth**
  - History
  - Ethics
- **Beauty**
  - Elegance
  - Communication
- **Engineering**
  - Tricks of the trade
  - Eyes open to the world

Beauty

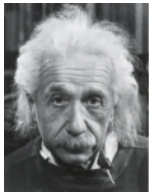


Protobytes
Ira Greenberg
Jellyfish.01
2004

Elegance

Gordon Bell: The cheapest, fastest and most reliable components are those that aren’t there. (Later paraphrased by Musk -- "The best part is no part")

Antoine de Saint Exupéry: A designer knows he has arrived at perfection not when there is no longer anything to add, but when there is no longer anything to take away.



Albert Einstein: Everything should be made as simple as possible, but no simpler.

A Problem

Count the number of occurrences of all characters in a file.

Write an algorithm for this task

A C++ Program

```
// count # occurrences of all characters in a file
// written: 8/5/94, Owen Astrachan, modified 5/1/99


void Print(const tvector<int> & counts, int total);
void Count(istream & input, tvector<int> & counts, int & total);


int main()
{
    int totalAlph = 0;
    string filename = PromptString("enter name of input file: ");
      ifstream input(filename.c_str());


    if (input.fail() )
    {   cout << "could not open file " << filename << endl;
        exit(1);
    }
    tvector<int> charCounts(CHAR_MAX+1,0);    // all initialized to 0

    Count(input,charCounts,totalAlph);
    Print(charCounts,totalAlph);

    return 0;
}


void Count(istream & input, tvector<int> & counts, int & total)
// precondition: input open for reading
//               counts[k] == 0, 0 <= k < CHAR_MAX
// postcondition: counts[k] = # occurrences of character k
//                total = # alphabetic characters
{
    char ch;
    while (input.get(ch))               // read a character
    {   if (isalpha(ch))                // is alphabetic (a-z)?
        {   total++;
        }
        ch = tolower(ch);               // convert to lower case
        counts[ch]++;                   // count all characters
    }
}
```

```
void Print(const tvector<int> & counts, int total)
// precondition: total = total of all entries in
// counts['a']..counts['z']
// postcondition: all counts from 'a' to 'z' printed
{
    const int MIDALPH = 13;
    cout.setf(ios::fixed);    // print 1 decimal place
    cout.precision(1);
    char k;
    for(k = 'a'; k <= 'm'; k++)
    {   cout << k << setw(7) << counts[k] << " ";
        cout << setw(4) << 100 * double(counts[k])/total
             << "% \t\t";
        cout << char(k+MIDALPH) << setw(7)
             << counts[k+MIDALPH] << " ";
        cout << setw(4)
             << 100 * double(counts[k+MIDALPH])/total
             << "%" << endl;
    }
}
```

A Longer Program

Roberts, *The Art and Science of C* (A-W, 1995)

The code is 3 1/2 pages long

Contains these Functions:

```
main, CountLetters, CountLettersInString,
RecordLetter, DisplayLetterCounts, LetterIndex,
ClearIntegerArray
```

Go Version1

```
func main() {
    var m [1000]int
    reader := bufio.NewReader(os.Stdin)
    for {
        rune, n, err := reader.ReadRune()
        if err != nil || n==0 {
           break
        }
        m[rune] = m[rune]+1
    }
    for i,v := range m {
        if (v!=0) {
            fmt.Printf("%c %4d\n", i,v)
        }
    }
}
```

Go Version 2

```
func main() {
    m := make(map[rune]int)
    reader := bufio.NewReader(os.Stdin)
    for {
        rune, n, err := reader.ReadRune()
        if err != nil || n==0 {
            break
        }
        m[rune] = m[rune]+1
    }
    println(m)
}
```

Comparing v1 and v2

- Algorithmically equivalent (?)
- v1 might crash
- v1 is faster
  - 640.260362ms vs 3877.255929ms on an 18M file

Communication

- How to talk about algorithms & computing?
- How to write about it?
- How to do presentations?
- How to exchange ideas?

Strunk & White's Rule 17

Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts. This requires not that the writer make all sentences short or avoid all detail and treat subjects only in outline, but that every word tell.

Algorithms:
Truth, Beauty & Engineering

- **Truth**
  - History
  - Ethics
- **Beauty**
  - Elegance
  - Communication
- **Engineering**
  - Tricks of the trade
  - Eyes open to the world

A Quiz

- A TV Commercial
  - "U.S. college students eat 60 million slices of pizza per month."
  - Is this reasonable?
- How much does a one-hour college lecture cost?
- A program sorts 1 million integers in one second. How long to sort 2 million? 10 Million?
- How long will an exhaustive search take to solve a TSP of size 10? 20? 30?

- Given an array, A of n integers arranged in ascending order, and an integer x:

$$\text{search}(A, n, x) = \begin{cases} i, such\ that\ A[i] = n \\ -1, \qquad otherwise \end{cases}$$