

Sorting: In 2 Slides

- Why do people study sorting?
 - Because we have to
 - Because sorting is beautiful
 - Example of algorithm analysis in a simple, useful setting
- There are n sorting algorithms, how many should we study?
 - $O(n)$, $O(\log n)$, ...
 - Why do we study more than one algorithm?
 - Some are good, some are bad, some are very, very sad
 - Paradigms of trade-offs and algorithmic design
 - Which sorting algorithm is best?
 - Which sort should you call from code you write?

Sorting out sorts

- Simple, $O(n^2)$ sorts --- for sorting n elements
 - Selection sort --- n^2 comparisons, n swaps, easy to code
 - Insertion sort --- n^2 comparisons, n^2 moves, stable, fast
 - Bubble sort --- n^2 everything, slow, slower, and ugly
- Divide and conquer faster sorts: $O(n \log n)$ for n elements
 - Quick sort: fast in practice, $O(n^2)$ worst case
 - Merge sort: good worst case, great for linked lists, uses extra storage for vectors/arrays
- Other sorts:
 - Heap sort, basically priority queue sorting, Big-Oh?
 - Radix sort: doesn't compare keys, uses digits/characters $O(dn+kd)$
 - Shell sort: quasi-insertion, fast in practice, non-recursive $O(n^{1.5})$

Views of programming

- Writing code from the method/function view is pretty similar across languages
 - Organizing methods is different, organizing code is different, not all languages have classes,
 - Loops, arrays, arithmetic, ...
- Program using abstractions and high level concepts
 - Do we need to understand 32-bit twos-complement storage to understand $x = x+1$?
 - Do we need to understand how arrays map to contiguous memory to use ArrayLists?

Bottom up meets Top down

- We teach programming by teaching abstractions
 - What's an array? What's an ArrayList?
 - What's an array in C?
 - What's a map, Hashmap? Treemap?
- How are programs stored and executed in the JVM?
 - Differences on different architectures?
 - Similarities between Java and C++ and Python and PHP?
- What about how the CPU works? How memory works? Shouldn't we study this too?

From bit to byte to char to int to long

- Ultimately everything is stored as either a 0 or 1
 - Bit is **binary digit** a byte is a **binary term** (8 bits)
 - We should be grateful we can deal with Strings rather than sequences of 0's and 1's.
 - We should be grateful we can deal with an int rather than the 32 bits that comprise an int
- If we have 255 values for R, G, B, how can we pack this into an int?
 - Why should we care, can't we use one int per color?
 - How do we do the packing and unpacking?

Signed, unsigned, and why we care

- Some applications require attention to memory-use
 - Differences: one-million bytes, chars, and int
 - First requires a megabyte, last requires four megabytes
 - When do we care about these differences?
 - Memory is cheaper, faster, ... But applications expand to use it
- Java byte, int, long are signed values, char unsigned
 - Signed values represented in twos-complement
- Java signed byte: -128..127, # bits?
 - What if we only want 0-255? (Huff, pixels, ...)
 - Convert negative values or use char, trade-offs?
- Java char unsigned: 0..65,536 # bits?
 - Why is char unsigned? Why not as in C++/C?

More details about bits

- How is 13 represented?
 - ... $\frac{0}{2^4}$ $\frac{0}{2^3}$ $\frac{1}{2^2}$ $\frac{1}{2^1}$ $\frac{0}{2^0}$ $\frac{1}{2^0}$
 - Total is $8+4+1 = 13$
- What is bit representation of 32? Of 15? Of 1023?
 - What is bit-representation of $2^n - 1$?
 - What is bit-representation of 0? Of -1?
 - Study later, but -1 is all 1's, left-most bit determines < 0
- Determining what bits are on? How many on?
 - Understanding, problem-solving

How are data stored?

- To facilitate Huffman coding we need to read/write one bit
 - Why do we need to read one bit?
 - Why do we need to write one bit?
 - When do we read 8 bits at a time? 32 bits?
- We can't actually write one bit-at-a-time. We can't really write one char at a time either.
 - Output and input are buffered, minimize memory accesses and disk accesses
 - Why do we care about this when we talk about data structures and algorithms?
 - Where does data come from?

Representing pixels

- A pixel typically stores RGB and alpha/transparency values

- Each RGB is a value in the range 0 to 255
- The alpha value is also in range 0 to 255

```
Pixel red = new Pixel(255,0,0,0);  
Pixel white = new Pixel(255,255,255,0);
```

- Typically store these values as int values, a picture is simply an array of int values

```
void process(int pixel){  
    int blue = pixel & 0xff;  
    int green = (pixel >> 8) & 0xff;  
    int red = (pixel >> 16) & 0xff;  
}
```

Bit masks and shifts

```
void process(int pixel){  
    int blue = pixel & 0xff;  
    int green = (pixel >> 8) & 0xff;  
    int red = (pixel >> 16) & 0xff;  
}
```

- Hexadecimal number: 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
 - f is 15, in binary this is 1111, one less than 10000
 - The hex number 0xff is an 8 bit number, all ones
- Bitwise & operator creates an 8 bit value, 0–255
 - Must use an int/char, what happens with byte?
 - 1&1 == 1, otherwise we get 0 like *logical and*
 - Similarly we have |, bitwise or

A Rose by any other name...C or Java?

- Why do we use Java in our courses (royal we?)
 - 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-existent (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?
 - Depends on what different means.
 - 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, we want to understand the machine at many levels, from the abstract to the ridiculous
 - Or at all levels of hardware and software
 - Some problems are better suited to one language
 - What about writing an operating system? Linux?

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);
        }
    }
}
```

Bjarne Stroustrup, Designer of C++

- Numerous awards, engineering and science
 - ACM Grace Hopper
- Formerly at Bell Labs
 - Now Texas A&M



- "There's an old story about the person who wished his computer was as easy to use as his telephone. That wish has come true, since I no longer know how to use my telephone."

Bjarne Stroustrup

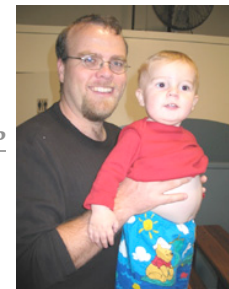
Unique words in C++

```
#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;
}
```

PHP, Rasmus Lerdorf and Others

- Rasmus Lerdorf
 - Qeqertarsuaq, Greenland
 - 1995 started PHP, now part of it
 - <http://en.wikipedia.org/wiki/PHP>
- Personal Home Page
 - No longer an acronym



- "When the world becomes standard, I will start caring about standards."

Rasmus Lerdorf

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>";
}
?>
```

Guido van Rossum



- **BDFL for Python development**
 - Benevolent Dictator For Life
 - Late 80's began development
- **Python is multi-paradigm**
 - OO, Functional, Structured, ...
- We're looking forward to a future where every computer user will be able to "open the hood" of their computer and make improvements to the applications inside. We believe that this will eventually change the nature of software and software development tools fundamentally.

Guido van Rossum, 1999!

Unique Words in Python

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

import sys
import re

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

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

Kernighan and Ritchie



- **First C book, 1978**
- **First 'hello world'**
- **Ritchie: Unix too!**
 - Turing award 1983
- **Kernighan: tools**
 - Strunk and White
- 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 strcmpare(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 *), strcmpare);
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? What's analogous to comparator?
    > Why do we call free? Necessary in this program? Why?
```

The exam

- Thursday, December 10, 9am-12pm in classroom
- Open book/open note
- ~40% multiple choice/short answer
- Cumulative
- By Monday, December 7:
 - > All grades up (except huff)
 - > Test solutions out
 - > Grade problems:
Submit assignment *issues*
- Final grades up Saturday, December 12
- Available by appointment throughout reading period and exam week
- Help session Friday in lab (optional)