Announcements

- HTML/XML homework due on Thursday before class
- Change of Plan: Will be here regular class time in regular place on Thursday
- Understand if people have troubling getting onto campus but turn in your homework electronically
- Class lecture will be electronically available.
And now for something completely different!!

Computability and correctness of a computer program.
Thinking About Computing

- Computers do things quickly ... except when they don’t
  - Usually we don’t know why
  - It could just be congestion on the internet
  - Or, when saving large files, like movies, we’re waiting for the hard disk to copy everything

- Describing Computability
  - *Time proportional to* $X$

- Sometimes the time a computer takes is *linearly* proportional to how much data there is ...
  - Looking for something in a list: could go through the list from beginning to end, looking at each item
  - A movie file you have to save each scene
Linear = Time Proportional To N

Each item in a list that is N items long

- Problems whose work (computation time) is proportional to n are called n-time or linear time problems
  - Making an image lighter in your photo software
  - Adding a column of numbers in a spreadsheet
  - Crawling the Internet looking for links
  - ... many more ... linear problems are common

- Some problems are not ...
Algorithms

- Def. An algorithm is a precise, systematic process for an agent to produce a specified result
- Programs are algorithms
- Five properties characterize algorithms
  - **Input specified** – tell form and amount of input required
  - **Output specified** – tell form and amount of output produced
  - **Definiteness** – say explicitly what to do & in what order
  - **Effectiveness** – operations within agent’s abilities
  - **Finiteness** – will stop and give an answer or say “none”
People try to design algorithms e.g. to sort things, to run quickly
Linear = Time Proportional To N

Each item in a list that is N items long

- Problems whose work (computation time) is proportional to n are called n-time or linear time problems
  - Making an image lighter in your photo software
  - Adding a column of numbers in a spreadsheet
  - Crawling the Internet looking for links
  - ... many more ... linear problems are common

- Some problems are not ... 
Linear Search

- Pick a number between 1 and 1000. How many guesses will I need?

- What about Google’s index with 25 million entries?

- The search happens in memory, but the list of URLs associated with the “term” will likely be on disk.
Binary Search

- Pick a number between 1 and 1000. How many guesses will I need?

- Web Search: What about that index with 25 million entries?

- The search happens in memory, but the list of URLs associated with the “term” will likely be on disk.
Sorting

- Putting a sequence of items into alphabetical or numerical order
- First: let’s try exchanging them as we go along (bubbling them along)
  
  **walrus seal whale gull clam**

- Algorithm: compare to all following items, reorder if needed
- Other ways to sort we will talk about in a minute
Sorting Algorithm: Bubble Sort

Bubblesort (int data[], int n) {
    int tmp, i, j;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (data[j] > data[j+1]) {
                tmp = data[j];
                data[j] = data[j+1];
                data[j+1] = tmp;
            }
        }
    }
}
How Long To Sort w/ Bubble Sort?

- The pattern is, for n items
  n-1 focus on first item in the list
  n-2 focus on second item
  n-3 focus on third item
  ...
  1 on next to last
- n-1 rows in list, we just want a good estimate
- average of each row n/2,
- so (n-1) times (each row)
- Multiply by average = n/2
  = (n^2 − n)/2
- Computing time = Time proportional to n^2

```c
Bubblesort (int data[], int n) {
    int tmp, i, j;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++)
            if (data[j] > data[j+1]) {
                tmp = data[j];
                data[j] = data[j+1];
                data[j+1] = tmp;
            }
    }
}
```
Time proportional to $n^2 = \text{Polynomial time}$

- Other computations have running time
  - proportional to $n^3$ – matrix multiplication
  - proportional to $n^4$
  - ...

- All of them are lumped together as “polynomial time computations”
  - Considered to be realistic ... a person can wait
  - Polynomial, but not linear
  - Many algorithms you would learn in 12B, 101
Many Alternative Algorithms …

- There are always different algorithms to pick
- Consider sorting …
  - We just discussed
  - Bubble Sort for
    - walrus
    - seal
    - whale
    - gull
    - clam
  - Compare with every value following

```
wa se wh g   c
se wa wh g   c
se wa wh g   c
g wa wh se c
```
There are Different Algorithms

- Is there a better way to do sorting?

**QUICKSORT**

- **Fastest** known sorting algorithm in practice
- Average case: $O(N \log N)$ (we don’t prove it)
- Worst case: $O(N^2)$
  - But, the worst case seldom happens.
- A divide-and-conquer recursive algorithm
Quick-sort with Hungarian (Küküllőmenti legényes) folk dance

facebook.com/AlgoRythmics

Intercultural Computer Science Education

Created at Sapientia University (in cooperation with "Maros Művészegyüttes")
Different sorting algorithms and animations

Quicksort is the best: Divide and Conquer

- **Divide step:**
  - Pick any element (*pivot*) \( v \) in \( S \)
  - Partition \( S - \{v\} \) into two disjoint groups
    \[
    S_1 = \{ x \in S - \{v\} \mid x \leq v \} \\
    S_2 = \{ x \in S - \{v\} \mid x \geq v \}
    \]

- **Conquer step:** recursively sort \( S_1 \) and \( S_2 \)

- **Combine step:** the sorted \( S_1 \) (by the time returned from recursion), followed by \( v \), followed by the sorted \( S_2 \) (i.e., nothing extra needs to be done)

To simplify, we may assume that we don’t have repetitive elements, So to ignore the ‘equality’ case!
Example

select pivot

partition
Sorting Algorithm: Quicksort

Quicksort (int data[], int left, int right)
    int mid, tmp, i, j;
    i = left;
    j = right;
    mid = data[(left + right)/2];
    do{
        while(data[i] < mid)
            i++;
        while(mid < data[j])
            j--;
        if (i <= j){
            tmp = data[i];
            data[i] = data[j];
            data[j] = tmp;
            i++;
            j--;
        }
    } while (i <= j);
    if (left < j) Quicksort(data, left, j);
    if (i < right) Quicksort(data, i, right);
Different sorting algorithms and animations

This one is the parameterizable web page that has the same code and you can give different parameters for number of things to sort, speed of animation etc.


- This one is one of the ‘dancing animations’ we looked at both bubble sort and quick sort. This shows the “recursion” in a better way

- Hungarian Folk Dance Quick Sort ttp://www.youtube.com/watch?v=ywWBBy6J5gz8
Intracultural Computer Science!

Quick-sort with Hungarian (Küküllőmenti legényes) folk dance

Created at Sapientia University (in cooperation with "Meres Művészegyüttes")
Also CORRECTNESS!

- How do we know that the algorithms work?
  - Developing algorithms is not just thinking them up
  - It is also reasoning through why they work ... you need to know why explicitly enough to tell someone else

- What is involved in doing that?
  - Usually some kind of “proof”,
    - by considering different possible cases
    - By imagining it didn’t work and proving a contradiction
Why Does Bubble Sort Work?

Why do you think it sorts?

wa se wh g  c
se wa wh g  c
se wa wh g  c
g wa wh se  c
c wa wh se  g
c wa wh se  g
c wa wh se  g
c se wh wa g
c g wh wa se
c g wa wh se
c g se wh wa
c g se wa wh
Why Does Bubble Sort Work?

- Why do you think it sorts?
  - There are several passes through the data with *leading item* fixed (marked with lines)
- Notice this property: After each pass, the *leading item* must be the smallest of all processed on the pass
- The leading item is first the first item, then the second etc.
- Proof by induction
Summary: Proving Correctness

- It is not sufficient to think up a clever algorithm ... you need to know why it works
- It's usually not tough, because the logic of your method typically translates into an explanation of why it works.
- This is what you are often doing when you are trying to fix your program
To Infinity And Beyond

- There are more complex computations ...
  - Suppose you want to visit 28 cities in the US (for a rock concert?) and you want to minimize your how much you pay for airplane tickets
  - You could select an ordering of cities (SEA → PDX → SFO → LAX ...) and compute the ticket price.
  - Then pick another ordering (SEA → SFO → LAX → PDX ... ), compute this ticket price and compare to the previous one
  - Always keep the cheapest itinerary

- This seems very dumb ... is there a better way?
Traveling Salesman Problem

- Actually, no one knows a way to solve this problem significantly faster than checking all routes and picking the cheapest ...
- Not polynomial time ... so we are guessing that there is no polynomial solution (Non Polynomial = NP)
- This is what is called *an NP-Complete problem*
  - Many many related problems ... the best solution is “generate and check”
    - Best way to pack a container ship
    - Most efficient scheduling for high school students’ classes
    - Least fuel to deliver UPS packages in Washington
    - Fewest public alert broadcast stations for US
Astonishing Fact

- Although there are thousands of NP-Hard Problems, meaning they’re basically “generate and check” ...
- NP-Complete computations (like traveling salesman) have the property that if any one of them can be done fast (n^x-time, say) then EVERY SINGLE ONE of the related problems can be too!

- Is Traveling Salesman solvable in n^x time?
- One of the great open questions in computer science

Be Famous ... Answer This Question
There’s Stuff A Computer Can’t Do

- Some problems are too big – combinatorial explosive – like checking each chess game to see if there is a guaranteed win for White
  - Too many items to check
  - Doable in principle, however

- The Halting Problem:
  - Will this program stop running at some point
  - Cannot be solved.
  - Would learn about in CMPS 130 if you take that
Many computations have *time proportional to* $n$

Many, like sort, have running *time proportional to* $n^2$

Others have running time proportional to $n^3, n^4, \ldots$

Some computations are computable in principle but not in practice: *NP-complete*

Some things cannot be computed at all, such as the *Halting Problem*
Needles in the Haystack: Google and Other Brokers in the Bits Bazaar

Blown To Bits Chapter 4
Search for “travel”
How many different pages?

- Searching for “travel” in Google today yielded “About 3,770,000,000 results (0.33 seconds)”
- At least 40-50 billion pages
  (www.worldwidewebssize.com)
Important questions: What you should learn

- How can a search engine respond so fast?
- Does it find every relevant link?
- How does a search engine decide what gets listed first?
- If you try another search engine will you get the same result? If so, which is right? Which is better? Which is more authoritative?
- Are sponsored links better than “organic” links? Is the advertising necessary?
- What is the role of government? What should it be?
“There is no practical obstacle whatever now to the creation of an efficient index to all human knowledge, ideas and achievements, to the creation, that is, of a complete planetary memory for all mankind… The whole human memory can be, and probably in a short time will be, made accessible to every individual… This is no remote dream, no fantasy.”
As we may think: Vannevar Bush 1940’s

“The difficulty seems to be, not so much that we publish unduly … but rather that publication has been extended far beyond our present ability to make real use of the record. The summation of human experiences is being expanded at a prodigious rate, and the means we use for threading through the consequent maze to the momentarily important item is the same as was used in the days of square-rigged ships. … Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing.”
In the beginning

- Early catalogues both on the Internet (even before the WWW) and in print.
- “The Whole Internet Catalog” (1994)
- Yahoo was like a library index compiled by human editors.
- Search engines were invented in early 90s with the growing popularity of the web.
Sample Page: Agriculture

- Look up by topic, like a library index
- Where to go
The Library vs the Information Bazaar

- “Yellow pages”, directories, and catalogues
- The “Web” is not hierarchical
  - no structure like a library
- Catalogues are out - search engines are in.
- Why?
How many hosts are out there?

Internet Domain Survey Host Count

Source: Internet Systems Consortium (www.isc.org)
Big data is here
The Library vs the Information Bazaar

- “Yellow pages”, directories, and catalogues
- The “Web” is not hierarchical
  - no structure like a library
- Catalogues are out - search engines are in.
- But - search engines control what you see
“The search tools that help us find needles in the digital haystack have become the lenses through which we view the digital landscape. Businesses and governments use them to distort our picture of reality.”

Blown to Bits - pg 110
"For the user, search is the power to find things, and for whoever controls the engine, search is the power to shape what you see."

Blown to Bits pg 112
Web search: It Matters How It Works

1. Gather information.
2. Keep copies.
3. Build an index.
4. Understand the query.
5. Determine the relevance of each possible result to the query.
6. Determine the ranking of the relevant results.
7. Present the results.
Its all free?? : Well no. Who Pays for What?

- Users could pay a subscription fee (early AOL and CompuServe)
- Web sites could pay for being indexed.
- The government could pay (taxes?).
- Advertisers could pay.

- And it matters who pays cause it affects how it works
Page Rank Algorithms

- The “crown jewels” of search engines lie in their page rank algorithms.

- Factors include:
  - keywords in heading or titles
  - keyword only in the body text
  - site is “trustworthy”
  - links on this page are to relevant pages
  - links to this page are relevant
  - age of the page
  - quality of the text (e.g. absence of misspellings)
Web search: It Matters How It Works

1. Gather information.
2. Keep copies.
3. Build an index.
4. Understand the query.
5. Determine the relevance of each possible result to the query.
6. Determine the ranking of the relevant results.
7. Present the results.
Search Engines

No one controls what’s published on the WWW ... it is totally decentralized

To find out, \textit{search engines crawl} Web

- Two parts
  - \textit{Crawler} visits Web pages building an \textit{index} of the content (stored in a database)
  - \textit{Query processor} checks user requests against the index, reports on known pages [You use this!]

Only a fraction of the Web’s content is crawled
1. Gather Information

- Spiders or web crawlers wander the web building indices.
- Estimates range from .02% to 3% of information is indexed.
- How often does a page get visited?
  - some frequently (daily whitehouse.gov), others rarely
  - Crawler keeps track of which pages change frequently.
- How does the crawler find its way and not go in circles?
- Logins keep bots/crawlers out.
How to crawl the Web:

- Begin with some Web sites, entered “manually”
- Select page not yet crawled; look at its HTML
  - For each keyword, associate it with this page’s URL
  - Harvest words from URL and inside <title> tags ...
  - For every link tag on the page, associate the URL with the words inside of the anchor text, that is,
- Save all links and add to list to be crawled
Net Result From Crawling A Page

- After crawling a page like users.soe.ucsc.edu/~maw
- the crawler will associate many terms with the URL: reinforcement learning,
- Terms from URL and anchor are more important in describing the page
Net Result of Crawling All Pages

- When the crawling is “done” (it’s never done), the result is an index, a special data structure that a query processor can use to look up your queries:
  
  **Free:** ..., www.cs.washington.edu/cse120/freeProgramming.html, ...

  **Programming:** ..., www.cs.washington.edu/cse120/freeProgramming.html, ...

  **Picasso:** ..., www.cs.washington.edu/cse120/freeProgramming.html, ...

3/1/12 © 2011 Larry Snyder, CSE
2. Keep Copies

- Spider downloads the page as part of the “visit” in order to create the index.
- Search engine may “cache” the copy.
- Is this legal? What about copyright?
- But wait, browsing requires copying as well.

“(AFP) – Sep 15, 2011

NEW YORK — Google and publishers told a US judge Thursday they are close to settling a lawsuit over the Internet giant's controversial book-scanning project…”
3. Build an Index

- list of terms and for each term a list of where it appeared
- more than just the terms
  - terms in bigger font might be more important
  - terms in the title might be more important
- must be very fast to lookup
- could be millions of entries (not just words, but names, special numbers, etc.) requiring Gigabytes of memory
- must fit in the computers memory (see next slide)
Accessing RAM vs Disk

- How much longer does it take to access a random location in your computer's RAM memory compared to accessing a random location on the disk?

Answer in terms of “n times longer” like twice as long or 10 times as long.
Binary Search

- Pick a number between 1 and 1000. How many guesses will I need?

- What about that index with 25 million entries?

- The search happens in memory, but the list of URLs associated with the “term” will likely be on disk.
With 8 bits how many different letters?
\[2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256\]
4. Understand the Query

- Steps 1-3 happen in “the background”
- Not much “understanding” in today’s search engines but that could change soon.
- Advanced search engine features help

Cardinal’s beat Rangers
Vs.
Ranger’s beat Cardinal’s

What about a business called “THE”? 
Make A Query

- When Google gets the query

  ![Google Search](image)

  - It “ands” the two lists together, finding URLs that are on both lists
  - It counts them up, records time, shows 10 hits
5. Determine Relevance

- “Recall” - what percentage of relevant documents are returned by the search?
- Simple relevance calculation -
  - count the number of times each search word appears in the document, add them all up
- Long documents get higher scores.
- Uninteresting words like “the” contribute to the score.
- All word occurrences are not equal (title words should count more).
6. Determine Ranking

- Which of the relevant documents should be displayed first?
- Simple solution - put one with highest relevance score first.
- What if many have the same score?
- Are ones with the highest relevance score really the most important? What about the source of the document (e.g. NY Times vs some random blog post).
The “crown jewels” of search engines lie in their page rank algorithms.

Factors include:

- keywords in heading or titles
- keyword only in the body text
- site is “trustworthy”
- links on this page are to relevant pages
- links to this page are relevant
- age of the page
- quality of the text (e.g. absence of misspellings)
You want the most likely hits ... how does Google show you what you want?

Page Rank – a mechanism to estimate the “importance” of a page; pages are listed by page rank, highest to lowest
Page Rank

- Google has never revealed all details of the ranking algorithm, but we know ...
  - URL’s are ranked higher for words that occur in the URL and in anchors
  - URL’s get ranked higher if more pages point to them, it’s like: A links to B is a vote by A for B
  - URL’s get ranked higher if the pages that point to them are ranked higher
Professor Marilyn Walker  
Department of Computer Science  
University of California Santa Cruz  
1156 N. High, SOE-3  
Santa Cruz, Ca. 95064  
Phone:831 429 1058  
Natural Language and Dialogue Systems Lab  
mailto:maw@soe.ucsc.edu

DATA and SOFTWARE DOWNLOADS

PROFESSIONAL EXPERIENCE
Professor of Computer Science, Natural Language and Dialogue Systems Lab, University of California, Santa Cruz, 2009 to present

Professor of Computer Science, Head of Cognitive Systems Group, University of Sheffield, Sheffield, England, 2003 to 2009

Principal Research Staff Member, ATT Labs - Research, Florham Park, N.J., Speech Processing Software and Technology Research, 1996 to 2003

Research Scientist, Mitsubishi Electric Research Laboratories, Cambridge, Ma., Interactive Learning and Entertainment, 1993 to 1996

Consultant, Hewlett Packard Laboratories, Bristol, England, on dialogue systems, speech technology, and personal information systems: 1989-1993

Researcher, Dialogue Modeling Department, Electrotechnical Laboratory, Tsukuba City, Japan: Summer 1991


EDUCATION


B.A. Computer and Information Science, With Honors, University of California Santa Cruz, 1984.
Google Query Algorithm

- In a first paper Larry Page and Sergey Brin gave their algorithm for processing a Google query

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.
8. Sort the documents that have matched by rank and return the top k.

Figure 4. Google Query Evaluation

- This algorithm is understandable to readers
Google Short Index & Long Index

- Known as the short barrels and the long barrels..

- **Short index:**
  - store the words in link texts that point to a page (inbound links!!)
  - the words in a page’s title, and one or two other special things.

- The link text words are attributed to the target page, and not to the page that the link is on.
  - In other words, if my page links to your page, using the link text “Miami hotels”, then the words “Miami” and “hotels” are stored in the short index as though they appeared in your page, but they belong to my page. If 100 pages link to your page, using those same words as link text, then your page will have a lot of entries in the short index for those particular words.

- The long index is used to store all the other words on a page – its actual content.
Processing a query: short and long indices

- First try to get enough results from the short index.
- If you can’t get enough results, then use the long index to add to what they have.
  - It means that, if they can get enough results from the short index – that’s the index that contains words in link texts and page titles – then they don’t even look in the long index where the actual contents of pages are stored. Page content isn’t even considered if they can get enough results from the link texts and titles index – the short index.
- Thus: link texts are very powerful for Google rankings.
  - Much more powerful than page titles, because a page can have the words from only one title in the short index, but it can have the words from a great many link texts in there.
  - Page titles and meta descriptions were the second most powerful ranking factors, because they are stored in the short index.
What does Google use to search?

- An inbound link is simply a hyperlink that can also have an href description: the anchor text

- Last Thurs: To put in a link, use 2 tags

\[<a href=http://users.soe.ucsc.edu/~maw>Prof. Walker's</a>\]

- Inbound links are important because of the way that Google stores a page’s data, and the way that they process a search query.
Google’s PageRank Algorithm

- If lots of pages point TO this page, this must be a “more important” page
- Tweaking the page rank algorithm can make or break a small business.
DATA and SOFTWARE DOWNLOADS

PROFESSIONAL EXPERIENCE

Professor of Computer Science, Natural Language and Dialogue Systems Lab, University of California, Santa Cruz, 2009 to present

Professor of Computer Science, Head of Cognitive Systems Group, University of Sheffield, Sheffield, England, 2003 to 2009

Principal Research Staff Member, ATT Labs - Research, Florham Park, N.J., Speech Processing Software and Technology Research, 1996 to 2003

Research Scientist, Mitsubishi Electric Research Laboratories, Cambridge, Ma., Interactive Learning and Entertainment, 1993 to 1996

Consultant, Hewlett Packard Laboratories, Bristol, England, on dialogue systems, speech technology, and personal information systems: 1989-1993

Researcher, Dialogue Modeling Department, Electrotechnical Laboratory, Tsukuba City, Japan: Summer 1991


EDUCATION


B.A. Computer and Information Science, With Honors, University of California Santa Cruz, 1984.
Who points to a page?

Site-specific searches

You can use special searches (called operators) to find pages that are similar to or link to a specific URL.

Learn more about these and other site-specific searches in the Webmaster Tools Help Center.

Search for pages that link to a URL

To search for web pages that link to a URL, use the "link:" operator. For example, to find pages that link to www.google.com, use [link:google.com]. You can also search for links to specific pages or directories, e.g. [link:google.com/webmasters].

Search for pages that are similar to a URL

The "related:" operator displays pages similar to a URL. For example, to find pages similar to nytimes.com, search for [related:nytimes.com]. We look at similarities between the link structures as well as other factors to help determine similar pages.

The "related:" operator returns the same results as clicking the Similar link within a search result's Instant Preview.
Stopped HERE