CS 361, Lecture 28

Jared Saia
University of New Mexico

Outline

- Administrative
- Master Theorem
- Search Engines

Administrative

- The project is due this Thursday in class.
- This deadline is strict - late projects will recieve no credit. (A partially completed project turned in on time will get some credit but a complete project turned in late will get no credit)
- To give you more time to work on the project, I'm not going to have a hw due on Thursday

Project Deliverables

Each Group should turn in one group project consisting of:

- About 6-12 pages of text (can be longer with appendix)
- 6-12 figures (please put multiple figures on one page where possible)
- Task list giving which tasks were performed by which group members. This should be signed by all members of the group.

Project Deliverables

Each *individual* should turn in the following:

- An *evaluation* of the contribution to the group project of every member of your group on a scale of 1(poor) to 10(excellent). To do this, write down the name of each member of your group (beside yourself), and put a number between 1 and 10 next to each name. Please be honest and professional in your evaluation. Your evaluation will be one factor used to determine the project grade for each member of your group.

Project Comments

Before you turn in the project:

- Reread "The Top n Project Mistakes" section in the project description section on the course web page
- *You will loose points if you make these same mistakes*

Master Method

The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows:

- If $a\,f(n/b) \le f(n)/K$ for some constant $K > 1$, then $T(n) = \Theta(f(n))$.
- If $a\,f(n/b) \ge K\,f(n)$ for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $a\,f(n/b) = f(n)$, then $T(n) = \Theta(f(n)\log_b n)$.

Proof

- If $f(n)$ is a *constant factor larger* than $a\,f(b/n)$, then the sum is a descending geometric series. The sum of any geometric series is a constant times its largest term. In this case, the largest term is the first term $f(n)$.
- If $f(n)$ is a *constant factor smaller* than $a\,f(b/n)$, then the sum is an ascending geometric series. The sum of any geometric series is a constant times its largest term. In this case, this is the last term, which by our earlier argument is $\Theta(n^{\log_b a})$.
- Finally, if $a\,f(b/n) = f(n)$, then each of the L terms in the summation is equal to $f(n)$.

Example

- $T(n) = T(n/2) + n\log n$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 1$,$b = 2$,$f(n) = n\log n$
- Here $a\,f(n/b) = n/2\log n/2$ is smaller than $f(n) = n\log n$ by a constant factor, so $T(n) = \Theta(n\log n)$

Last In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 2T(n/4) + n\log n}$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 2$,$b = 4$,$f(n) = n\log n$
- $f(n) = n\log n$ and $a\,f(n/b) = n/2\log(n/4)$
- $a\,f(n/b)$ is a constant factor smaller than $f(n)$, so the root node dominates.
- Thus the solution is $T(n) = \Theta(n\log n)$

In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 4T(n/2) + n^2}$
- Q: What is $f(n)$ and $a\,f(n/b)$?
- Q: Which of the three cases does the recurrence fall under (when n is large)?
- Q: What is the solution to this recurrence?

Limits of Master Theorem

- Consider the recurrence: $\boldsymbol{T(n) = 2T(n/2) + n/\log n}$
- We can't apply the master thm here, because $\boldsymbol{a\,f(n/b) = n/(\log n - 1)}$ isn't equal to $\boldsymbol{n/\lg n}$, but the difference isn't a constant factor
- We can use the recursion tree method

Limits of Master Theorem

- Consider the recurrence: $T(n) = 2T(n/2) + n/\log n$
- Not hard to see that the sum of the i-th level is $n/(\log n - i)$
- Depth of tree is $\log n$

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\log n} \frac{n}{\log n - i} &(1)\\
&= \sum_{j=1}^{\log n} \frac{n}{j} &(2)\\
&= n \sum_{j=1}^{\log n} \frac{1}{j} &(3)\\
&= \boxed{\Theta(n \log\log n)} &(4)
\end{aligned}$$

Take Away

- The Master Theorem is a fast way to solve certain types of recurrences
- However it is not as powerful as the recursion tree method
- When you see an appropriate recurrence, first try the Master Theorem, if this doesn't work, try recursion tree method or annihilators.

The Deep End of the Sandbox

"That's the deep end of the sandbox. I don't like to go there. A lephrechaun lives there and he tells me to burn things" - Ralph from the Simpsons

Beyond 361:

- CS461: More advanced data structures and algorithms, different classes of algorithms (greedy, dynamic programming), advanced mathematical analysis of algorithms (amortization, proofs of correctness, graph theory)
- Approximation Algorithms, Randomized Algorithms, Computational Geometry, Computational Biology, Graph Theoretic Algorithms, Cryptography, Online Algorithms, Experimental Algorithms, Smoothed Analysis, Complexity Theory, etc, etc.

Example Algorithmic Application

- Modern search engines are beginning to use algorithmic tools
- Google and *Clever* are two good search engines that use algorithmic techniques
- Their ranking algorithms are based on linear algebra (they make use of the eigenvectors of a certain type of matrix)
- *These algorithms could not have been developed without rigorous mathematical analysis*

The Two Types of Search Queries

Two types of search engine queries

- *Specific Queries*: E.g. "Does Netscape support the JDK 1.1 codesigning API?"
- *General Queries*: E.g. "Find Information about Java"

The Abundance Problem

For General Queries, we have

- **The Abundance Problem: The number of relevant pages is far too large for a human to digest.**

More specifically:

- There are now hundreds of millions of pages on the web
- For most search queries, the number of pages that contain the query words is on the order of thousands
- A person typically wants to look at less than five pages

What we want

One way to handle the abundance problem:

- Filter out from the huge set of relevant pages, those few pages which are most "authoritative" or definitive"
- To do this, we need a notion of what makes a page "authoritative"

An Example Problem

- If we search for all pages containing the word "Microsoft" we get back over a million pages.
- There is nothing about *content* of the page www.microsoft.com that makes it stand out.
  - This page doesn't use the term "microsoft" most frequently.
  - This page doesn't use the term "microsoft" most prominently.
  - There is nothing completely unique in the *textual content* of the page that identifies it as being authoritative.
- What is it that makes the page www.microsoft.com most authoritative?

Link Analysis

- Link Structure is one external measure of authority of a page
- If page x links to page y, x is conferring authority on page y
- There are some potential pitfalls here
  - Links don't always confer authority (e.g. they could be for navigation or to ads)
  - There is a balance between popularity and relevance

Crash Course in Graph Theory

A *Graph* is:

- A set of *nodes*, V, and
- A set of *edges*, E, where each edge is directed from some node in V to some other node in V

Note:

- A tree is a just a graph without cycles
- There are many useful algorithms known for graphs, so it's useful to frame problems in terms of Graph Theory where possible.

The Web as a Graph

- Let each page in the web, be a node of a graph
- Let there be an edge from node x to node y iff there is a link from page x to page y
- This formulation allows us to bring lots of great tools of graph theory to bear on the problem of link analysis.

Popularity and Relevance

- Consider the simple heuristic for returning authoritative pages: of all pages containing the search term, return those which have the greatest number of in-links.
- *This heuristic fails*
  - Many authoritative pages for a search term do not contain that term.
  - Examples: "seach engines", "automobile manufactures"
  - Many pages which are *not* authoritative for a given term would be incorrectly identified.
  - Examples: www.yahoo.com and www.microsoft.com would be considered authoritative for any search term these pages contained.

The *Clever* Approach

- Key observation: Certain pages point mainly to sites that are authoritative.
- These pages are called good *Hubs*
- Examples:
  - www.yahoo.com
  - “Jared’s Big Page Of Links To Cool Sites”
- Question: But then how do we define what makes a page a good Hub?
- The *Clever* algorithm and Google solve this problem

The *Clever* Approach

- We use a mutually-recursive definition of the notion of Authoritative and Hub-like
- Good *Hub* pages point to many good *Authoritative* pages
- Good *Authoritative* pages are pointed to by many good *Hub* pages

The Clever Algorithm

High Level Idea:

- Clever collects all pages containing the search term, or pages that are neighbors of pages containing the search term.
- It then iteratively propogates “authority” scores and “hub” scores
  - All pages start with equal authority and hub scores. In each step:
  - The pages with hub weight confer authority weight to the pages they point to.
  - The pages with authority weight confer hub weight to the pages *that point to them*