CS Colloquium

- Greedy Algorithms and Linear Programming - a two pronged attack on NP-Completeness
- Monday, Feb 8th, 4-5pm, Professor Samir Khuller

Since the development of the theory of NP-completeness in the early 70's, thousands of papers have appeared proving that various optimization problems are NP-hard. However, most of these papers do not suggest ways of coping with the intractability of optimization problems.

Approximation algorithms are essentially heuristics with a guaranteed performance behavior. In the first part of the talk I will outline the major goals and objectives of this research area, and will give a flavor of the kinds of results that we will discuss in a full course (CMSC 858Y) this spring. In the second part of the talk, we will discuss some advances in optimization problems arising in the area of scheduling. We will illustrate how linear programs can be used to develop approximation algorithms, and will also illustrate how they can be used in conjunction with greedy algorithms.
Submission confusion

• Automatic submission wasn't working for some people
• We'll deal, don't worry
• If you had course project manager installed and did a Team->update, you should have been set to go
• Preferences should have shown course project manager enabled for your project
Project 2: Idea book

- Due Tuesday, Feb 9th, 11:59pm
- signed in users add ideas and can vote + or - on ideas
- each user gets one vote per idea, can change vote
- list ideas by net votes
- Individual project: do not share or look at code by other students
Project 2

• Does not need to be scriptless
  • but encourage you to do so
• Doesn't need to protect against XSS
  • but encourage you to do so
Submitting project 2

- Deploy to Google Appengine
- you can take down or augment your project 1 deployment

- Enter information at https://spreadsheets.google.com/viewform?formkey=dFJaRHZmUUtPbXVGR0REM2tFLW02U0E6MA

- Commit to CVS and/or submit project to submit server
Cookies

• Today's cookies are full
• Tomorrow's cookies canceled due to pending blizzard
  • AWC Casino night also canceled
• Added cookies next Tuesday, 10 am
  • Sign up on course web page
Do something impressive

• You are just getting exposed to the bare edges of what can be done with web servers and appengine
  • mail, chat, sms, cron jobs
• An opportunity to do something impressive above and beyond assigned work
• Have an opportunity before spring break for demos and non-grade prizes
Recommended Servlet/JSP book

• I’m a fan of the head first series.

• Some people hate them

• Available free on line via Safari books

• when connecting from campus
Search engines are your friend

- Bing/google "core jstl" and you will get lots of pointers to the core taglibs in jstl
HTTP sessions

- HTTP is stateless
- Each connection is distinct, not associated by http to previous connections from the browser/user
- but... must... buy... from... amazon
Sessions

• Server creates a session object, and generates a random session identifier

• Each request needs to send session identifier back to server

• sending username or credit card number in each request would be a really bad idea
Sending the session identifier

- Cookies
- URL rewriting (add it to each URL)
- Hidden form fields
Cookies

• Cookies are easiest

• Server sends a cookie to browser that says:
  • everytime you send a request to www.foo.com, send this cookie
  • and the cookie expires in 2 hours

• Some browsers/users disable cookies
  • some proxies disable cookies

• You close your windows, walk away from the computer
  • still has cookies, still logged into Amazon, Fidelity, whatever...
URL rewriting and hidden form fields

• Require effort everywhere you have a URL or form in generated web pages

• that you want to have connect to the same session

• Doesn't depend on browser/user/proxy support

• If you open a new web page to a site, you get a new session

• But even if all windows closed, browser history may provide access to old session
Automatic sessions

• Generally, servers will try to maintain a session using either cookies or url rewritting, which ever is available
Scriptless JSPs

- No java source code in your jsp file
- Can go part way just using beans
- But probably want to go to using JavaServer Pages Standard Tag Library (JSTL)
JSP Header

<%@ page contentType="text/html;charset=UTF-8" isELIgnored="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:choose>
    <c:when test="${empty user}"/>
    <p>Welcome. <a href="${loginURL}">Log in</a> to sign ideas you post.</p>
</c:when>
    <c:otherwise/>
    <p>Hello <c:out value='${user.nickname}' escapeXml='true' /> (<a href="${logoutURL}">sign out</a>).</p>
</c:otherwise>
</c:choose>
Two different things going on

• `${….}` is a JSTL expression
• `<c:choose>`, `<c:when>` and `<c:otherwise>` are JSTL tags
  • the `c` comes from the prefix for the taglib, but `c` is standard for the JSTL core
JSTL EL expressions

- constant
- attribute_name
- implicit_object_name
- exp.name
- exp[exp]
- unaryOp exp
- exp binaryOp exp
Servlet attributes

• An attribute can be associated with a session, request or page
• Each attribute has a name
• Easy to set attributes via filters
  • For example, user, loginURL and logoutURL are attributes
Setting attributes in a filter

UserService userService = UserServiceFactory.getUserService();
User user = userService.getCurrentUser();
request.setAttribute("user", user);
String requestURI = request.getRequestURI();
String logIn = userService.createLoginURL(requestURI);
request.setAttribute("loginURL", logIn);
String logOut = userService.createLogoutURL(requestURI);
request.setAttribute("logoutURL", logOut);
empty \, exp

- True if \( \exp \) is null, an empty string, or an empty collection
exp.name

- Expects that exp evaluates to an object that has a getName or isName method
- for ${user.nickname}, user is an attribute of type User, which has a method
  public String getNickname() { … }
- Or, if exp is a map, exp.name is the same as exp.get("name")
\[ \text{exp}_1[\text{exp}_2] \]

- If \( \text{exp}_1 \) is a map or list,
  - gives \( \text{exp}_1.\text{get(\text{exp}_2)} \)
- If \( \text{exp}_1 \) is an array,
  - gives \( \text{exp}_1[\text{exp}_2] \)
- Otherwise, same as \( \text{exp}_1.\text{exp}_2 \), except that \( \text{exp}_2 \) is evaluated, rather than being required to be a constant
implicit objects

- param
- header
- cookie
- pageContext
  - gives access to request and response
- plus 7 more, less common ones
Examples

• On test.jsp, have
  • `<p>x = ${param.x}</p>`

• request test.jsp?x=5
  • Will generate `<p>x = 5</p>`
\texttt{<c:choose>}, \texttt{<c:when>}, ... \\

- \texttt{<c:choose>} can have any number of \texttt{<c:when>} clauses
- executes first with test that evaluates to true
- if none match, will execute \texttt{<c:otherwise>} clause
- Can also use \texttt{<c:if test="${exp}$">}, but no way to define an else clause
More core jstl tags

<c:forEach var="greeting" items="$\{greetings\}">
    <c:choose>
        <c:when test="$\{empty greeting.author\}">
            <p>An anonymous person wrote:</p>
        </c:when>
        <c:otherwise>
            <p><c:out value="$\{greeting.author.nickname\}" /> wrote:</p>
        </c:otherwise>
    </c:choose>
    <blockquote><c:out value="$\{greeting.content\}" /></blockquote>
</c:forEach>
• Can iterator through a collection
  • <c:forEach var="greeting" items="${greetings}"/>
• Or a range of values
  • <c:forEach var="x" begin="0" end="10" step="2"/>
• Ended by </c:forEach>
<c:out value=""...""> 

- Value is written to JSP output
- output is XML escaped
- can disable escaping with xmlEscape="false"
- Can supply default for when value is empty
- Can just embed ${...} in JSP body, but no escaping will occur
<c:set>

• Define Can use `<c:set var="name" value="${exp}"/>

• or

  • `<c:set var="name">
    <c:out value="${exp}"/>
  </c:set>`
<c:url>

• Encode a url
  • encodes parameters for the URL
    • parameter values are URL encoded
  • adds session identifier if needed
    • required to support sessions if cookies not enabled
c:url example

<c:url var="previous" value="foreach.jsp">
    <c:param name="first"
        value="$\{param.first - noOfRows\}" />
</c:url>

<a href="<c:out value="$\{previous\}"/>">Previous Page</a>
Filters and forwarding

- You can define filters that validate/transform/lookup parameters, and put them into attributes
  - can also check authorization
- A servlet or JSP page can also forward to another page
- A servlet can also send a redirect
  - can only be done in jsp page by including java source (script code)
Forwarding in a servlet

// forward request
RequestDispatcher view
    = req.getRequestDispatcher("/view.jsp");
view.forward(req, resp);

// send redirect
resp.sendRedirect("/guestbook.jsp");
Forwarding in a JSP

- `<jsp:forward page="view.jsp"/>`
Forwarding and redirects

- You can't do either if you've already flushed any output
- Should be OK if you've written output but not flushed it
- Forward takes all the information in the request (including attributes) and says "use this URL to determine the output that should be generated"
forward vs. redirect

- a forward stays within the server
- browser is ignorant of the redirected location
- a redirect goes back to the browser
- redirected URL shows in the location field of the browser
- reload loads the redirected URL