1 handout: slides
asst 4 posted
check your Wildcat Pass before coming to campus
send me email if you need to isolate/quarantine
EOLQs

COVID

CSPs
Constraint Satisfaction Problems
Map coloring: Given a map of \( n \) countries and a set of \( k \) colors, color every country differently from its neighbors.

\( n \)-queens: Given an \( n \times n \) chessboard, arrange \( n \) queens so that none is attacking another.

configuration: Given \( d_i \) options for each of the \( n \) components of a computer system (CPU, backplane, storage system, NICs), find a set of options compatible with the choices the customer has already made.

scheduling: Given a set of temporal constraints (eg, \( t_2 \geq t_1 + 30 \)), find a feasible set of times.

What algorithm would you use?
Types of Search Problems

- Shortest-path (vacuum, tile puzzle, M&C)
  - given operators and their costs
  - want least-cost path to a goal
  - sequential decision-making
  - goal depth/cost unknown

- Decisions with an adversary (chess, tic-tac-toe)
  - adversary might prevent path to best goal
  - want best assured outcome

- Constraint satisfaction (map coloring, $n$-queens)
  - set of unordered decisions
  - any goal is fine
  - fixed depth
  - explicit constraints on partial solutions
Gene Freuder: Father of Constraint Programming

Gene Freuder (UNH 1977–2001)
Do not expand any partial solution that violates a constraint.
asst 3, asst 4
projects
When assigning a variable, remove the conflicting values for all connected variables. Backtrack on domain wipeout.

Arc consistency: for every value in the domain of $x$, there exists a value in the domain of $y$ that satisfies all the constraints.
Variable choice: choose most constrained variable (smallest domain)
- want to keep tree small, failing quickly

Value choice: try least constraining value first (fewest removals)
- might as well succeed sooner if possible
### Example Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>BT</th>
<th>FC</th>
<th>FC+MCV</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>&gt; 1M</td>
<td>2K</td>
<td>60</td>
</tr>
<tr>
<td>n-Queens</td>
<td>&gt; 40M</td>
<td>&gt; 40M</td>
<td>820K</td>
</tr>
<tr>
<td>Zebra</td>
<td>3.9M</td>
<td>35K</td>
<td>500</td>
</tr>
<tr>
<td>Random 1</td>
<td>420K</td>
<td>26K</td>
<td>2K</td>
</tr>
<tr>
<td>Random 2</td>
<td>940K</td>
<td>77K</td>
<td>15K</td>
</tr>
</tbody>
</table>
Maintaining Arc Consistency

Ensure every value for $x$ has a legal value in all neighbors $y$. If one doesn’t, remove it and ensure consistency of all $y$. 
Ensure every value for $x$ has a legal value in all neighbors $y$. If one doesn’t, remove it and ensure consistency of all $y$.

while $Q$ is not empty
  $(x, y) \leftarrow \text{pop } Q$
  if revised$(x, y)$ then
    if $x$’s domain is now empty, return failure
    for every other neighbor $z$ of $x$
      push $(z, x)$ on $Q$

revised$(x, y)$
revised$\leftarrow$ false
foreach $v$ in $x$’s domain
  if no value in domain of $y$ is compatible with $v$
    remove $v$ from $x$’s domain
    revised$\leftarrow$ true
return revised
Other Algorithms for CSPs

- (Conflict-directed) Backjumping
- Dynamic backtracking
- Randomized restarting

Course projects!
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

Thanks!