Scenario Week 4
(comp203p)

Ilya Sergey

scenario@cs.ucl.ac.uk

22-26 February 2016
How many guards do we really need?

The answer depends on the shape of the gallery.
How many guards do we really need?

The answer depends on the shape of the gallery.

Here just 1 guard is okay.
How many guards do we really need?
How many guards do we really need?
How many guards do we really need?
How many guards do we really need?

3 guards will do.
How many guards do we really need?
Art Gallery Problem

For a given gallery (polygon), find the *minimal* set of guards’ positions, so together the guards can “see” the *whole* interior.

- *Complexity-wise, harder* than
  - SAT
  - Travelling salesman
  - Hamiltonian paths
  - Knapsack problem

**NP-hard**
Cheap-and-cheerful “almost” solutions

• Putting guard in each vertex
  ‣ n guards for a polygon with n vertices

• Václav Chvátal’s solution (1975)
  ‣ based on triangulation, ⌊n/3⌋ guards;

  ‣ **Chvátal’s theorem**: this number is always sufficient and is in some cases necessary.
Chvátal’s solution in practice

- 246 vertices
- 79 guards

Can we do better?
Scenario Week 4
(comp203p)

Art Gallery Competition

scenario@cs.ucl.ac.uk

22-26 February 2016
Part 1: Computing “good enough” set of guards

- **30** galleries of different shapes;
  - File with galleries: `guards.pol` (see Moodle page);
  - sizes of problems: small (<10) to large (~300);
- Compute a *complete* set of guards for each one of them;
- **Baseline** — Chvátal’s boundary (cannot get worse than that);
- Grading: **30 points**, one per gallery, for any solution, which is not worse than the baseline.
Encoding of the problems (Part 1)

guards.pol

<table>
<thead>
<tr>
<th>Number</th>
<th>Polygon</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0, 0), (2, 0), (2, 1), (1, 1), (1, 3), (0, 3)</td>
</tr>
<tr>
<td>2</td>
<td>(0, 0), (5, 0), (5, 2), (4.2312351, 1.234), (1, 1), (0, 2)</td>
</tr>
</tbody>
</table>

- Polygon is “on the left”
- No holes inside
The next subsections outline the details of the tasks.

2.1 Computing the Set of Guards (Part I)

In this task, you will be given a text file \texttt{guards.pol} (available from the Moodle page of the course), containing definitions of \textbf{30} simple polygons (with no holes or self-intersections).

Each line contains a polygon number, followed by its definition after a colon (ignoring possible spaces between other lexical tokens). The definition of a polygon is a list of coordinates of its vertices \((x, y)\), where \(x, y\) can be integers or double-precision floating-point numbers. The sequence of the vertices is arranged in a way that the interior of the polygon will stay on the left, when one "walks" from one vertex to the next one. The successor of the last vertex in the list is the first vertex. For instance, the following file describes two polygons in the defined format, numbered 1 and 2, correspondingly and depicted in Figure 1:

\begin{verbatim}
guards.pol
1: (0, 0), (2, 0), (2, 1), (1, 1), (1, 3), (0, 3)
2: (0, 0), (5, 0), (5, 2), (4.2312351, 1.234), (1, 1), (0, 2)
\end{verbatim}

Your goal for this task is to compute, for each polygon, a set of guards, which together can see its whole interior, and the size of the set is not larger than Chvátal's boundary \(b_n/c\).

The solution for this task is a text file. You can implement your algorithm in any programming language of your preference and use any libraries you consider necessary. You don't have to (and should not) submit the code.

The file with the results should start with the first line containing the name of the team and the second line being its password. If those don't match, the file will not be accepted by the system. The remaining lines should contain the solutions in the following format:

\texttt{polygon number : comma-separated list of guards}\n
where guards are represented as pairs of their coordinates \((x, y)\). A solution for each problem, along with its number, should be placed on a separate line. There is no specific order imposed on the sequence of the guards or solutions. For instance, a solution for the above file \texttt{guards.pol}, submitted by the team \texttt{tiger} with a password \texttt{lt671vecrskq} might look as follows:

\begin{verbatim}
tiger
lt671vecrskq
2: (0, 2), (4.3, 1)
1: (0.2, 2.5), (2, 0.5)
\end{verbatim}

**Solution file:**

- **team name**: tiger
- **team's password**: lt671vecrskq
- **per-polygon guards**:
  - **Polygon 1**: \((0.2, 2.5), (2, 0.5)\)
  - **Polygon 2**: \((0, 2), (4.3, 1)\)
Checking and submitting solutions

• **Warning:** double-precision floating-point arithmetic
  • all equalities are up to $\varepsilon = 0.000,000,000,1$
• Details on acceptance criteria are in the *specification* (on Moodle)
• Submit your solutions here (under Part 1):

  http://artgallery.cs.ucl.ac.uk

Solutions are accepted until **14:00 GMT 26 Feb 2016**
Part 2: Checking a (flawed) set of guards

• **20** galleries of different shapes with sets of guards;
  • File with problems: `check.pol` (see Moodle page);
  • sizes of problems: small (<10) to gigantic (~500);
• Find a *refutation* (a point within a polygon, not visible from the given guards) for *each* problem in the set;
• *Any* refutation will do.
• Grading: **20 points**, one per problem/refutation.
Encoding of the problems (Part 2)

File with problems

check.pol

1: (0, 0), (2, 0), (2, 1), (1, 1), (1, 3), (0, 3); (0, 3), (1, 2)
2: (0, 0), (5, 0), (5, 2), (4.2312351, 1.234), (1, 1), (0, 2); (0, 2), (3, 1)
Encoding your solutions (Part 2)

Solution file:

<table>
<thead>
<tr>
<th>Team Name</th>
<th>Password</th>
<th>Polygon 1</th>
<th>Polygon 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiger</td>
<td>lt671vecrskq</td>
<td>(1.56, 0.53)</td>
<td>(4.74, 1.53)</td>
</tr>
</tbody>
</table>

• Submit your solutions here (under Part 2):

http://artgallery.cs.ucl.ac.uk

Solutions are accepted until 14:00 GMT 26 Feb 2016
Part 3: Visualisation

- Implement a visualiser for galleries, guards and visibility:
  - drawing galleries;
  - drawing visibility areas from specific guards;
  - drawing refutations for incomplete guard sets.

- Grading: **15 points**

- Assessed by the organisers from **14:00 till 17:00, 26 Feb 16**

- book a slot for your team!
Part 4: Implementation report

• Describe your implementation experience
  • language, algorithms, etc.
  • details in the specification (see Moodle)

• Grading: 15 points

• Submit electronically by 17:00, 26 Feb 2016 (one per team)
Part 5: The Competition!

- Compete with other teams for the best solutions in Part 1.
- Teams with all accepted solutions ranked amongst each other first.
- Check the score table http://artgallery.cs.ucl.ac.uk at for details
- Grading: up to 20 points.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2-3</td>
<td>15</td>
</tr>
<tr>
<td>4-5</td>
<td>10</td>
</tr>
<tr>
<td>6-7</td>
<td>5</td>
</tr>
<tr>
<td>≥8</td>
<td>0</td>
</tr>
</tbody>
</table>
Overall grading

<table>
<thead>
<tr>
<th>Task</th>
<th>Max grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing “good enough” guard set</td>
<td>30</td>
</tr>
<tr>
<td>Checking a flawed guard set</td>
<td>20</td>
</tr>
<tr>
<td>Visualisation of the solutions</td>
<td>15</td>
</tr>
<tr>
<td>Implementation report</td>
<td>15</td>
</tr>
<tr>
<td>The Competition</td>
<td>20</td>
</tr>
<tr>
<td>Time</td>
<td>Monday, 22 Feb</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>10:00-11:00</td>
<td>ULU Malet Suite (Introductory lecture)</td>
</tr>
<tr>
<td>11:00-13:00</td>
<td>Lunch</td>
</tr>
<tr>
<td>13:00-14:00</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00-16:00</td>
<td>Cruciform B404 - LT2</td>
</tr>
<tr>
<td>16:00-18:00</td>
<td>Roberts 106</td>
</tr>
</tbody>
</table>

Helpdesk (green) = Time and locations where staff and/or TAs will be present so you could ask questions.
Lectures (blue) = Introductory and concluding lectures
Good luck!