Scenario Week 4:
Art Gallery Competition

Organisers' Report

scenario@cs.ucl.ac.uk

26 February 2016
The Week

- 94 participants
- 24 teams
- 1892 submissions for Part 1
- 468 submissions for Part 2
- One subtle server bug discovered (non-lethal)
  - floating-point arithmetic is a nasty thing…
The Problem and its solutions

• Original formulation is due to Chvátal (1975)
• Textbook algorithm by Fisk (1978)
  • Triangulation and 3-colouring, delivers a decent $\lfloor n/3 \rfloor$ solution
• Better solutions exist for specific polygons
  • L-partitioning for rectilinear polygons: $\lfloor n/4 \rfloor$ solution
  • Detecting convex suqppolygons — just one guard required;
  • Even better: detecting “star” sub-polygons;
• A good survey: “Art Gallery Theorems and Algorithms” by O’Rourke
Initial setup

- **Part 1**: 30 polygons for finding the best guards sets
  - 1–5 are trivial — to test intuition (small size);
  - 8–13 are rectilinear (74–334 vertices);
  - 15–17 composed from triangles (42-360 vertices);
  - 18–26 are “quasi-convex” with large convex regions;
  - 27–30 composed from various random shapes.

- **Part 2**: 20 polygons/guards to find refutations
  - About 2/3 problems had one node non-covered (easy to find);
  - 6 or 7 problems required a proper algorithms (or a lot of patience).
Part 1, polygon 14
Checking your solutions

• Server is written in Scala via Spray framework on servlets (1500 LOC);
  • Run during the week on a single Linux machine with 4 GB RAM;
  • Each team’s submissions are processed by a separate actor (non-blocking);
• All geometric processing is implemented in Scala from scratch, no third-party libraries (1800 LOC, including tests);
• ~150 unit tests + several randomised testing procedures (bazillions of randomly-generated polygons);
  • still missed one floating-point bug :(
• Guards checking procedure is a slightly modified version of Joe-Simpson algorithm for visibility polygons (1985).
Checking your solutions
Detecting grey areas precisely

Step 1: compute *all* individual visibility areas via Joe-Simpson algorithm.
Detecting grey areas precisely

Step 2: triangulate the initial polygon
Detecting grey areas *precisely*

**Step 3**: add visibility areas one by one, compute intersections with present triangles and $\Delta$-partition again.
Detecting grey areas precisely

**Step 3**: add visibility areas one by one, compute intersections with present triangles and $\Delta$-partition again
Detecting grey areas precisely

Step 3*: add visibility areas one by one, compute intersections with present triangles and Δ-partition again
Detecting grey areas precisely

**Loop Invariant**: at the end of each iteration, each triangle is either *fully visible* or is *fully grey* (invisible).

**Proof**
By induction on edges of visibility areas.
Detecting grey areas precisely

Step 4: iterate through all the triangles of the partition and check if a centre of each belongs to some visibility area. If not, return the centre of such triangle as a refutation.
Behind the Scenes
Geeks and repetitive tasks

time spent

geek
non-geek

does it manually
writes script to automate
gets annoyed
runs script

makes fun of geek's complicated method

loses

wins

task size
Kareem’s Demo
Analysing submission patterns

- Taking data about **Part 1** submissions
- Recording time of successful submissions (green)
- Propagated submissions (purple)
- No submission (blank)
“Experimentators”
“Hard workers”
“Real geeks”
“Late bloomers”
“Parallel computers”
Part 1 problems: Toughies

- polygon 10 (338 vertices)
- polygon 12 (288 vertices)
- polygon 13 (334 vertices)
- polygon 17 (360 vertices)
Shameless Advertisement
MSc Programme by PPLV: Logic, Semantics and Verification of Programs.

• Analysis of the correctness of large systems;
• Concurrent and distributed programming;
• Formal methods and theorem proving (yay!);
• Dark magic of abstract algebra and category theory to make better software (without actual bugs);
• Starts next year, apply in 2017!

http://pplv.cs.ucl.ac.uk
The Competition
Ranking solutions

• Solutions were **not** ranked based on the total sum of guards;

• Instead, (1) for each polygon, teams were grouped according to the number of guards, smaller is better (e.g., 5 groups of solutions)
  • Teams that didn’t solve a polygon were all put into the “last” group for this polygon (e.g., group 6 for the previous example)

• (2) Next, per-polygon rankings were aggregated for each team;

• (3) *Overall* ranking is based on a *sum* of per-polygon rankings;
  • Team B that did worse than team A for some problems might still be ranked above A
  • Teams that solved all 30 problems were ranked first amongst each other.
Expectations and Surprises

- For the first three days results in Part 1 were consistent with the triangulation-based algorithm.
- Last-minute results look way better than the baseline.
- Several top-ranked solutions are astonishingly good.
  - Although we suspect some of them to be hand-crafted.
- Part 2 didn’t seem to pose too much challenge after all.
## Finish line

<table>
<thead>
<tr>
<th>Last submitted</th>
<th>Rank</th>
<th>Done</th>
<th>1 (3)</th>
<th>2 (4)</th>
<th>3 (6)</th>
<th>4 (8)</th>
<th>5 (12)</th>
<th>6 (180)</th>
<th>7 (92)</th>
<th>8 (244)</th>
<th>9 (74)</th>
<th>10 (338)</th>
<th>11 (104)</th>
<th>12 (288)</th>
<th>13 (334)</th>
<th>14 (58)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:27:23, 24 Feb 2016</td>
<td>1</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>47</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:10:28, 26 Feb 2016</td>
<td>2</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>48</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:50:36, 26 Feb 2016</td>
<td>2</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>48</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:51:13, 26 Feb 2016</td>
<td>3</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>48</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:49:19, 26 Feb 2016</td>
<td>4</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>48</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:54:27, 26 Feb 2016</td>
<td>5</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>48</td>
<td>14</td>
<td>41</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>13:46:45, 26 Feb 2016</td>
<td>6</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>25</td>
<td>11</td>
<td>34</td>
<td>10</td>
<td>47</td>
<td>14</td>
<td>41</td>
<td>47</td>
<td>5</td>
</tr>
<tr>
<td>13:45:49, 26 Feb 2016</td>
<td>6</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>11</td>
<td>34</td>
<td>10</td>
<td>51</td>
<td>14</td>
<td>43</td>
<td>47</td>
<td>5</td>
</tr>
<tr>
<td>13:58:56, 26 Feb 2016</td>
<td>7</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>49</td>
<td>14</td>
<td>41</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>13:47:00, 26 Feb 2016</td>
<td>7</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>49</td>
<td>14</td>
<td>41</td>
<td>46</td>
<td>5</td>
</tr>
<tr>
<td>13:28:53, 26 Feb 2016</td>
<td>8</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>11</td>
<td>33</td>
<td>10</td>
<td>111</td>
<td>14</td>
<td>93</td>
<td>43</td>
<td>5</td>
</tr>
</tbody>
</table>
The Winners
Surprises

- Part 1, polygon 23
- 100 vertices
- 7 guards (best solution)
In conclusion

• This week was fun to design…
• …and even more fun to observe.
• We hope, it was fun to participate in it.

Have a nice weekend…
… and take some time to enjoy art in galleries, which are now well-guarded.

Thanks!