NWERC 2015
Presentation of solutions

The Jury

2015-11-29

NWERC 2015 Jury

- Per Austrin (KTH Royal Institute of Technology)
- Gregor Behnke (Ulm University)
- Jeroen Bransen (Utrecht University)
- Egor Dranischnikow (CST AG, Darmstadt)
- Tommy Färnqvist (Linköping University)
- Florin Ghesu (Siemens Healthcare/FAU Erlangen-Nürnberg)
- Robin Lee (Google)
- Lukáš Poláček (Spotify)
- Stefan Toman (TU München)
- Tobias Werth (FAU Erlangen-Nürnberg)
- Paul Wild (FAU Erlangen-Nürnberg)

Big thanks to our test solvers

- Michal Forišek (Comenius University)
- Jan Kuipers (AppTornado)

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution

1. The lowest bit x_0 of `x` and `s` must be identical.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is x_0^s_1.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is x_0^s_1.
3. Repeat until the highest bit.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution – Option 1

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is $x_0 \hat{} s_1$.
3. Repeat until the highest bit.

Solution – Option 2

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution – Option 1

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is x_0^s_1.
3. Repeat until the highest bit.

Solution – Option 2

1. The computation is bijective.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution – Option 1

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is $x_0\hat{}s_1$.
3. Repeat until the highest bit.

Solution – Option 2

1. The computation is bijective.
2. Iterate over all values (there are only 256) for `x` and store the result in a map.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution – Option 1

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is x_0^s_1.
3. Repeat until the highest bit.

Solution – Option 2

1. The computation is bijective.
2. Iterate over all values (there are only 256) for `x` and store the result in a map.
3. Access the map to produce the output.

J – Jumbled Communication

Problem

Assume the equation `s = x ^ (x << 1)`. Given `s` determine `x`.

Solution – Option 1

1. The lowest bit x_0 of `x` and `s` must be identical.
2. The next bit x_1 of `x` is x_0^s_1.
3. Repeat until the highest bit.

Solution – Option 2

1. The computation is bijective.
2. Iterate over all values (there are only 256) for `x` and store the result in a map.
3. Access the map to produce the output.

Statistics: 131 submissions, 92 accepted

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$
3. For each digit:

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$
3. For each digit:
   - Multiply x and y by 2

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$
3. For each digit:
   - Multiply x and y by 2
   - If the digit is 1 or 3: add 1 to x

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$
3. For each digit:
   - Multiply x and y by 2
   - If the digit is 1 or 3: add 1 to x
   - If the digit is 2 or 3: add 1 to y

I – Identifying Map Tiles

Problem

Given quadkey such as 130, output zoom level and coordinates.

Solution

1. Zoom level is length of quadkey
2. Start with $x = 0$ and $y = 0$
3. For each digit:
    - Multiply x and y by 2
    - If the digit is 1 or 3: add 1 to x
    - If the digit is 2 or 3: add 1 to y

Possible pitfalls

1. Reading in the quadkey as integer

Statistics: 119 submissions, 94 accepted

A – Assigning Workstations

Problem

Schedule jobs minimizing the number of machines used such that no machine locks between consecutive jobs, i.e. it is unused for more than m minutes.

Solution

A – Assigning Workstations

Problem

Schedule jobs minimizing the number of machines used such that no machine locks between consecutive jobs, i.e. it is unused for more than m minutes.

Solution

1. Sort the jobs by starting time.

A – Assigning Workstations

Problem

Schedule jobs minimizing the number of machines used such that no machine locks between consecutive jobs, i.e. it is unused for more than m minutes.

Solution

1. Sort the jobs by starting time.
2. Assign jobs greedily: if there are unlocked machines use the one that will lock first.

A – Assigning Workstations

Problem

Schedule jobs minimizing the number of machines used such that no machine locks between consecutive jobs, i.e. it is unused for more than m minutes.

Solution

1. Sort the jobs by starting time.
2. Assign jobs greedily: if there are unlocked machines use the one that will lock first.
3. Save the times when machines will lock in a priority queue.

A – Assigning Workstations

Problem

Schedule jobs minimizing the number of machines used such that no machine locks between consecutive jobs, i.e. it is unused for more than *m* minutes.

Solution

1. Sort the jobs by starting time.
2. Assign jobs greedily: if there are unlocked machines use the one that will lock first.
3. Save the times when machines will lock in a priority queue.

Statistics: 258 submissions, 63 accepted

E – Elementary Math

Problem

Given n pairs of numbers, put +, - or $*$ between them such that no two results are the same.

Solution

E – Elementary Math

Problem

Given n pairs of numbers, put +, - or $*$ between them such that no two results are the same.

Solution

1. Make bipartite graph with n nodes on one side, and a node for every possible answer on the other side

E – Elementary Math

Problem

Given n pairs of numbers, put +, - or $*$ between them such that no two results are the same.

Solution

1. Make bipartite graph with n nodes on one side, and a node for every possible answer on the other side
2. Connect each pair of numbers to their 3 possible answers

E – Elementary Math

Problem

Given n pairs of numbers, put +, - or $*$ between them such that no two results are the same.

Solution

1. Make bipartite graph with n nodes on one side, and a node for every possible answer on the other side
2. Connect each pair of numbers to their 3 possible answers
3. Find a maximum matching, if this is of size n we have a solution

E – Elementary Math

Problem

Given n pairs of numbers, put +, - or $*$ between them such that no two results are the same.

Solution

1. Make bipartite graph with n nodes on one side, and a node for every possible answer on the other side
2. Connect each pair of numbers to their 3 possible answers
3. Find a maximum matching, if this is of size n we have a solution

Possible pitfalls

1. Result does not fit in an `int`
2. Greedy or brute force approach will not work

Statistics: 194 submissions, 45 accepted

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
   1. If any of the dishes in conflict, skip it.

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
   1. If any of the dishes in conflict, skip it.
   2. List all ingredients included, without repetitions.

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
   1. If any of the dishes in conflict, skip it.
   2. List all ingredients included, without repetitions.
   3. Ways of doing this combo is product of number of brands of these ingredients.

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
   1. If any of the dishes in conflict, skip it.
   2. List all ingredients included, without repetitions.
   3. Ways of doing this combo is product of number of brands of these ingredients.
2. Add up the results.

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
    1. If any of the dishes in conflict, skip it.
    2. List all ingredients included, without repetitions.
    3. Ways of doing this combo is product of number of brands of these ingredients.
2. Add up the results.
3. Pitfall: be *very careful* with overflows!

K - Kitchen Combinatorics

Problem

How many ways to pick three non-conflicting dishes and brands for the ingredients included?

Solution

1. For all $\leq 25^3$ combinations of starter, main, and dessert:
   1. If any of the dishes in conflict, skip it.
   2. List all ingredients included, without repetitions.
   3. Ways of doing this combo is product of number of brands of these ingredients.
2. Add up the results.
3. Pitfall: be *very careful* with overflows!

Statistics: 186 submissions, 46 accepted

D – Debugging

Problem

Find an optimal stategy minimizing the time for debugging n lines of code (LoC), given:

- r - time needed to compile+run the program
- p - time needed to insert a `printf` statement

D – Debugging

Problem

Find an optimal stategy minimizing the time for debugging n lines of code (LoC), given:

- r - time needed to compile+run the program
- p - time needed to insert a `printf` statement

Solution (greedy part)

1. best strategy for inserting k `printf` statements in n LoC - equidistant, every $\left\lceil \frac{n}{k+1} \right\rceil$ lines - minimizing the number of LoCs in the next step.
2. for n LoC the best strategy could be to insert 1, 2, … or $n-1$ `printf` statements.

D – Debugging

Solution (recursive part)

Boils down to calculating the recursion:

$$T(1) = 0$$
$$T(n) = \min_{1 \leq k < n} \left\{ k \cdot p + T\left(\left\lceil \frac{n}{k+1} \right\rceil \right) \right\} + r$$

1. naive recursion - way too slow
2. naive DP calculates too many not needed states (e.g. $n-1$, $n-2$, $\ldots$) $\rightarrow O(n^2)$ - too slow
3. memoization calculates only the needed states $\rightarrow O(n \log n)$ - good enough
4. cleverer versions of DP could be also fast enough

D – Debugging

Solution (recursive part)

Boils down to calculating the recursion:

$$T(1) = 0$$
$$T(n) = \min_{1 \leq k < n} \left\{ k \cdot p + T\left(\left\lceil \frac{n}{k+1} \right\rceil\right) \right\} + r$$

1. naive recursion - way too slow
2. naive DP calculates too many not needed states (e.g. $n-1,\ n-2,\ \ldots$) $\to O(n^2)$ - too slow
3. memoization calculates only the needed states $\to O(n \log n)$ - good enough
4. cleverer versions of DP could be also fast enough

Statistics: 142 submissions, 25 accepted

C - Cleaning Pipes

Problem

Given a set of lines (with one end called "origin" or well). Is there a subset L of the lines such that for every intersection of two lines exactly one of the pipes intersection is in L?

C - Cleaning Pipes

Problem

Given a set of lines (with one end called “origin” or well). Is there a subset L of the lines such that for every intersection of two lines exactly one of the pipes intersection is in L?

Solution

1. Find all intersecting pairs of pipes (coordinates and number of pipes are small, so don’t have to be careful).

C - Cleaning Pipes

Problem

Given a set of lines (with one end called "origin" or well). Is there a subset L of the lines such that for every intersection of two lines exactly one of the pipes intersection is in L?

Solution

1. Find all intersecting pairs of pipes (coordinates and number of pipes are small, so don't have to be careful).
2. Make graph: edge from pipe i to pipe j if they intersect.

C - Cleaning Pipes

Problem

Given a set of lines (with one end called "origin" or well). Is there a subset L of the lines such that for every intersection of two lines exactly one of the pipes intersection is in L?

Solution

1. Find all intersecting pairs of pipes (coordinates and number of pipes are small, so don't have to be careful).
2. Make graph: edge from pipe i to pipe j if they intersect.
3. Cleaning is possible if this graph is bipartite (send in robots to one side of the bipartition).

C - Cleaning Pipes

Problem

Given a set of lines (with one end called "origin" or well). Is there a subset L of the lines such that for every intersection of two lines exactly one of the pipes intersection is in L?

Solution

1. Find all intersecting pairs of pipes (coordinates and number of pipes are small, so don't have to be careful).
2. Make graph: edge from pipe i to pipe j if they intersect.
3. Cleaning is possible if this graph is bipartite (send in robots to one side of the bipartition).

Statistics: 132 submissions, 33 accepted

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

Solution

- If x, y appear in different order in two permutations π, ϕ, we call this an *inversion* of π and ϕ.

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

Solution

- If x, y appear in different order in two permutations π, ϕ, we call this an *inversion* of π and ϕ.
- If x, y appear in the same order everywhere, they don't form any inversions.

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

Solution

- If x, y appear in different order in two permutations π, ϕ, we call this an *inversion* of π and ϕ.
- If x, y appear in the same order everywhere, they don't form any inversions.
- If x, y do not appear in the same order, they form exactly two inversions: two permutations agree while the third one is different.

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

Solution

- If x, y appear in different order in two permutations π, ϕ, we call this an *inversion* of π and ϕ.
- If x, y appear in the same order everywhere, they don't form any inversions.
- If x, y do not appear in the same order, they form exactly two inversions: two permutations agree while the third one is different.
- Counting the number of inversions can be done in $O(n \log n)$.

G – Guessing Camels

Problem

Given are three permutations. Count the number of pairs of numbers appearing in the same order in all three of them.

Solution

- If x, y appear in different order in two permutations π, ϕ, we call this an *inversion* of π and ϕ.
- If x, y appear in the same order everywhere, they don't form any inversions.
- If x, y do not appear in the same order, they form exactly two inversions: two permutations agree while the third one is different.
- Counting the number of inversions can be done in $O(n \log n)$.

Statistics: 97 submissions, 13 accepted

H – Hole in One

Problem

Given a set of line segments, and points s and t, find a way to bounce a ball from s to t. Line segments disappear once hit. Maximize number of hits.

H – Hole in One

Problem

Given a set of line segments, and points s and t, find a way to bounce a ball from s to t. Line segments disappear once hit. Maximize number of hits.

Solution

1. The number of walls is small, so try all sequences of bounces.

H – Hole in One

Problem

Given a set of line segments, and points s and t, find a way to bounce a ball from s to t. Line segments disappear once hit. Maximize number of hits.

Solution

1. The number of walls is small, so try all sequences of bounces.
2. For each sequence:
    1. Calculate the target direction by reflecting t on all walls in reverse order.

H – Hole in One

Problem

Given a set of line segments, and points s and t, find a way to bounce a ball from s to t. Line segments disappear once hit. Maximize number of hits.

Solution

1. The number of walls is small, so try all sequences of bounces.
2. For each sequence:
   1. Calculate the target direction by reflecting t on all walls in reverse order.
   2. Perform ray tracing in that direction, keep track of deleted walls, check if reflections agree with sequence.

H – Hole in One

Problem

Given a set of line segments, and points s and t, find a way to bounce a ball from s to t. Line segments disappear once hit. Maximize number of hits.

Solution

1. The number of walls is small, so try all sequences of bounces.
2. For each sequence:
   1. Calculate the target direction by reflecting t on all walls in reverse order.
   2. Perform ray tracing in that direction, keep track of deleted walls, check if reflections agree with sequence.

Statistics: 14 submissions, 3 accepted

B - Better Productivity

Problem

Find a partition of n time intervals into p groups such that the sum of intersections, computed for each group, is maximized. *Problem inspired from artificial neural activity and interaction models.*

B - Better Productivity

Problem

Find a partition of n time intervals into p groups such that the sum of intersections, computed for each group, is maximized. *Problem inspired from artificial neural activity and interaction models.*

Solution

1. Split intervals in two disjoint categories:
   - Good workers (n_1): covering the interval of another worker
   - Bad workers ($n - n_1$): all others

B - Better Productivity

Problem

Find a partition of n time intervals into p groups such that the sum of intersections, computed for each group, is maximized. *Problem inspired from artificial neural activity and interaction models.*

Solution

1. Split intervals in two disjoint categories:
   - Good workers (n_1): covering the interval of another worker
   - Bad workers ($n - n_1$): all others
2. Partition the bad workers using following DP scheme:
   $d(i, j)$ - answer using first i intervals partitioned to j groups

B - Better Productivity

Problem

Find a partition of n time intervals into p groups such that the sum of intersections, computed for each group, is maximized. *Problem inspired from artificial neural activity and interaction models.*

Solution

1. Split intervals in two disjoint categories:
   - Good workers (n_1): covering the interval of another worker
   - Bad workers ($n - n_1$): all others
2. Partition the bad workers using following DP scheme:
   $d(i, j)$ - answer using first i intervals partitioned to j groups
3. Observation: good workers do **not** affect the current partition

B - Better Productivity

Greedy Improvement

1. Improve solution by building extra groups, containing only the largest t good workers

Greedy Improvement

1. Improve solution by building extra groups, containing only the largest t good workers
2. The final answer is:

$$\max_{1 \leq t \leq p} dp(n_1, t) + \sum(\text{largest } p - t \text{ intervals of good workers})$$

B - Better Productivity

Greedy Improvement

1. Improve solution by building extra groups, containing only the largest t good workers
2. The final answer is:

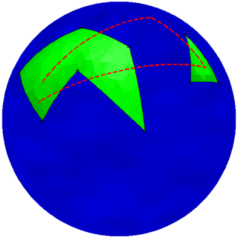
$$\max_{1 \leq t \leq p} dp(n_1, t) + \sum(\text{largest } p - t \text{ intervals of good workers})$$

Statistics: 8 submissions, ?? accepted

F - Flight Plan Evaluation

Problem

Given some spherical arcs and spherical polygons, how much of the arcs is contained in the polygons?

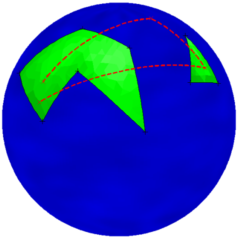


Solution

F - Flight Plan Evaluation

Problem

Given some spherical arcs and spherical polygons, how much of the arcs is contained in the polygons?



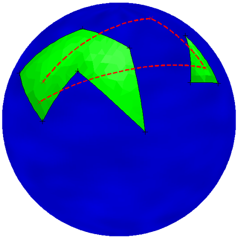
Solution

1. For each route segment, find all points at which we switch between land and water.
   - We were promised that no tricky cases occur $\Rightarrow$ only need to find all intersections of route segment with polygon segments.

F - Flight Plan Evaluation

Problem

Given some spherical arcs and spherical polygons, how much of the arcs is contained in the polygons?



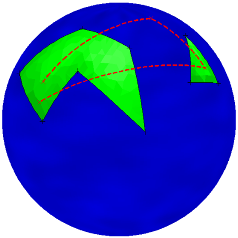
Solution

1. For each route segment, find all points at which we switch between land and water.
    - We were promised that no tricky cases occur $\Rightarrow$ only need to find all intersections of route segment with polygon segments.
2. Sort points by distance from start of segment

F - Flight Plan Evaluation

Problem

Given some spherical arcs and spherical polygons, how much of the arcs is contained in the polygons?



Solution

1. For each route segment, find all points at which we switch between land and water.
   - We were promised that no tricky cases occur ⇒ only need to find all intersections of route segment with polygon segments.
2. Sort points by distance from start of segment
3. Alternate between land and water (we know that we start on land)

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

1. The great circle on which the arc from P_1 to P_2 runs are the unit vectors orthogonal to $P_1 \times P_2$

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

1. The great circle on which the arc from P_1 to P_2 runs are the unit vectors orthogonal to $P_1 \times P_2$
2. Intersection of $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$ (if it exists) must lie on both great circles $\Leftrightarrow$ orthogonal to both $P_1 \times P_2$ and $Q_1 \times Q_2$.

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

1. The great circle on which the arc from P_1 to P_2 runs are the unit vectors orthogonal to $P_1 \times P_2$
2. Intersection of $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$ (if it exists) must lie on both great circles $\Leftrightarrow$ orthogonal to both $P_1 \times P_2$ and $Q_1 \times Q_2$.
3. Only two possible candidate intersection points:
$$\pm \frac{(P_1 \times P_2) \times (Q_1 \times Q_2)}{\|(P_1 \times P_2) \times (Q_1 \times Q_2)\|_2}$$
Watch out for division by zero (cocircular arcs).

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

1. The great circle on which the arc from P_1 to P_2 runs are the unit vectors orthogonal to $P_1 \times P_2$
2. Intersection of $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$ (if it exists) must lie on both great circles $\Leftrightarrow$ orthogonal to both $P_1 \times P_2$ and $Q_1 \times Q_2$.
3. Only two possible candidate intersection points:
$$\pm \frac{(P_1 \times P_2) \times (Q_1 \times Q_2)}{\|(P_1 \times P_2) \times (Q_1 \times Q_2)\|_2}$$
Watch out for division by zero (cocircular arcs).
4. To test if candidate point R is on $P_1 \curvearrowright P_2$: check if $d(P_1, R) + d(R, P_2) = d(P_1, P_2)$.

Flight Plan Evaluation

Sub-problem

Find intersection of two spherical arcs $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$.

Solution

1. The great circle on which the arc from P_1 to P_2 runs are the unit vectors orthogonal to $P_1 \times P_2$
2. Intersection of $P_1 \curvearrowright P_2$ and $Q_1 \curvearrowright Q_2$ (if it exists) must lie on both great circles $\Leftrightarrow$ orthogonal to both $P_1 \times P_2$ and $Q_1 \times Q_2$.
3. Only two possible candidate intersection points:
$$\pm \frac{(P_1 \times P_2) \times (Q_1 \times Q_2)}{\|(P_1 \times P_2) \times (Q_1 \times Q_2)\|_2}$$
Watch out for division by zero (cocircular arcs).
4. To test if candidate point R is on $P_1 \curvearrowright P_2$: check if $d(P_1, R) + d(R, P_2) = d(P_1, P_2)$.

Statistics: 9 submissions, 2 accepted

Random numbers produced by the jury

1217 number of posts made in the jury's forum.
(NWERC 2014: 1087)

915 commits made to the problem set repository.
(NWERC 2014: 671)

416 number of lines of code used in total by the shortest judge solutions to solve the entire problem set.
(NWERC 2014: 380)

16.6 average number of jury solutions per problem, including incorrect ones.
(NWERC 2014: 16.7)

62.5% fraction of judge solutions to problem B that were successfully challenged by a single test case 4 days before the contest.