NWERC 2014
Presentation of solutions

The Jury

2014-11-30

NWERC 2014 Jury

- Per Austrin (KTH Royal Institute of Technology)
- Thomas Beuman (Leiden University)
- Jeroen Bransen (Utrecht University)
- Tommy Färnqvist (LiU)
- Jan Kuipers (AppTornado)
- Lukáš Poláček (KTH and Spotify)
- Alexander Rass (FAU)
- Fredrik Svensson (Autoliv Electronics)
- Tobias Werth (FAU)

J – Judging

Problem

Given two lists of strings, match them up so that as many as possible are identical.

J – Judging

Problem

Given two lists of strings, match them up so that as many as possible are identical.

Solution

1. Count how many times each word appears in each list.

J – Judging

Problem

Given two lists of strings, match them up so that as many as possible are identical.

Solution

1. Count how many times each word appears in each list.
2. Number of matches for each word is minimum of #occurrences in first list and #occurrences in second list.

J – Judging

Problem

Given two lists of strings, match them up so that as many as possible are identical.

Solution

1. Count how many times each word appears in each list.
2. Number of matches for each word is minimum of #occurrences in first list and #occurrences in second list.
3. Sum over all words.

J – Judging

Problem

Given two lists of strings, match them up so that as many as possible are identical.

Solution

1. Count how many times each word appears in each list.
2. Number of matches for each word is minimum of #occurrences in first list and #occurrences in second list.
3. Sum over all words.

Statistics: 135 submissions, 92 accepted

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:
- Calculate value at some small and large c (e.g. 1 and 100)

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:
- Calculate value at some small and large c (e.g. 1 and 100)
- Calculate value at 1/3 and 2/3 of this interval (i.e. 34 and 67)

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:
- Calculate value at some small and large c (e.g. 1 and 100)
- Calculate value at 1/3 and 2/3 of this interval (i.e. 34 and 67)
- If f(34) $<$ f(68), the minimum is in [1,68], otherwise in [34, 100].

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:
- Calculate value at some small and large c (e.g. 1 and 100)
- Calculate value at 1/3 and 2/3 of this interval (i.e. 34 and 67)
- If f(34) < f(68), the minimum is in [1,68], otherwise in [34, 100].
- Iterate this a few dozen times to narrow down the interval.

E – Euclidean TSP

Problem

Find minimum of the following function:

$$f(c) = \frac{n \cdot (\log_2 n)^{c\sqrt{2}}}{10^9 p} + (1 + \frac{1}{c}) \cdot \frac{s}{v}$$

Solution

- This is a convex function, so use ternary search:
- Calculate value at some small and large c (e.g. 1 and 100)
- Calculate value at 1/3 and 2/3 of this interval (i.e. 34 and 67)
- If f(34) < f(68), the minimum is in [1,68], otherwise in [34, 100].
- Iterate this a few dozen times to narrow down the interval.

Statistics: 91 submissions, 61 accepted

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.

C – Cent Savings

Problem

Split list of n item prices into $d + 1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i, j)$ be minimum cost of first i items using j groups.
- $C(i, j) = \min_{i'<i}\Big\{C(i', j-1) + \text{Round}(x_{i'+1} + \ldots + x_i)\Big\}$

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.
- $C(i,j) = \min_{i'<i}\Big\{C(i',j-1) + \mathrm{Round}(x_{i'+1} + \ldots + x_i)\Big\}$
- Compute with dynamic programming.

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.
- $C(i,j) = \min_{i'<i}\Big\{C(i',j-1) + \mathrm{Round}(x_{i'+1} + \ldots + x_i)\Big\}$
- Compute with dynamic programming.
- As written $O(n^3 d)$ – too slow

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.
- $C(i,j) = \min_{i'<i}\left\{ C(i', j-1) + \text{Round}(x_{i'+1} + \ldots + x_i) \right\}$
- Compute with dynamic programming.
- As written $O(n^3 d)$ – too slow
- Don't recompute sums $O(n^2 d)$ – fast enough

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.
- $C(i,j) = \min_{i'<i}\left\{C(i',j-1) + \text{Round}(x_{i'+1} + \ldots + x_i)\right\}$
- Compute with dynamic programming.
- As written $O(n^3 d)$ – too slow
- Don't recompute sums $O(n^2 d)$ – fast enough
- Be more clever $O(nd)$ – even faster

C – Cent Savings

Problem

Split list of n item prices into $d+1$ groups, minimize sum of rounded costs.

Solution

- Let $C(i,j)$ be minimum cost of first i items using j groups.
- $C(i,j) = \min_{i'<i}\Big\{C(i',j-1) + \text{Round}(x_{i'+1} + \ldots + x_i)\Big\}$
- Compute with dynamic programming.
- As written $O(n^3 d)$ – too slow
- Don't recompute sums $O(n^2 d)$ – fast enough
- Be more clever $O(nd)$ – even faster

Statistics: 216 submissions, 62 accepted

H – Hyacinth

Problem

Given a tree, assign a number (the frequency) to each edge, such that every node is adjacent to at most 2 different numbers (its NIC's frequencies), and as many numbers as possible are used.

H – Hyacinth

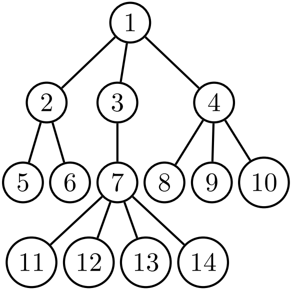
Problem

Given a tree, assign a number (the frequency) to each edge, such that every node is adjacent to at most 2 different numbers (its NIC's frequencies), and as many numbers as possible are used.

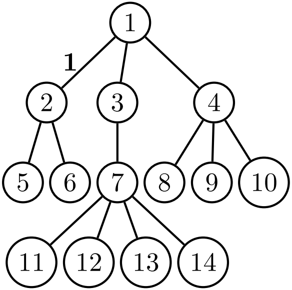
Solution

Greedy: start somewhere, assign a new number to an edge if possible, otherwise reuse one, and then recurse.

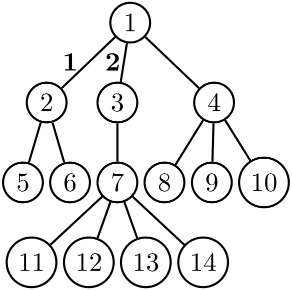
H – Hyacinth (sample 2)



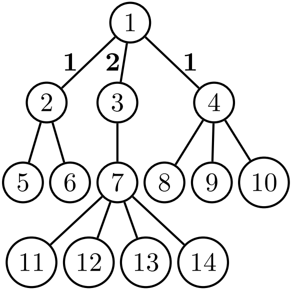
H – Hyacinth (sample 2)



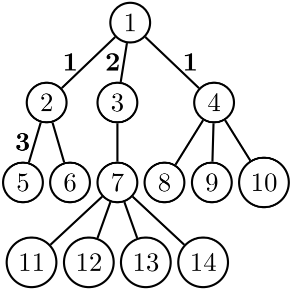
H – Hyacinth (sample 2)



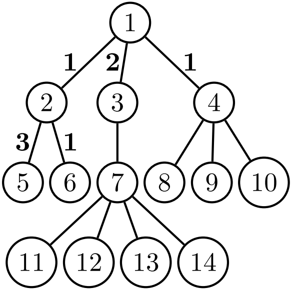
H – Hyacinth (sample 2)



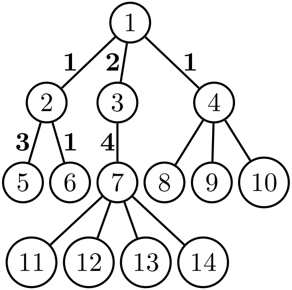
H – Hyacinth (sample 2)



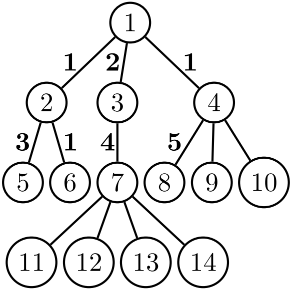
H – Hyacinth (sample 2)



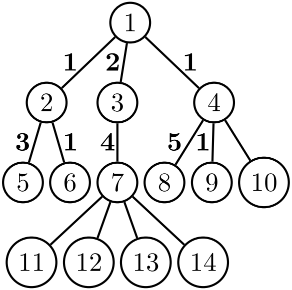
H – Hyacinth (sample 2)



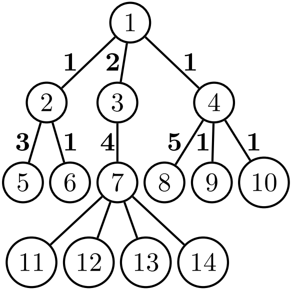
H – Hyacinth (sample 2)



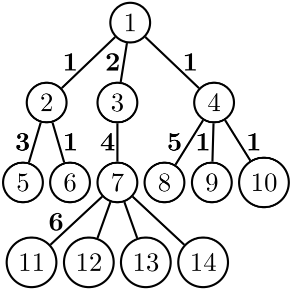
H – Hyacinth (sample 2)



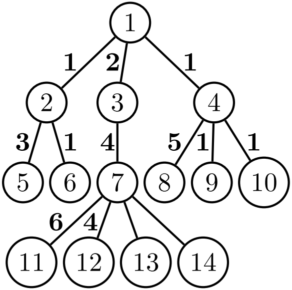
H – Hyacinth (sample 2)



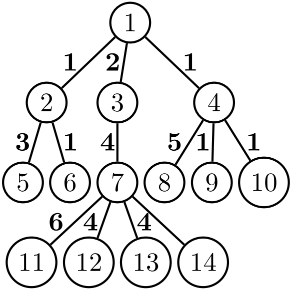
H – Hyacinth (sample 2)



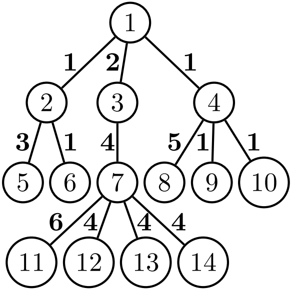
H – Hyacinth (sample 2)



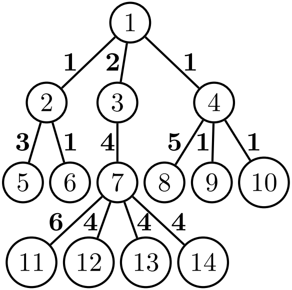
H – Hyacinth (sample 2)



H – Hyacinth (sample 2)

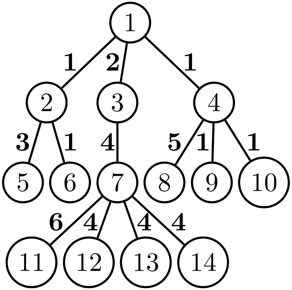


H – Hyacinth (sample 2)



Note: the number of different used channels equals the number of internal nodes + 1.

H – Hyacinth (sample 2)



Note: the number of different used channels equals the number of internal nodes + 1.

Statistics: 213 submissions, 70 accepted

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.
- Probability that a random point lies on the line is $\geq p$.

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.
- Probability that a random point lies on the line is $\geq p$.
- Repeat 250 times:
  1. Pick two distinct points uniformly at random
  2. Check if line defined by the points has enough points.

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.
- Probability that a random point lies on the line is $\geq p$.
- Repeat 250 times:
  1. Pick two distinct points uniformly at random
  2. Check if line defined by the points has enough points.
- Pr[false negative] $\approx (1 - p^2)^{250} \leq (1 - 1/25)^{250} < 5 \cdot 10^{-5}$ (cheated slightly – it's a bit worse)

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.
- Probability that a random point lies on the line is $\geq p$.
- Repeat 250 times:
  1. Pick two distinct points uniformly at random
  2. Check if line defined by the points has enough points.
- Pr[false negative] $\approx (1 - p^2)^{250} \leq (1 - 1/25)^{250} < 5 \cdot 10^{-5}$ (cheated slightly – it's a bit worse)

Many other ways e.g. divide and conquer

F – Finding Lines

Problem

Given n points in $\mathbb{R}^2$, is there a straight line passing through at least a p fraction of them?

Solution

- If line exists it is uniquely determined by any two of its points.
- Probability that a random point lies on the line is $\geq p$.
- Repeat 250 times:
  1. Pick two distinct points uniformly at random
  2. Check if line defined by the points has enough points.
- Pr[false negative] $\approx (1 - p^2)^{250} \leq (1 - 1/25)^{250} < 5 \cdot 10^{-5}$ (cheated slightly – it's a bit worse)

Many other ways e.g. divide and conquer

Statistics: 222 submissions, 23 accepted

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

Solution

- Don't simulate 10^{18} balls one by one!

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

Solution

- Don't simulate 10^{18} balls one by one!
- Observation: if N balls arrive at a node, $\lceil N/2 \rceil$ go to one side, and $\lfloor N/2 \rfloor$ to the other side, and its state is flipped iff N is odd.

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

Solution

- Don't simulate 10^{18} balls one by one!
- Observation: if N balls arrive at a node, $\lceil N/2 \rceil$ go to one side, and $\lfloor N/2 \rfloor$ to the other side, and its state is flipped iff N is odd.
- Process nodes one by one

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

Solution

- Don't simulate 10^{18} balls one by one!
- Observation: if N balls arrive at a node, $\lceil N/2 \rceil$ go to one side, and $\lfloor N/2 \rfloor$ to the other side, and its state is flipped iff N is odd.
- Process nodes one by one
- Do it in *topological sort* order.

D – Digi Comp II

Problem

Given a directed acyclic graph, where the nodes are switches that send balls to the left and right in turns, what is its end state.

Solution

- Don't simulate 10^{18} balls one by one!
- Observation: if N balls arrive at a node, $\lceil N/2 \rceil$ go to one side, and $\lfloor N/2 \rfloor$ to the other side, and its state is flipped iff N is odd.
- Process nodes one by one
- Do it in *topological sort* order.

Statistics: 327 submissions, 43 accepted

K – Knapsack Collection

Problem

How long does it take to pick up all n knapsacks at the luggage carousel?

K – Knapsack Collection

Problem

How long does it take to pick up all n knapsacks at the luggage carousel?

Solution

- Given starting position, can simulate the process in $O(n \log n)$
  - Keep positions of knapsacks in a set for fast lookup of when the next knapsack will arrive.

K – Knapsack Collection

Problem

How long does it take to pick up all n knapsacks at the luggage carousel?

Solution

- Given starting position, can simulate the process in $O(n \log n)$
  - Keep positions of knapsacks in a set for fast lookup of when the next knapsack will arrive.
- The interesting starting positions are the $\leq n$ slots where there are knapsacks.

K – Knapsack Collection

Problem

How long does it take to pick up all n knapsacks at the luggage carousel?

Solution

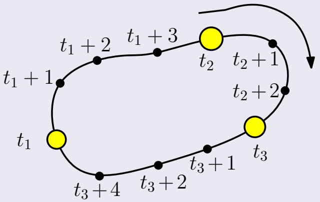
- Given starting position, can simulate the process in $O(n \log n)$
  - Keep positions of knapsacks in a set for fast lookup of when the next knapsack will arrive.
- The interesting starting positions are the $\leq n$ slots where there are knapsacks.
- Time from others grow arithmetically

K – Knapsack Collection

Problem

How long does it take to pick up all *n* knapsacks at the luggage carousel?

Solution

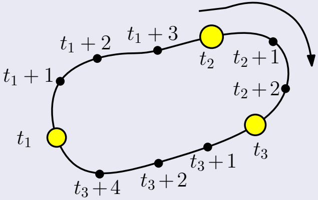


K – Knapsack Collection

Problem

How long does it take to pick up all n knapsacks at the luggage carousel?

Solution



Statistics: 68 submissions, 22 accepted

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow ($14! = 87 \cdot 10^9$).

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow $(14! = 87 \cdot 10^9)$.
- Take 0 as starting point and loop over middle point: $\mathcal{O}(N)$.

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow ($14! = 87 \cdot 10^9$).
- Take 0 as starting point and loop over middle point: $\mathcal{O}(N)$.
- Partition remaining points into 1st and 2nd half: $\mathcal{O}(2^N)$.

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow ($14! = 87 \cdot 10^9$).
- Take 0 as starting point and loop over middle point: $\mathcal{O}(N)$.
- Partition remaining points into 1st and 2nd half: $\mathcal{O}(2^N)$.
- Calculate possible lengths of half-cycles: $\mathcal{O}((N/2)!)$.

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow ($14! = 87 \cdot 10^9$).
- Take 0 as starting point and loop over middle point: $\mathcal{O}(N)$.
- Partition remaining points into 1st and 2nd half: $\mathcal{O}(2^N)$.
- Calculate possible lengths of half-cycles: $\mathcal{O}((N/2)!)$.
- For every half-cycle length K, check whether $L - K$ exists in $\mathcal{O}(1)$ with hashset or so.

I – Indoorienteering

Problem

Given a weighted graph, determine whether there exists a cycle with length L that visits every vertex once.

Solution

- First of all, $\mathcal{O}(N!)$ is too slow $(14! = 87 \cdot 10^9)$.
- Take 0 as starting point and loop over middle point: $\mathcal{O}(N)$.
- Partition remaining points into 1st and 2nd half: $\mathcal{O}(2^N)$.
- Calculate possible lengths of half-cycles: $\mathcal{O}((N/2)!)$.
- For every half-cycle length K, check whether $L - K$ exists in $\mathcal{O}(1)$ with hashset or so.

Statistics: 79 submissions, 7 accepted

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- The "ideal" position is $x^* = \text{median}(x_1, \ldots, x_n)$ and $y^* = \text{median}(y_1, \ldots, y_n)$

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- The "ideal" position is $x^* = \text{median}(x_1, \ldots, x_n)$ and $y^* = \text{median}(y_1, \ldots, y_n)$
- But might be more than distance d away from some point...

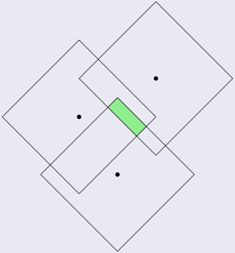
G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

Feasible region is intersection of diamonds, forms a "skewed diamond"



G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- Key observation: there is an optimal point which either:
  1. Is a corner of the feasible region, or
  2. Shares x-value with an input point, or
  3. Shares y-value with an input point.

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- Key observation: there is an optimal point which either:
  1. Is a corner of the feasible region, or
  2. Shares x-value with an input point, or
  3. Shares y-value with an input point.
- Given a candidate value for x^*, we get a range of allowed y values $y_{lo}...y_{hi}$. Best choice is median y (if in range) or y_{lo} or y_{hi}.

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- Key observation: there is an optimal point which either:
  1. Is a corner of the feasible region, or
  2. Shares x-value with an input point, or
  3. Shares y-value with an input point.
- Given a candidate value for x^*, we get a range of allowed y values $y_{lo} ... y_{hi}$. Best choice is median y (if in range) or y_{lo} or y_{hi}.
- (Similarly for best x value given candindate value of y^*)

G – Gathering

Problem

Given points $(x_1, y_1), \ldots, (x_n, y_n)$, find point (x^*, y^*) minimizing sum of Manhattan distances to all points, while being within Manhattan distance d of all points.

Solution

- Key observation: there is an optimal point which either:
    1. Is a corner of the feasible region, or
    2. Shares x-value with an input point, or
    3. Shares y-value with an input point.
- Given a candidate value for x^*, we get a range of allowed y values $y_{lo} \ldots y_{hi}$. Best choice is median y (if in range) or y_{lo} or y_{hi}.
- (Similarly for best x value given candindate value of y^*)

Statistics: 20 submissions, ? accepted

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

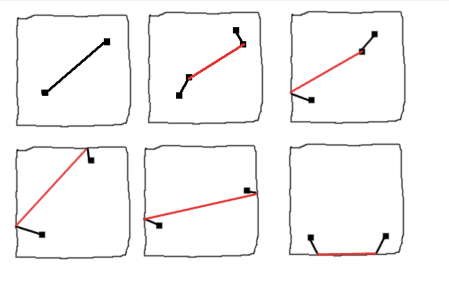
B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

Possible solutions use 0 or 2 stations:



B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).
- Calculate distance.

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).
- Calculate distance.
- To determine correct location on the boundary use ternary search, since distance as function of position is a convex function.

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).
- Calculate distance.
- To determine correct location on the boundary use ternary search, since distance as function of position is a convex function.
- For 2 boundary stations, use nested ternary searches.

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).
- Calculate distance.
- To determine correct location on the boundary use ternary search, since distance as function of position is a convex function.
- For 2 boundary stations, use nested ternary searches.
- Doing it with math becomes a terrible mess.

B – Biking Duck

Problem

Find shortest path, possibly via some of the given bike stations or the (infinitely many possible) stations the boundary.

Solution

- Loop over station 1 and 2 (existing, or NSEW boundary).
- Calculate distance.
- To determine correct location on the boundary use ternary search, since distance as function of position is a convex function.
- For 2 boundary stations, use nested ternary searches.
- Doing it with math becomes a terrible mess.

Statistics: 7 submissions, ? accepted

A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.

A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.

Solution

- Without the outer boundary: convex hull

A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.

Solution

- Without the outer boundary: convex hull
- A somewhat unusual convex hull algorithm.
  1. Find a point where we make a right-turn, shortcut if possible
  2. Repeat until done

A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.

Solution

- Without the outer boundary: convex hull
- A somewhat unusual convex hull algorithm.
  1. Find a point where we make a right-turn, shortcut if possible
  2. Repeat until done
- With outer boundary: when shortcutting, can't go straight, need to wrap around the outer boundary
  - Convex hull computation

A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.

Solution

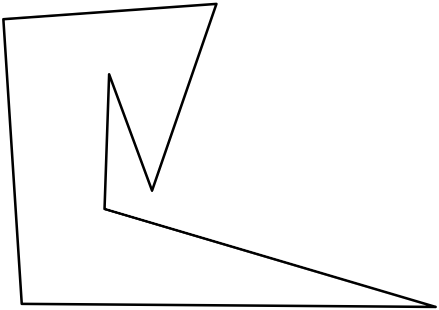
- Without the outer boundary: convex hull
- A somewhat unusual convex hull algorithm.
  1. Find a point where we make a right-turn, shortcut if possible
  2. Repeat until done
- With outer boundary: when shortcutting, can't go straight, need to wrap around the outer boundary
  - Convex hull computation

Can also use Dijkstra, need to be careful to make a full lap around the inner polygon.

A – Around the Track

Problem

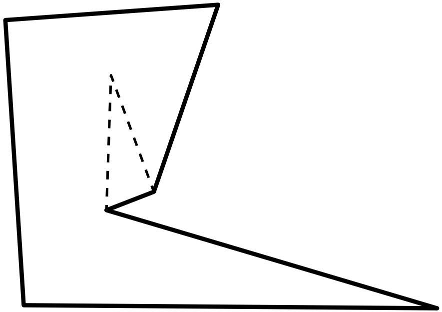
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

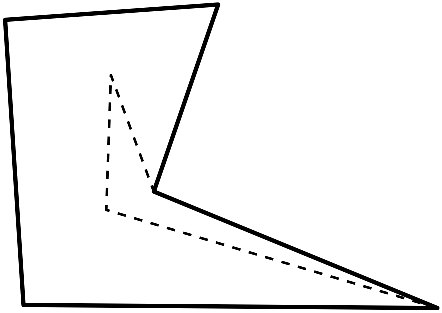
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

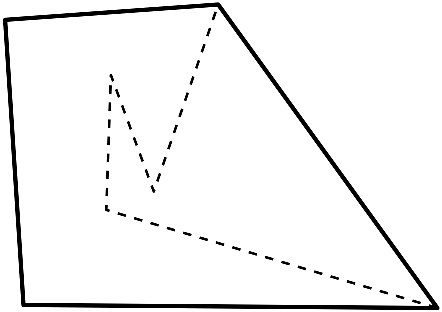
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

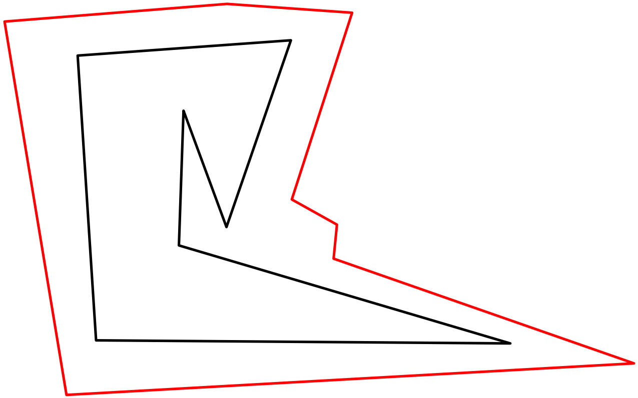
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

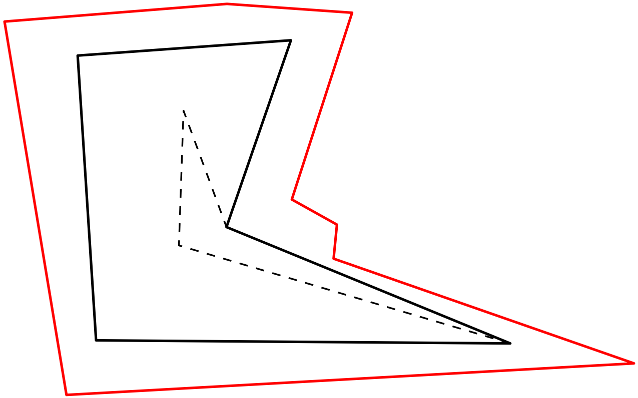
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

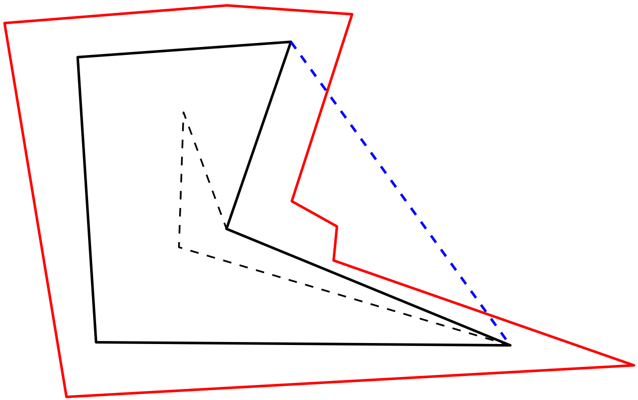
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

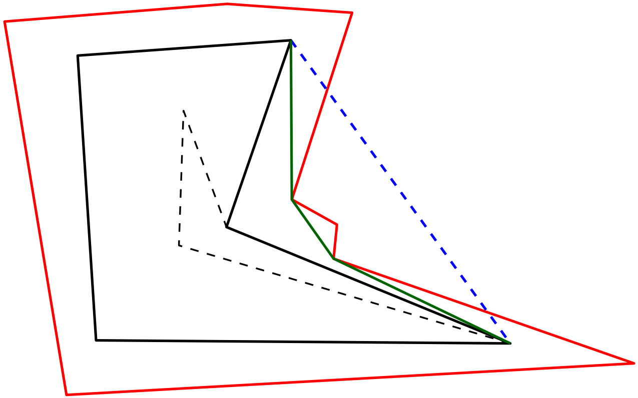
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

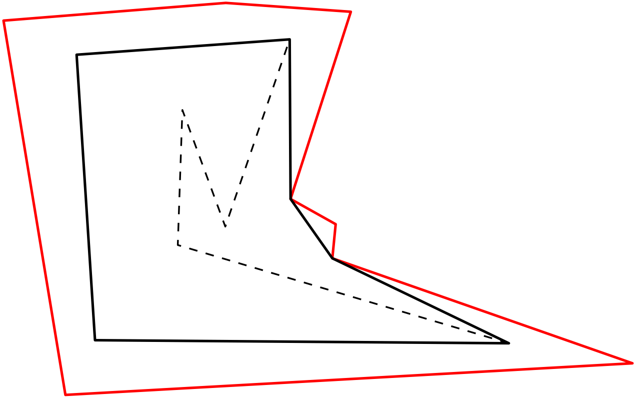
Find shortest tour around a track defined by two polygons.



A – Around the Track

Problem

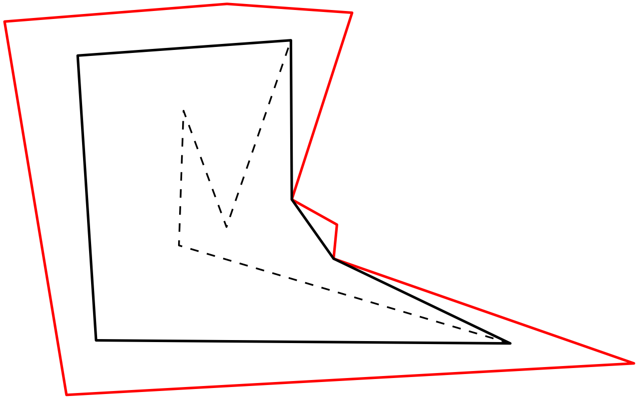
Find shortest tour around a track defined by two polygons.



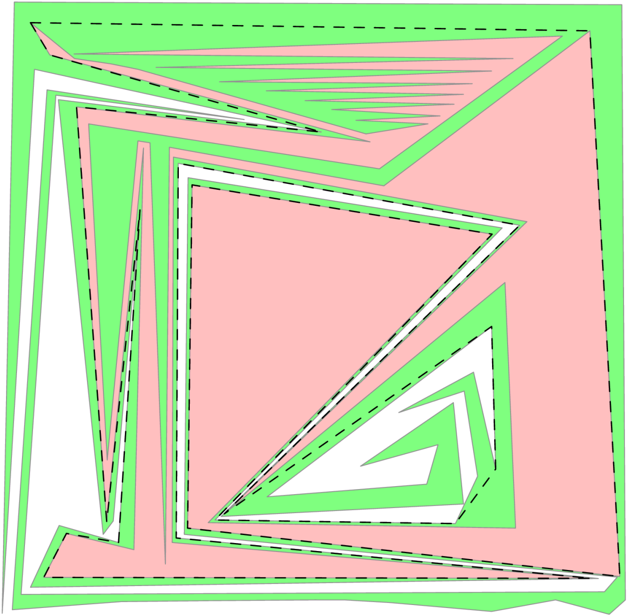
A – Around the Track

Problem

Find shortest tour around a track defined by two polygons.



Statistics: 12 submissions, ? accepted



Random numbers produced by the jury

1087 number of posts made in the jury's forum.

671 commits made to the problem set repository.

380 number of lines of code used in total by the shortest judge solutions to solve the entire problem set.

16.7 average number of jury solutions per problem (including incorrect ones)

1 number of beers Jan lost to Per over bets on the problems

Random numbers produced by the jury

- **1087** number of posts made in the jury's forum.
- **671** commits made to the problem set repository.
- **380** number of lines of code used in total by the shortest judge solutions to solve the entire problem set.
- **16.7** average number of jury solutions per problem (including incorrect ones)
- **1** number of beers Jan lost to Per over bets on the problems
- **1** number of beers Per lost to Jan over bets on the contest results