Finding the biggest and smallest element with one lie

D. Gerbner[2], D. Pálvölgyi[1], B. Patkós[1] and G. Wiener[3]

1 Eötvös University (ELTE), Budapest
2 A. Rényi Institute of Mathematics, Budapest
3 University of Technology and Economics (BME), Budapest

International Conference on Interdisciplinary Mathematical and Statistical Techniques (IMST 2008 / FIM XVI) at University of Memphis, May 15–18.

What is a search game?

What is a search game?

20 questions game - Find an element of a given set using Yes-No questions.

What is a search game?

20 questions game - Find an element of a given set using Yes-No questions.

Generally - Decide if an element of a set has a property using certain questions.

What is a search game?

20 questions game - Find an element of a given set using Yes-No questions.

Generally - Decide if an element of a set has a property using certain questions.

Faulty coin problem - 27 coins, one is lighter, find it using 3 measurements on a scale.

What is a search game?

20 questions game - Find an element of a given set using Yes-No questions.

Generally - Decide if an element of a set has a property using certain questions.

Faulty coin problem - 27 coins, one is lighter, find it using 3 measurements on a scale.

Lies - Some of the answers might be erroneous.

What is a search game?

20 questions game - Find an element of a given set using Yes-No questions.

Generally - Decide if an element of a set has a property using certain questions.

Faulty coin problem - 27 coins, one is lighter, find it using 3 measurements on a scale.

Lies - Some of the answers might be erroneous.

In our model - Fixed number of lies, best worst-case algorithm, questions can depend on previous answers.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting:

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest:

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Finding the biggest and the smallest:

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Finding the biggest and the smallest: $\lceil \frac{3n}{2} \rceil - 2$.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Finding the biggest and the smallest: $\lceil \frac{3n}{2} \rceil - 2$.

Finding the biggest with k lies:

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Finding the biggest and the smallest: $\lceil \frac{3n}{2} \rceil - 2$.

Finding the biggest with k lies: $(k+1)(n-1) + k$.

Our model

We are given an ordered set of size n (with no equal elements) and we can use pairwise comparisons.

Sorting: $n \log n + \Theta(n)$.

Finding the biggest: $n - 1$.

Finding the biggest and the smallest: $\lceil \frac{3n}{2} \rceil - 2$.

Finding the biggest with k lies: $(k+1)(n-1) + k$.

Finding the biggest and the smallest with k lies - Our topic.

Finding the biggest and the smallest with k lies

Earlier examined by Martin Aigner.

Finding the biggest and the smallest with k lies

Earlier examined by Martin Aigner.

For general k: $\leq (k + \Theta(\sqrt{k}))n + \Theta_k(1)$.

Finding the biggest and the smallest with k lies

Earlier examined by Martin Aigner.

For general k: $\leq (k + \Theta(\sqrt{k}))n + \Theta_k(1)$.

For $k = 1$: $\leq \frac{87}{32}n + \Theta(1)$.

Finding the biggest and the smallest with k lies

Earlier examined by Martin Aigner.

For general k: $\leq (k + \Theta(\sqrt{k}))n + \Theta_k(1)$.

For $k = 1$: $\leq \frac{87}{32}n + \Theta(1)$.

Our main result.

Finding the biggest and the smallest with k lies

Earlier examined by Martin Aigner.

For general k: $\leq (k + \Theta(\sqrt{k}))n + \Theta_k(1)$.

For $k = 1$: $\leq \frac{87}{32}n + \Theta(1)$.

Our main result.

For $k = 1$: $\frac{87}{32}n + \Theta(1)$.

Terminology

Terminology

teams - elements of the set

Terminology

teams - elements of the set

match - comparison

Terminology

teams - elements of the set

match - comparison

win - bigger in comparison

Terminology

teams - elements of the set

match - comparison

win - bigger in comparison

lose - smaller in comparison

Terminology

teams - elements of the set

match - comparison

win - bigger in comparison

lose - smaller in comparison

championship graph - digraph, ab is an edge iff a has beaten b

Finding the biggest with k lies

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound:

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound:

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough:

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough: After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best.

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough: After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best. Trivially???

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough: After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best. Trivially??? If there is a directed cycle in the graph, then no!

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough:
After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best. Trivially???
If there is a directed cycle in the graph, then no!
But if the Adversary never lies, then yes.

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough:
After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best. Trivially???
If there is a directed cycle in the graph, then no!
But if the Adversary never lies, then yes.

Rule of thumb - The evil Adversary never lies.

Finding the biggest with k lies

Claim. Requires exactly $(k+1)(n-1)+k$.

Upper bound: Ask everything you would ask without lies until you get the same answer $k+1$ times.

Easy Lower bound: $< (k+1)(n-1)$ questions are not enough:
After $< (k+1)(n-1)$ questions, there are at least two teams with in-degree $\leq k$, trivially any of these can be the best. Trivially???
If there is a directed cycle in the graph, then no!
But if the Adversary never lies, then yes.

Rule of thumb - The evil Adversary never lies. Except maybe at the end.

Finding the biggest with k lies

Claim. This requires $(k+1)(n-1)+k$.
Rule of thumb - The evil Adversary never lies. Except maybe at the end.

Lower bound: $< (k+1)(n-1)+k$ questions are not enough:

Finding the biggest with k lies

Claim. This requires $(k+1)(n-1)+k$.
Rule of thumb - The evil Adversary never lies. Except maybe at the end.

Lower bound: $< (k+1)(n-1)+k$ questions are not enough:

Reply without lies until there are only two teams with in-degree $\leq k$ and one of them has in-degree 0, the other k.

Finding the biggest with k lies

Claim. This requires $(k+1)(n-1)+k$.
Rule of thumb - The evil Adversary never lies. Except maybe at the end.

Lower bound: $< (k+1)(n-1)+k$ questions are not enough:

Reply without lies until there are only two teams with in-degree $\leq k$ and one of them has in-degree 0, the other k.

After this we "change our mind" and reply such that the team with in-degree k always wins.

Finding the biggest and smallest without lies

Finding the biggest and smallest without lies

Proof is folklore

Finding the biggest and smallest without lies

Proof is folklore - give every team a red and a blue pebble.

Finding the biggest and smallest without lies

Proof is folklore - give every team a red and a blue pebble.

Take the red if it loses, take the blue if it wins.

Finding the biggest and smallest without lies

Proof is folklore - give every team a red and a blue pebble.

Take the red if it loses, take the blue if it wins.

Game ends when there is only one pebble of each color.

Finding the biggest and smallest without lies

Proof is folklore - give every team a red and a blue pebble.

Take the red if it loses, take the blue if it wins.

Game ends when there is only one pebble of each color.

Adversary can answer such that only matches between two newbies result in loss of two pebbles

Finding the biggest and smallest without lies

Proof is folklore - give every team a red and a blue pebble.

Take the red if it loses, take the blue if it wins.

Game ends when there is only one pebble of each color.

Adversary can answer such that only matches between two newbies result in loss of two pebbles $\Rightarrow \frac{3}{2}n - 2$ questions are needed.

Algorithm for k lies

Algorithm for k lies

Give every team $k+1$ red and $k+1$ blue pebbles.

Algorithm for k lies

Give every team $k+1$ red and $k+1$ blue pebbles.

Match teams with the same number of pebbles until possible.

Algorithm for k lies

Give every team $k+1$ red and $k+1$ blue pebbles.

Match teams with the same number of pebbles until possible.

After that only $\Theta_k(1)$ teams remain.

Algorithm for k lies

Give every team $k+1$ red and $k+1$ blue pebbles.

Match teams with the same number of pebbles until possible.

After that only $\Theta_k(1)$ teams remain.

How many comparisons are needed?

Potential function p

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$p(0, 0) = 0$,

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

$$2p(a, b) = p(a - 1, b) + p(a, b - 1) + 1 \text{ for } a, b > 0.$$

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

$$2p(a, b) = p(a - 1, b) + p(a, b - 1) + 1 \text{ for } a, b > 0.$$

$\sum p(a, b) + \Theta_k(1)$ questions are always enough.

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

$$2p(a, b) = p(a - 1, b) + p(a, b - 1) + 1 \text{ for } a, b > 0.$$

$\sum p(a, b) + \Theta_k(1)$ questions are always enough.

Following a little magic with the binomial coefficients:
$p(a, a) = a(1 + \binom{2a}{a} 2^{-2a})$.

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

$$2p(a, b) = p(a - 1, b) + p(a, b - 1) + 1 \text{ for } a, b > 0.$$

$\sum p(a, b) + \Theta_k(1)$ questions are always enough.

Following a little magic with the binomial coefficients:
$p(a, a) = a(1 + \binom{2a}{a} 2^{-2a})$.

So $(k + \Theta(\sqrt{k}))n + \Theta_k(1)$ questions suffice.

Potential function p

$p(a, b)$ measures number of questions needed in average for a team with a red and b blue pebbles.

$$p(0, 0) = 0, \qquad p(a, 0) = p(0, a) = a,$$

$$2p(a, b) = p(a - 1, b) + p(a, b - 1) + 1 \text{ for } a, b > 0.$$

$\sum p(a, b) + \Theta_k(1)$ questions are always enough.

Following a little magic with the binomial coefficients:
$p(a, a) = a(1 + \binom{2a}{a} 2^{-2a})$.

So $(k + \Theta(\sqrt{k}))n + \Theta_k(1)$ questions suffice. - Not optimal.

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

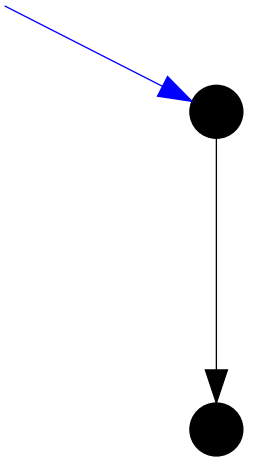
Make a cycle unless $\sum p(a, b)$ decreases.

Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

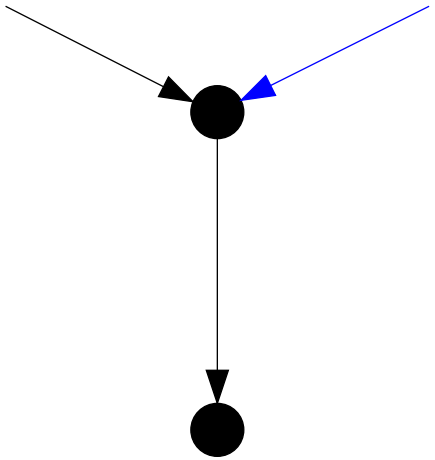


Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

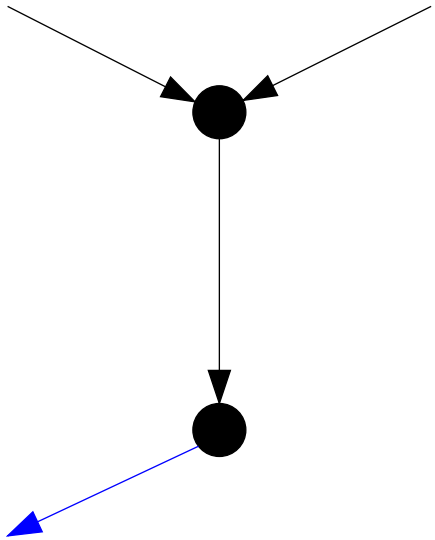


Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

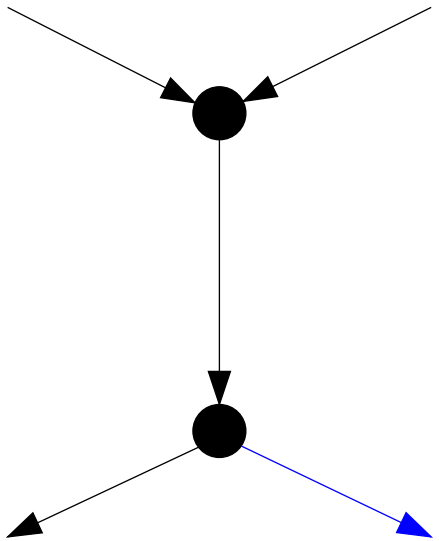


Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.



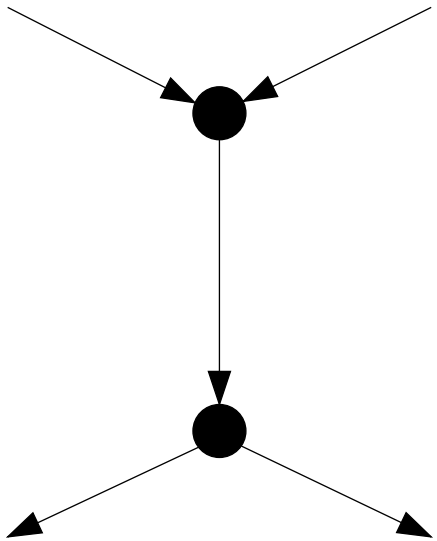
Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

The p of upper and lower team are both 1.



Main idea for 1 lie

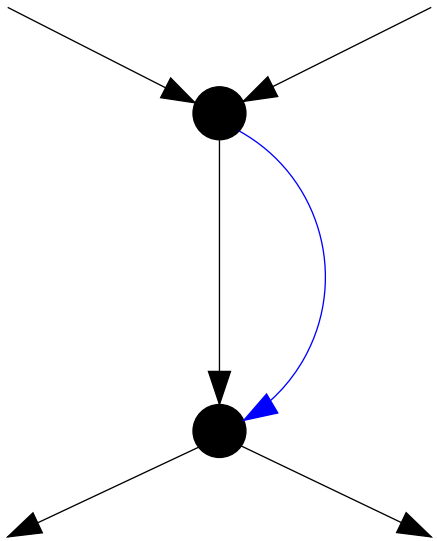
How to improve on the $2.75n$ of the previous algorithm?

Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

The p of upper and lower team are both 1.

If they play, $\sum p(a, b)$ decreases by 2



Main idea for 1 lie

How to improve on the $2.75n$ of the previous algorithm?

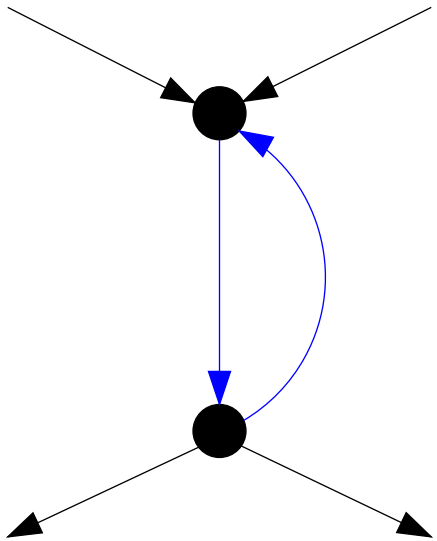
Cannot be improved without forcing cycles.

Make a cycle unless $\sum p(a, b)$ decreases.

The p of upper and lower team are both 1.

If they play, $\sum p(a, b)$ decreases by 2

or a lie is discovered.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

Algorithm for 1 lie

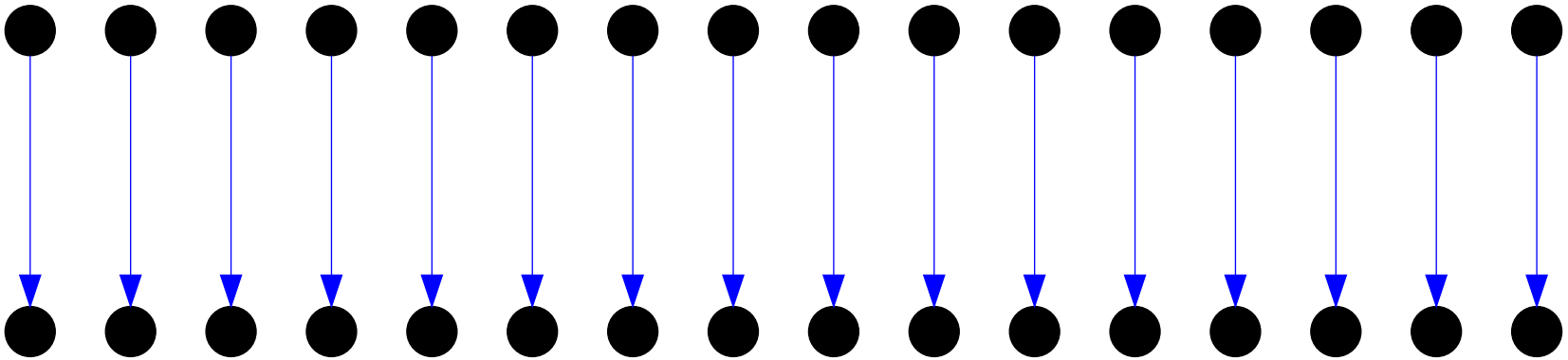
Uses $(2.75 - \frac{1}{32})n$ questions.

Groups of 32, spare a comparison in each.

Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

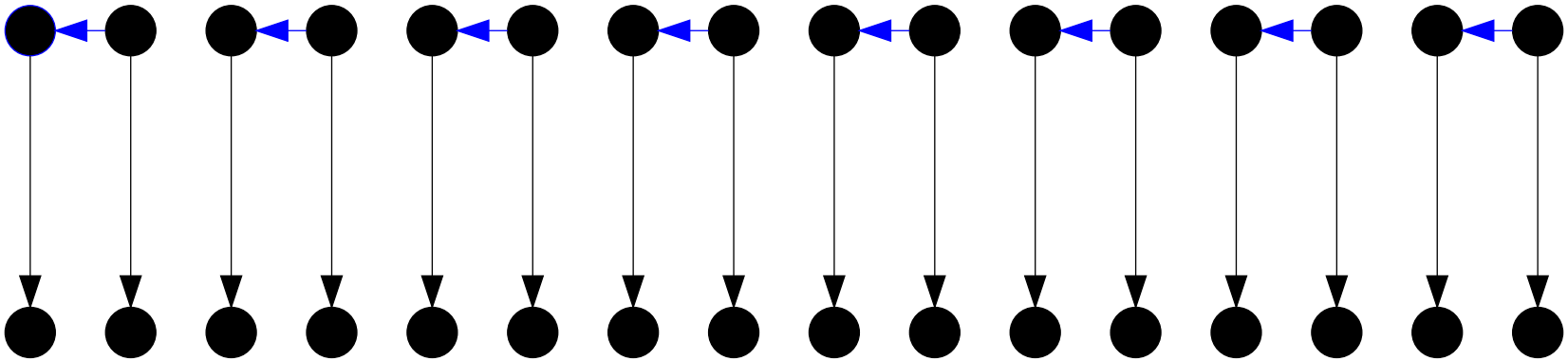
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

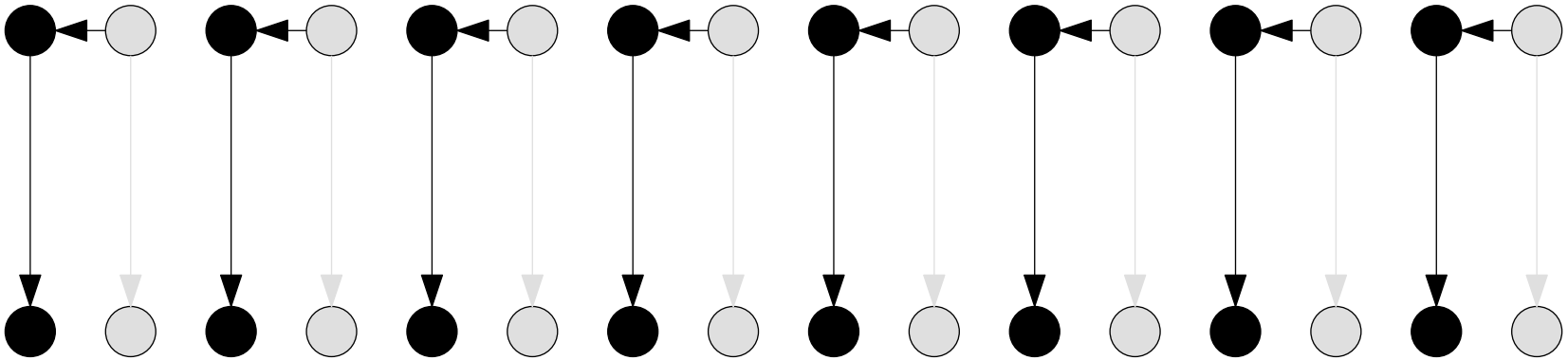
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

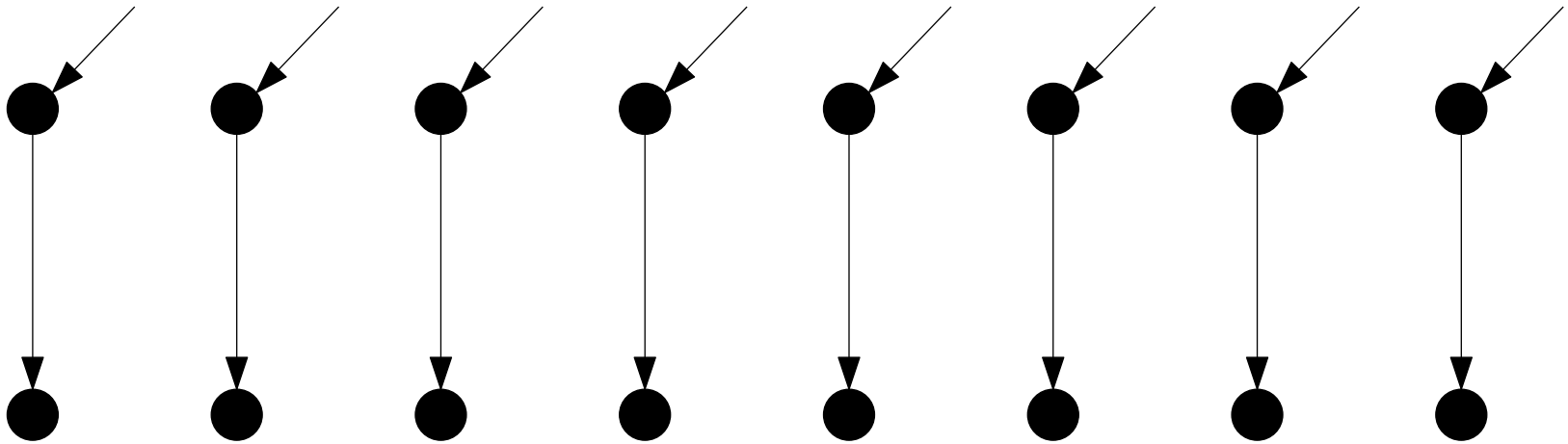
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

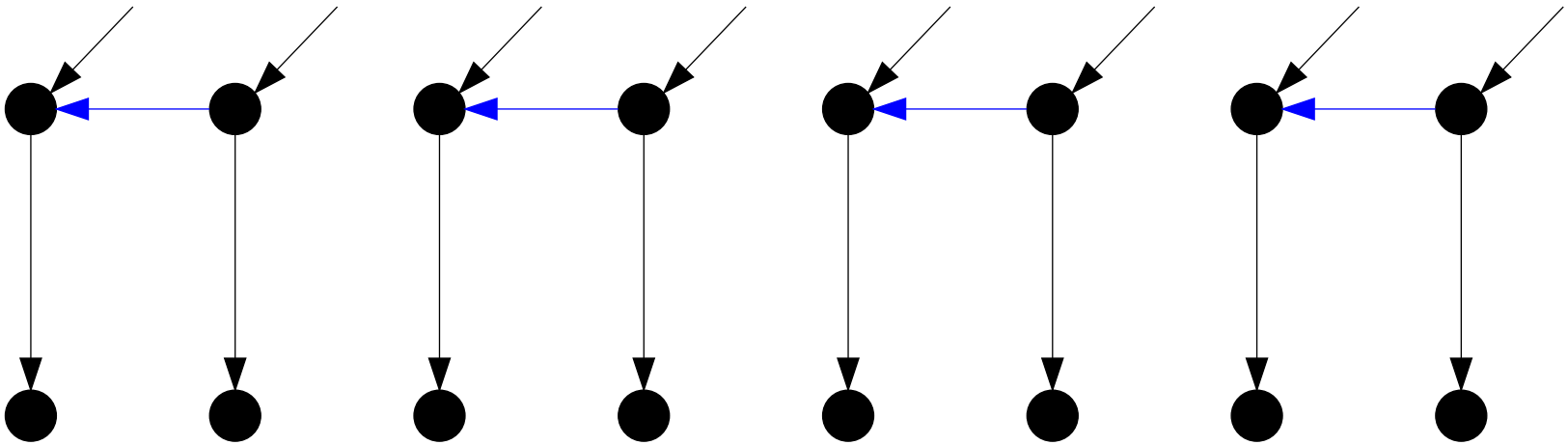
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

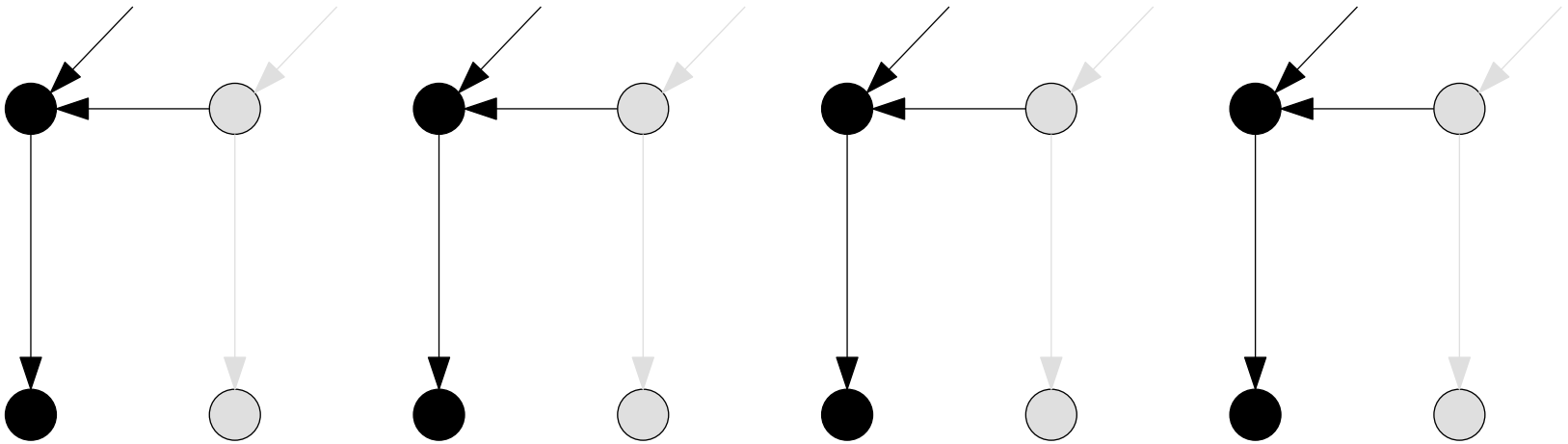
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

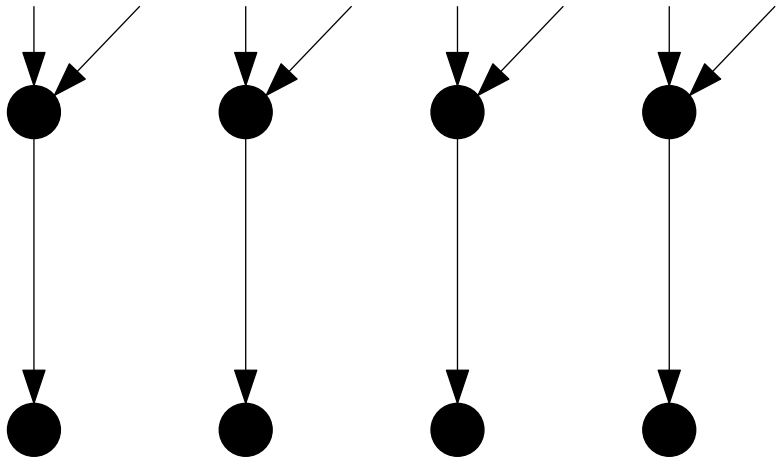
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

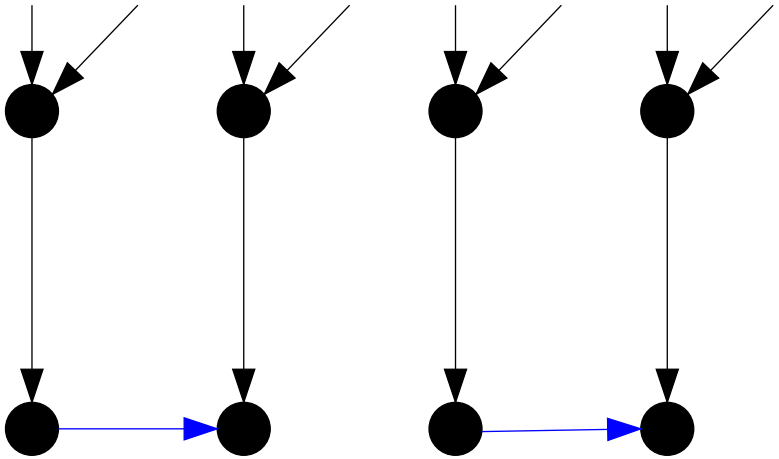
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

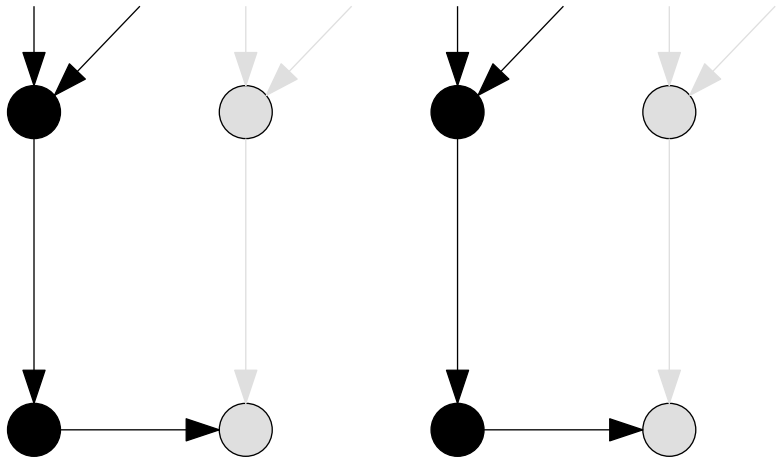
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

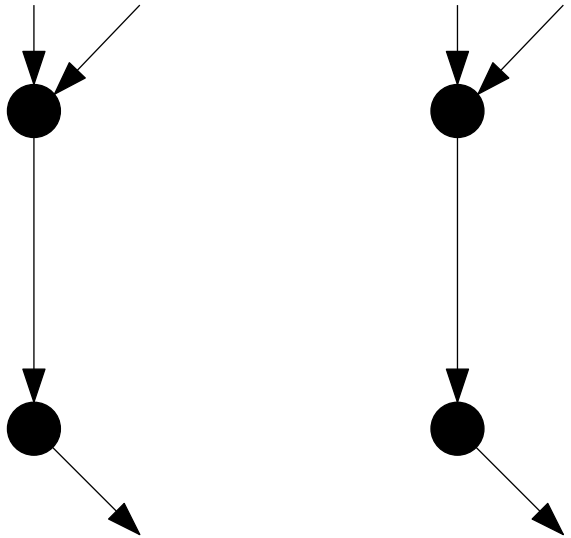
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

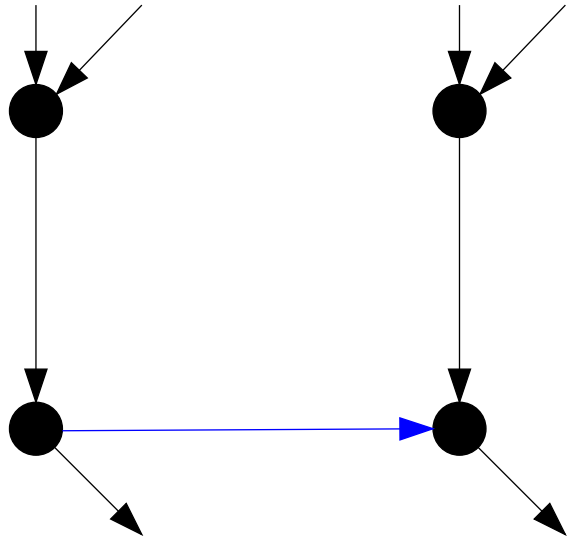
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

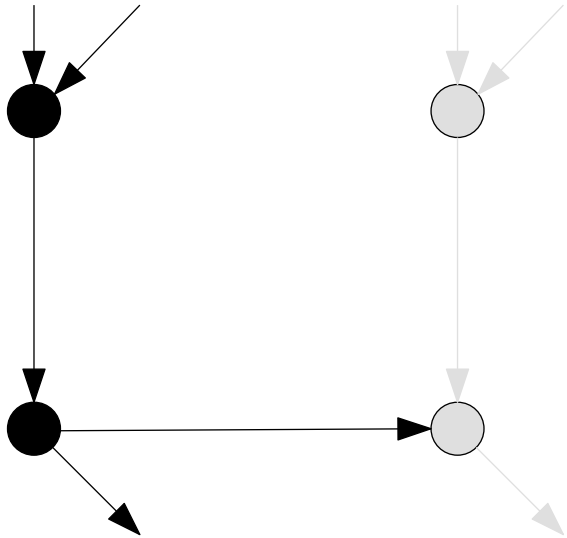
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

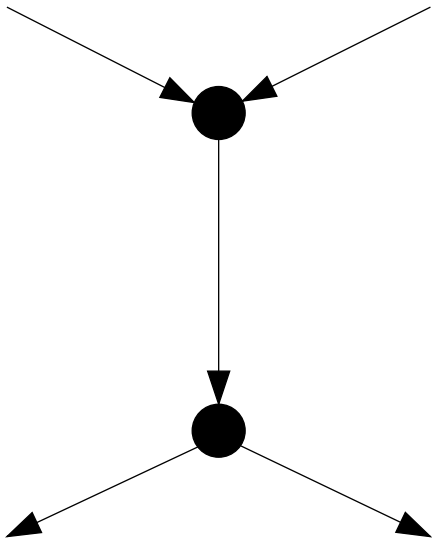
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

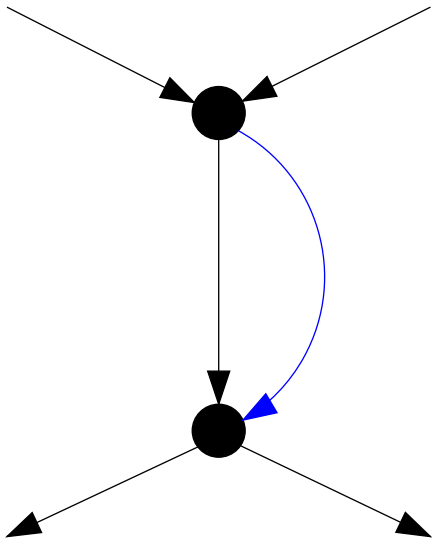
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

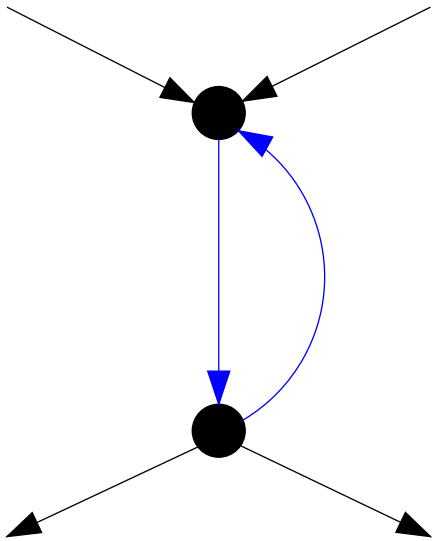
Groups of 32, spare a comparison in each.



Algorithm for 1 lie

Uses $(2.75 - \frac{1}{32})n$ questions.

Groups of 32, spare a comparison in each.



Modified potential

To obtain a matching lower bound, modify p.

Modified potential

To obtain a matching lower bound, modify p.

$$c(x,y) = \begin{cases} 2^{-5} & \text{if } x = y \text{ and have not played yet,} \\ 0 & \text{if } x \text{ was not the first opponent of } y, \\ 2^{losses(x)+wins(y)-4} & \text{if } x \text{ beat } y \text{ in their first game,} \\ & \text{and } wins(x) = 1 \text{ and } losses(y) = 1. \end{cases}$$

Modified potential

To obtain a matching lower bound, modify p.

$$c(x,y) = \begin{cases} 2^{-5} & \text{if } x = y \text{ and have not played yet,} \\ 0 & \text{if } x \text{ was not the first opponent of } y, \\ 2^{losses(x)+wins(y)-4} & \text{if } x \text{ beat } y \text{ in their first game,} \\ & \text{and } wins(x) = 1 \text{ and } losses(y) = 1. \end{cases}$$

Examples.

Modified potential

To obtain a matching lower bound, modify p.

$$c(x,y) = \begin{cases} 2^{-5} & \text{if } x = y \text{ and have not played yet,} \\ 0 & \text{if } x \text{ was not the first opponent of } y, \\ 2^{losses(x)+wins(y)-4} & \text{if } x \text{ beat } y \text{ in their first game,} \\ & \text{and } wins(x) = 1 \text{ and } losses(y) = 1. \end{cases}$$

Examples. If in their first game x beats y: $c(x,y) = 2^{-4}$.

Modified potential

To obtain a matching lower bound, modify p.

$$c(x,y) = \begin{cases} 2^{-5} & \text{if } x = y \text{ and have not played yet,} \\ 0 & \text{if } x \text{ was not the first opponent of } y, \\ 2^{losses(x)+wins(y)-4} & \text{if } x \text{ beat } y \text{ in their first game,} \\ & \text{and } wins(x) = 1 \text{ and } losses(y) = 1. \end{cases}$$

Examples. If in their first game x beats y: $c(x,y) = 2^{-4}$.
If later x looses twice and y wins twice, then $c(x,y) = 1$.

Adversary's strategy

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy:

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy:

1. Never lie.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy:

1. Never lie.
2. Decrease $\sum p(x) - c(x, y)$ by at most 1 after each question.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy:

1. Never lie.
2. Decrease $\sum p(x) - c(x, y)$ by at most 1 after each question.
3. Every opponent but the first of each active team is inactive.

Adversary's strategy

Champion's League - Teams who have won two games and were beaten only by teams in Champion's League.
Second Division - Teams who have lost two games and won only against teams in Second Division.
Active - If not in Champion's League or Second Division.

Adversary's strategy:

1. Never lie.
2. Decrease $\sum p(x) - c(x, y)$ by at most 1 after each question.
3. Every opponent but the first of each active team is inactive.

Proof is by analyzing a lot of cases.

Thank you for your attention!