H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

Solution? Based on the Naive Solution, try to gradually improve the LIS to n, but use fewer queries than random operations.

Number the elements from 1 to n from left to right (although this numbering may differ from the hidden permutation $p_1, \ldots, p_n$). Then store the array a_i which is the current permutation in your own numbering system.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

Solution? Based on the Naive Solution, try to gradually improve the LIS to n, but use fewer queries than random operations.

Number the elements from 1 to n from left to right (although this numbering may differ from the hidden permutation $p_1, \ldots, p_n$). Then store the array a_i which is the current permutation in your own numbering system.

Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere (using operations i 1, 1 2, 2 3, $\ldots$, $n-1$ n)

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

Solution? Based on the Naive Solution, try to gradually improve the LIS to n, but use fewer queries than random operations.

Number the elements from 1 to n from left to right (although this numbering may differ from the hidden permutation $p_1, \ldots, p_n$). Then store the array a_i which is the current permutation in your own numbering system.

Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere (using operations i 1, 1 2, 2 3, $\ldots$, $n-1$ n)

Then choose to place $a_i = x$ at position which gave the maximum LIS.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

Solution? Based on the Naive Solution, try to gradually improve the LIS to n, but use fewer queries than random operations.

Number the elements from 1 to n from left to right (although this numbering may differ from the hidden permutation $p_1, \ldots, p_n$). Then store the array a_i which is the current permutation in your own numbering system.

Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere (using operations i 1, 1 2, 2 3, $\ldots$, $n-1$ n)

Then choose to place $a_i = x$ at position which gave the maximum LIS.

Operations: This takes $n(n+1) < 4n^2$ operations, which easily fits within the limits.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Problem: There is a hidden permutation $p_1, \ldots, p_n$. In one step, you are allowed to remove an element from position i, and insert it at position j instead. After each operation you get told the length of a Longest Increasing Subsequence (LIS). Sort the permutation in at most $4n^2$ operations.

Naive Idea: Keep trying random pairs i j, and undoing the moves which do not increase the LIS, until LIS $= n$. You can undo an i j operation with the operation j i. This takes too many queries.

Solution? Based on the Naive Solution, try to gradually improve the LIS to n, but use fewer queries than random operations.

Number the elements from 1 to n from left to right (although this numbering may differ from the hidden permutation $p_1, \ldots, p_n$). Then store the array a_i which is the current permutation in your own numbering system.

Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere (using operations i 1, 1 2, 2 3, $\ldots$, $n-1$ n)

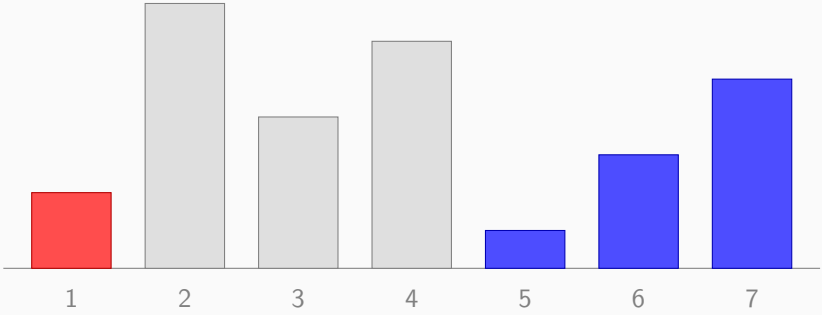
Then choose to place $a_i = x$ at position which gave the maximum LIS.

Operations: This takes $n(n+1) < 4n^2$ operations, which easily fits within the limits.

But why does this algorithm sort the permutation in the end?

H: Historical Hierarchy

Problem author: Reinier Schmiermann

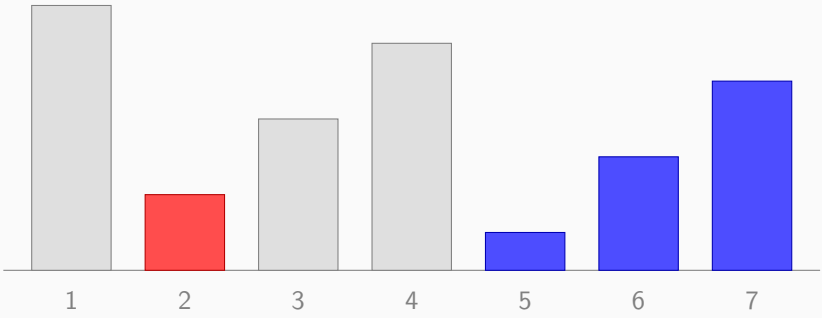


LIS: 3

1 1

H: Historical Hierarchy

Problem author: Reinier Schmiermann

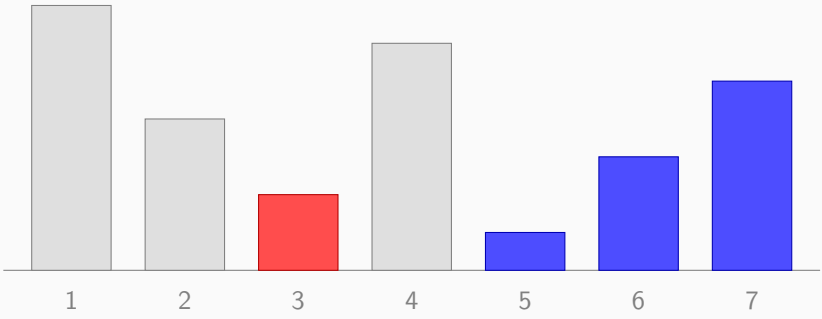


LIS: 3 3

1 2

H: Historical Hierarchy

Problem author: Reinier Schmiermann

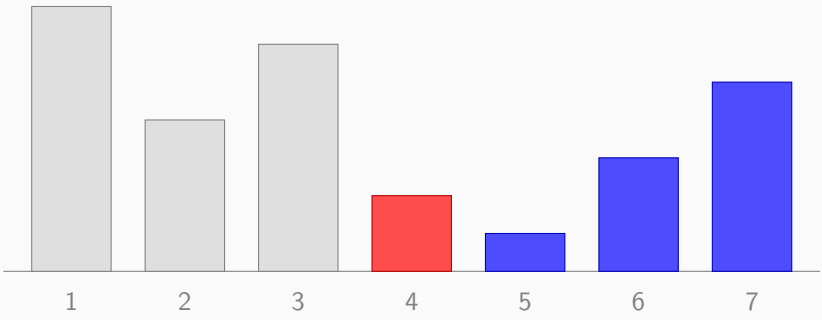


LIS: 3 3 3

2 3

H: Historical Hierarchy

Problem author: Reinier Schmiermann

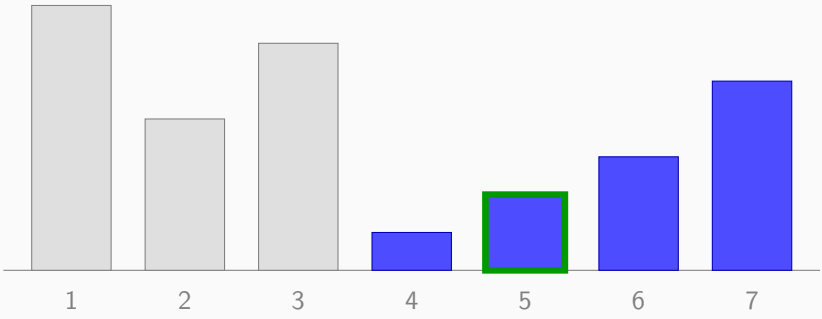


LIS: 3 3 3 3

3 4

H: Historical Hierarchy

Problem author: Reinier Schmiermann

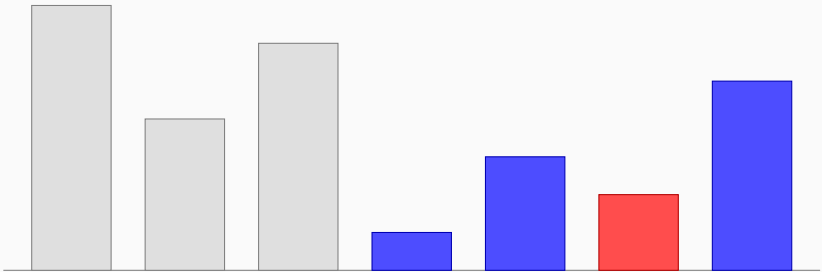


LIS: 3 3 3 3 **4**

4 5

H: Historical Hierarchy

Problem author: Reinier Schmiermann

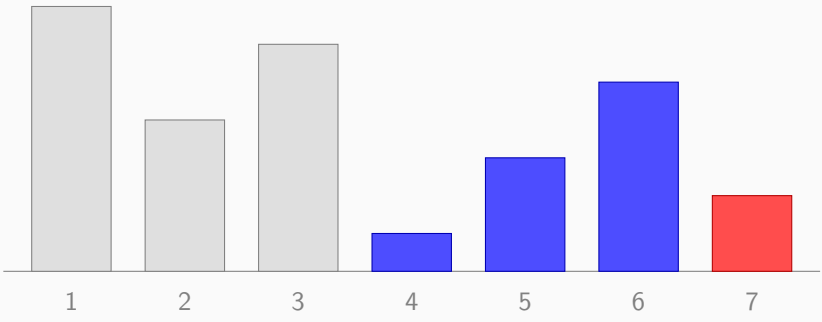


LIS: 3 3 3 3 4 3

5 6

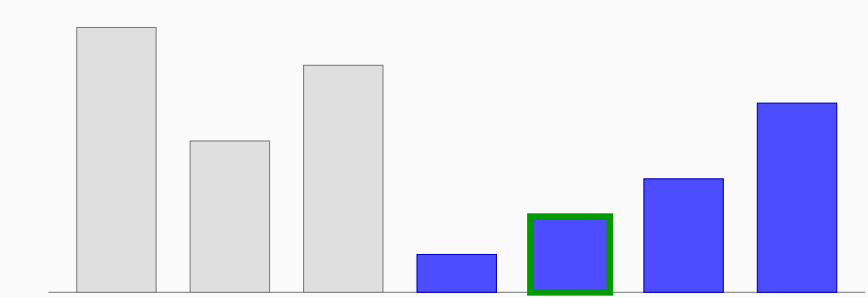
H: Historical Hierarchy

Problem author: Reinier Schmiermann



H: Historical Hierarchy

Problem author: Reinier Schmiermann



7 5

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere. Choose to place $a_i = x$ at position of max LIS.

But why does this algorithm sort the permutation in the end?

Issue: Multiple increasing subsequences that are tied for longest make things confusing.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere. Choose to place $a_i = x$ at position of max LIS.

But why does this algorithm sort the permutation in the end?

Issue: Multiple increasing subsequences that are tied for longest make things confusing.

Proof: Prove that after maximizing the LIS for the element $x = a_i$ the elements that we labeled $1, 2, \ldots, x$ are all part of all LISes $(*)$.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere. Choose to place $a_i = x$ at position of max LIS.

But why does this algorithm sort the permutation in the end?

Issue: Multiple increasing subsequences that are tied for longest make things confusing.

Proof: Prove that after maximizing the LIS for the element $x = a_i$ the elements that we labeled $1, 2, \ldots, x$ are all part of all LISes $(*)$.

Induction Step: Assume $(*)$ is true for $x - 1$.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere. Choose to place $a_i = x$ at position of max LIS.

But why does this algorithm sort the permutation in the end?

Issue: Multiple increasing subsequences that are tied for longest make things confusing.

Proof: Prove that after maximizing the LIS for the element $x = a_i$ the elements that we labeled $1, 2, \ldots, x$ are all part of all LISes $(*)$.

Induction Step: Assume $(*)$ is true for $x - 1$.

Option 1: x is already part of all LISes, then wherever it moves, the LIS can only stay equal or go down. In this case we should keep x on its original position, before the operations. Then $(*)$ holds for this x as well.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Algorithm: For all x from 1 to n, find the element with $a_i = x$, and try deleting it and placing it everywhere. Choose to place $a_i = x$ at position of max LIS.

But why does this algorithm sort the permutation in the end?

Issue: Multiple increasing subsequences that are tied for longest make things confusing.

Proof: Prove that after maximizing the LIS for the element $x = a_i$ the elements that we labeled $1, 2, \ldots, x$ are all part of all LISes $(*)$.

Induction Step: Assume $(*)$ is true for $x - 1$.

Option 1: x is already part of all LISes, then wherever it moves, the LIS can only stay equal or go down. In this case we should keep x on its original position, before the operations. Then $(*)$ holds for this x as well.

Option 2: x is not part of some LIS. Then in some operation it gets inserted at the right spot inside the gaps of this LIS, So there is a position which makes the length of LIS increment. There could be multiple previous LIS which now increased in length by 1. But by assumption, elements 1 to $x - 1$ are in all LISes, so also the ones that grew to length+1, so $(*)$ still holds after this step.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Fun Fact: It is possible to solve the problem in $n \log_2 n + \mathcal{O}(n)$ queries with high probability.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Fun Fact: It is possible to solve the problem in $n \log_2 n + \mathcal{O}(n)$ queries with high probability. Ask Mike for details.

H: Historical Hierarchy

Problem author: Reinier Schmiermann



Fun Fact: It is possible to solve the problem in $n \log_2 n + \mathcal{O}(n)$ queries with high probability. Ask Mike for details.

Statistics: 48 submissions, 9 accepted, 28 unknown