J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

First thought: Toposort on $a \ll b$.

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

First thought: Toposort on $a \ll b$.

Result: This fails if $a \approx b$, $b \approx c$, $a \ll c$ (sample 3).

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

First thought: Toposort on $a \ll b$.

Result: This fails if $a \approx b$, $b \approx c$, $a \ll c$ (sample 3).

Observation: If $a \approx b$, the list can only be sorted if there exists such a c to differentiate between them.

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

First thought: Toposort on $a \ll b$.

Result: This fails if $a \approx b$, $b \approx c$, $a \ll c$ (sample 3).

Observation: If $a \approx b$, the list can only be sorted if there exists such a c to differentiate between them.

Solution: For each pair $a \approx b$, look for a c that determines their order. Then toposort.

J: Juxtaposort

Problem author: Lammert Westerdijk



Problem: Sort a list of values, if unambiguously possible, using the results of all $a_i - 1 < a_j$ comparisons.

Observation: Comparing numbers a and b both ways results in the cases $a \leq b - 1$ ($a \ll b$), $b - 1 < a < b + 1$ ($a \approx b$), $b + 1 \leq a_i$ ($a \gg b$)

First thought: Toposort on $a \ll b$.

Result: This fails if $a \approx b$, $b \approx c$, $a \ll c$ (sample 3).

Observation: If $a \approx b$, the list can only be sorted if there exists such a c to differentiate between them.

Solution: For each pair $a \approx b$, look for a c that determines their order. Then toposort.

Result: $\mathcal{O}(n^3)$ is probably too slow (note: $\mathcal{O}(n^3/\mathrm{bitset})$ is doable).

J: Juxtaposort

Problem author: Lammert Westerdijk



Observation: Comparing with one number divides the other numbers into 3 sets.

J: Juxtaposort

Problem author: Lammert Westerdijk



Observation: Comparing with one number divides the other numbers into 3 sets.

Solution: Keep a sorted list of sets of unsorted values, and compare with each number to partition these sets.

J: Juxtaposort

Problem author: Lammert Westerdijk



Observation: Comparing with one number divides the other numbers into 3 sets.

Solution: Keep a sorted list of sets of unsorted values, and compare with each number to partition these sets.

Result: This works.

J: Juxtaposort

Problem author: Lammert Westerdijk



Observation: Comparing with one number divides the other numbers into 3 sets.

Solution: Keep a sorted list of sets of unsorted values, and compare with each number to partition these sets.

Result: This works.

Running time: $\mathcal{O}(n^2)$.

J: Juxtaposort

Problem author: Lammert Westerdijk



Observation: Comparing with one number divides the other numbers into 3 sets.

Solution: Keep a sorted list of sets of unsorted values, and compare with each number to partition these sets.

Result: This works.

Running time: $\mathcal{O}(n^2)$.

...but it is needlessly complicated!

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

Implementation: For each $a_i - 1 < a_j$, do `s[i]--` and `s[j]++`.

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

Implementation: For each $a_i - 1 < a_j$, do `s[i]--` and `s[j]++`.

Result: This works.

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

Implementation: For each $a_i - 1 < a_j$, do `s[i]--` and `s[j]++`.

Result: This works.

Why: If $a_i < a_j$, s_j will get at least as many increments as s_i, and s_i will get at least as many decrements as s_j.

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

Implementation: For each $a_i - 1 < a_j$, do `s[i]--` and `s[j]++`.

Result: This works.

Why: If $a_i < a_j$, s_j will get at least as many increments as s_i, and s_i will get at least as many decrements as s_j.

Running time: $\mathcal{O}(n^2)$.

J: Juxtaposort

Problem author: Lammert Westerdijk



Simple solution: Give each value a_i a score s_i, then sort and check for ambiguity.

Implementation: For each $a_i - 1 < a_j$, do `s[i]--` and `s[j]++`.

Result: This works.

Why: If $a_i < a_j$, s_j will get at least as many increments as s_i, and s_i will get at least as many decrements as s_j.

Running time: $\mathcal{O}(n^2)$.

Statistics: 31 submissions, 5 accepted, 18 unknown