Identifying and sorting Jordan sequences

Example:

2, 5, 8, 7, 6, 9, 4, 3, 10, 1, 11, 12

Jordan seq: result of this process:

Problem: Given sequence, verify if Jordan-seq, and sort.

Want to do it in \(O(n)\) time.

Claim: \# permutations of size \(m\) that are Jordan-seq, \(m \leq 5^n\).

(No inclusion lower bound does not apply, same argument would yield at best \(\Omega(n)\).

Proof of Claim:

Applications

Find intersections between polygon and line

(fast, use in order along polygon)

Want to report along line.

\(\Rightarrow\) Jordan sorting
Application (Computer Graphics)
- Find visible part of polygon inside window
- Find intersections along polygon
- Cut them along window
  \( \rightarrow \) Forda sorting

Algorithm

\[ x_0, x_1, \ldots \]

\[ 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 \]

Idea: Represent Forda sq. with two trees, one above one below

Obv. pair \( \{ x_{2i-1}, x_{2i} \} \) is a 'bump'

Obv. bumps cannot intersect

2 valid cases:
- OK.
- NO.

Upper tree
- Build tree with nodes \( \{ x_{2i-1}, x_{2i} \} \)
  for \( i = 1, 2, \ldots \)

  - Parent of \( \{ x_{2i-1}, x_{2i} \} \) is closest pair that contains \( x_0 \)
    (or \( \{ -\infty, \infty \} \))

Lower tree
- Nodes \( \{ x_{2i}, x_{2i+1} \} \)
  for \( i = 1, 2, \ldots \)
Algorithm

Input: Sequence $X_1, \ldots, X_n$ (list of intersections along curve)

Process $X_1, \ldots, X_n$

Build:
- upper tree
- lower tree
- sequence sorted along line

if get stuck, report
    "NOT Jordan-seq."

initialize:
- make $X_1$ root of both trees
- sorted list $(-\infty, X_1, +\infty)$
- for $i=2, \ldots, n$ process $X_i$

if $i$ even, then add $\{X_{i-1}, X_i\}$ to upper tree
(if $i$ odd, then add $\{X_{i-1}, X_i\}$ to lower tree) — symmetric, details skipped
(assume $X_{i-1} < X_i$, otherwise symmetric)

0() {
  \[ \begin{align*}
  &\text{(1) find } X \text{ in sorted list that is right neighbor of } X_{i-1} \\
  &\text{(2) find pair in upper tree containing } X, \quad X \in \{L, R\}
  \end{align*} \]
}

5 cases

A

\[ \begin{align*}
  &L < X_{i-1} < R < X_i \\
  &\text{Not A Jordan-seq. STOP}
  \end{align*} \]

B

\[ \begin{align*}
  &L < X_{i-1} < X_i < R \\
  &\text{make } \{X_{i-1}, X_i\} \text{ rightmost child of } \{L, R\} \\
  &\text{insert } X_i \text{ after } X_{i-1} \text{ a sorted list}
  \end{align*} \]
Data Structure

- Sorted list as doubly linked list (must allow query element in O(1) time)
- Lists of siblings (in upper and lower trees) as finger search trees

\[ x_{i-1} < x_i < L < R \]
- Make \( \{x_{i-1}, x_i\} \) new left sibling of \( \{L, R\} \)
- Insert \( x_i \) after \( x_{i-1} \) in sorted list

\[ x_{i-1} < L < x_i < R \]
- Not a Jordan seq. Stop

\[ x_{i-1} < L < R < x_i \]
- Find rightmost sibling of \( \{L, R\} \)
- i.e. \( L < x_i \)
  (If \( R > x_i \) then NOT Jordan seq. Stop)
- Find parent of \( \{L, R\} \), call it \( \{L', R'\} \)
  (If \( R' < x_i \) then NOT Jordan seq. Stop)
- Remove subtree of children \( \{L, R\}, \ldots, \{L', R'\} \) of \( \{L, R\} \)
  replace by new node \( \{x_{i-1}, x_i\} \)
  make removed subtree child of \( \{x_{i-1}, x_i\} \)
- Insert \( x_i \) after \( R' \) in sorted list
Running time

- last rightmost child $O(n)$
- last left neighbor $O(n)$

- $2$-pivot split, remove sublist of length $d$ from a list of length $t$ 
  (after searching for endpoint) 
  $O(lg \ min \{d, t-d\})$

  (recall line split of $F_{st}$)

Overall running time

Total cost of splitgings (renew analysis of Application-2 splittable lists)

- split $(ij)$ cost $lg \ min \{d, t-d\}$
- insert cost $O(1)$

Define potential $t - lg_{2}t$ for list of length $t$

Total potential $\phi = \sum_{w \in W} \phi(w)$

Amortized cost of split $O(1)$

Amortized cost of insert $O(1) + \Delta \phi = O(1)$

Proceed $X_i$ amortized $O(1)$

Total time for finder-sort $O(n)$. 