LONGEST INCREASING SUBSEQUENCE
**Longest increasing subsequence**

**Longest increasing subsequence.** Given a sequence of elements $c_1, c_2, \ldots, c_n$ from a totally-ordered universe, find the longest increasing subsequence.

**Ex.** 7 2 8 1 3 4 10 6 9 5.

**Application.** Part of MUMmer system for aligning entire genomes.

**$O(n^2)$ dynamic programming solution.** LIS is a special case of edit-distance.

- $x = c_1 c_2 \cdots c_n$.
- $y =$ sorted sequence of $c_k$, removing any duplicates.
- Mismatch penalty $= \infty$; gap penalty $= 1$. 
Patience solitaire

**Patience.** Deal cards $c_1, c_2, \ldots, c_n$ into piles according to two rules:
- Can't place a higher-valued card onto a lowered-valued card.
- Can form a new pile and put a card onto it.

**Goal.** Form as few piles as possible.
Patience: greedy algorithm

**Greedy algorithm.** Place each card on leftmost pile that fits.
Patience: greedy algorithm

**Greedy algorithm.** Place each card on leftmost pile that fits.

**Observation.** At any stage during greedy algorithm, top cards of piles increase from left to right.
Weak duality. In any legal game of patience, the number of piles ≥ length of any increasing subsequence.

Pf.
- Cards within a pile form a decreasing subsequence.
- Any increasing sequence can use at most one card from each pile.
Patience-LIS: strong duality

**Theorem.** [Hammersley 1972] Min number of piles = max length of an IS; moreover greedy algorithm finds both.

**Pf.** Each card maintains a pointer to top card in previous pile.
- Follow pointers to obtain IS whose length equals the number of piles.
- By weak duality, both are optimal. □
**Greedy algorithm: implementation**

**Theorem.** The greedy algorithm can be implemented in $O(n \log n)$ time.
- Use $n$ stacks to represent $n$ piles.
- Use binary search to find leftmost legal pile.

```plaintext
**PATIENCE** $(n, c_1, c_2, ..., c_n)$

**INITIALIZE** an array of $n$ empty stacks $S_1, S_2, ..., S_n$.

**FOR** $i = 1$ **TO** $n$

- $S_j ←$ binary search to find leftmost stack that fits $c_i$.
- **PUSH** ($S_j, c_i$).
- $pred[c_i] ←$ **PEEK** ($S_{j-1}$). ← null if $j = 1$

**RETURN** sequence formed by following pointers from top card of rightmost nonempty stack.
```
**Patience sorting**

**Patience sorting.** Deal all cards using greedy algorithm; repeatedly remove smallest card.

**Theorem.** For uniformly random deck, the expected number of piles is approximately $2n^{1/2}$ and the standard deviation is approximately $n^{1/6}$.

**Remark.** An almost-trivial $O(n^{3/2})$ sorting algorithm.

**Speculation.** [Persi Diaconis] Patience sorting is the fastest way to sort a pile of cards by hand.
**Bonus theorem**

**Theorem.** [Erdös-Szekeres 1935] Any sequence of $n^2 + 1$ distinct real numbers either has an increasing or decreasing subsequence of size $n + 1$.

**Pf.** [by pigeonhole principle]