**Longest Increasing Subsequence**

**Longest increasing sequence.** 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 whole genomes.

---

**Patience solitaire**

**Rules.** Deal cards \( c_1, c_2, \ldots, c_n \) into piles according to two rules:
- Can put next card into a new singleton pile.
- Can put next card on a pile if it's smaller than the top card of pile.

**Goal.** Form as few piles as possible.

---

**Patience-LIS: weak duality**

**Weak duality.** Length of any increasing subsequence \( \leq \) number of piles.

**Pf.**
- Cards within a pile form a decreasing subsequence.
- Any increasing sequence can use at most one card per pile.
**Patience-LIS: weak duality**

**Weak duality.** Length of any increasing subsequence ≤ number of piles.

**Corollary.** If length of an increasing subsequence = number of piles, then both are optimal.

**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.
- Following pointers yields an increasing subsequence.
- Length of this increasing subsequence = number of piles.
- By weak duality corollary, both are optimal.

**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.
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.

```
PATIENTE-SORT($n$, $c_1$, $c_2$, ..., $c_n$)
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 predecessor pointers from top card of rightmost nonempty stack.
```

Patience sorting

**Patience sorting.** [Ross, Mallows 1962]
- Deal cards using greedy algorithm.
- Repeatedly remove the smallest card among the remaining piles.

**Theorem.** Can implement patience sorting in $O(n \log n)$ time.
- To represent piles: use an array of stacks.
- To deal cards: use binary search to find leftmost pile.
- To remove cards: maintain piles in a binary heap (priority = top card).

**Speculation.** [Persi Diaconis] Is patience sorting the fastest way to sort a deck of cards by hand?

Bonus theorem

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

**Pf.** [by pigeonhole principle]
- Run greedy patience algorithm.
- Decreasing subsequence in each pile.
- Increasing subsequence using one card per pile.
- If $\leq n$ cards per pile and $\leq n$ piles, then $\leq n^2$ cards. ※