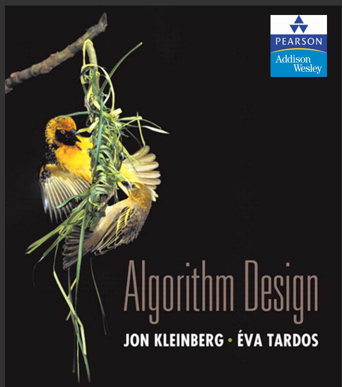


LONGEST INCREASING SUBSEQUENCE

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 4/11/18 8:47 AM

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

elements must be in order (but not necessarily contiguous)

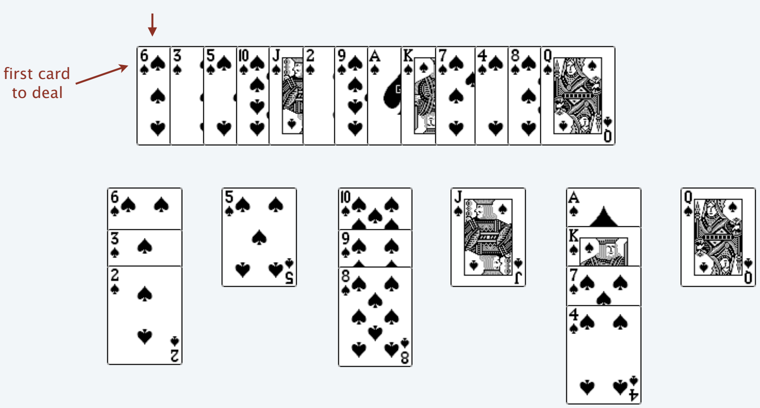
Application. Part of MUMmer system for aligning whole genomes.

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.

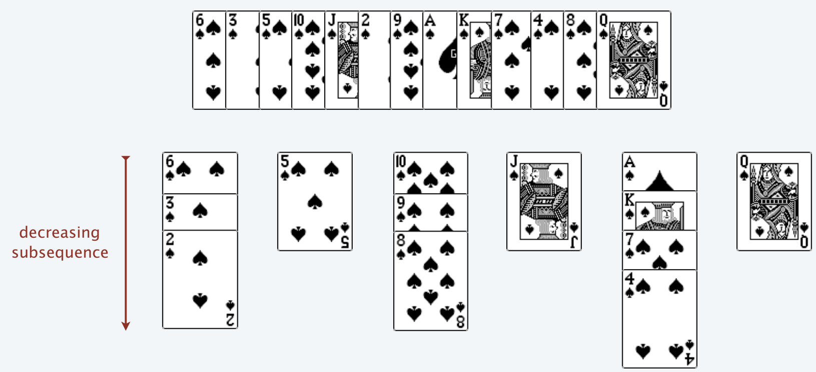


Patience-LIS: weak duality

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

Pf.

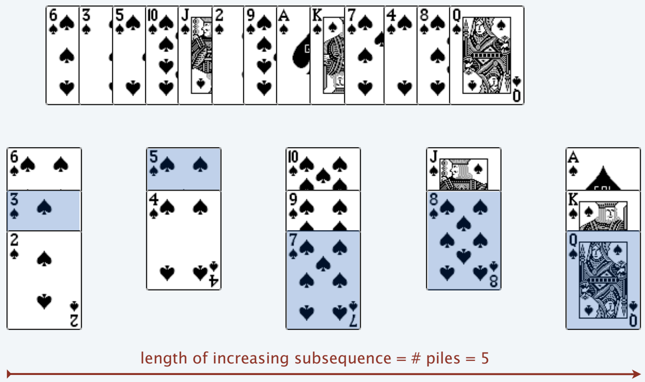
- Cards within a pile form a decreasing subsequence.
- Any increasing sequence can use at most one card per pile. ▪



Patience-LIS: weak duality

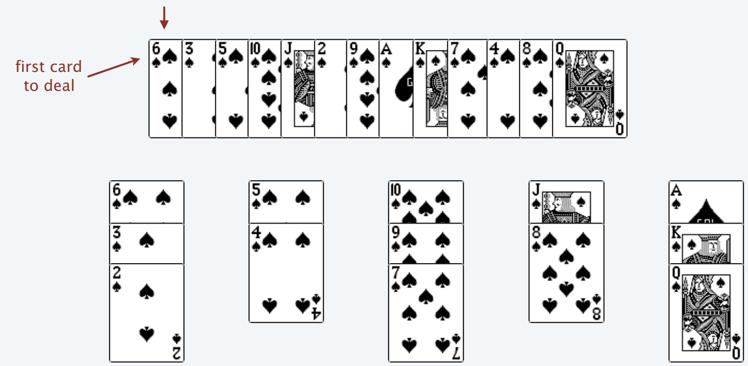
Weak duality. Length of any increasing subsequence ≤ number of piles.

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



Patience: greedy algorithm

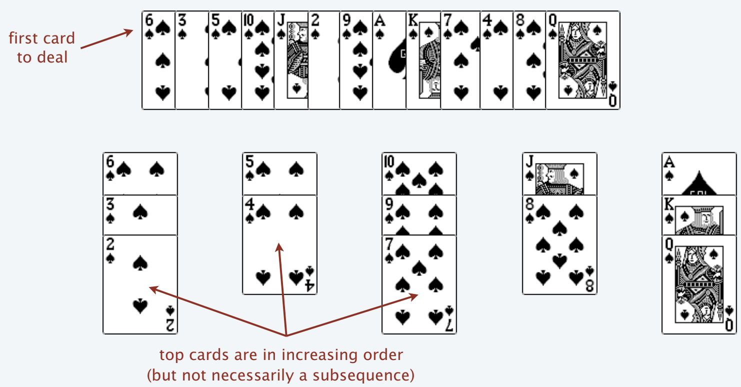
Greedy algorithm. Place each card on leftmost pile that fits.



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.



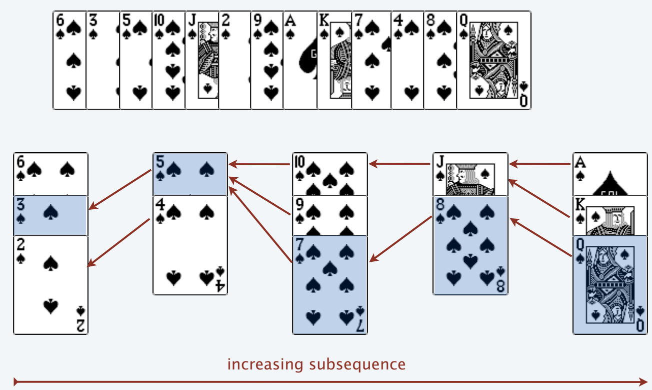
Patience-LIS: strong duality

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.

at time of insertion

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

```
PATIENCE-SORT(n, c1, c2, …, cn)

INITIALIZE an array of n empty stacks S1, S2, …, Sn.

FOR i = 1 TO n
   Sj ← binary search to find leftmost stack that fits ci.
   PUSH(Sj, ci).
   pred[ci] ← PEEK(Sj–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). ▪

shuffle deck before running algorithm

Theorem. The expected number of piles $\le 2\,n^{1/2}$.
Corollary. An elementary $O(n^{3/2})$ probabilistic sorting algorithm.

no need for even binary search

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 $\le n$ cards per pile and $\le n$ piles, then $\le n^2$ cards. ※