Functional Programming in Isabelle/HOL

Christian Sternagel

Computational Logic
Institute of Computer Science
University of Innsbruck

November 27, 2015

Course: Functional Programming 703024, WS 2015/16

Overview

- Summary of Last Week
- Some Background
- Examples

Summary of Last Week

Theory File: `Summary.thy`

- recursive functions
- proof by induction

Some Background

Theory File: `Insertion_Sort.thy`

- short history of Isabelle/HOL
- example – insertion sort

1969 **Dana Scott:**
Logic for Computable Functions

1969 1993

1969 **Dana Scott:**
Logic for Computable Functions

1969 — 1993

1972 **Robin Milner:**
Stanford LCF

1969 **Dana Scott:**
Logic for Computable Functions

1973 **ML:**
Meta Language

1969 1993

1972 **Robin Milner:**
Stanford LCF

1969 **Dana Scott:**
Logic for Computable Functions

1973 **ML:**
Meta Language

1969 — 1993

1972 **Robin Milner:**
Stanford LCF

1977 **Edinburgh LCF**

1969 **Dana Scott:**
Logic for Computable Functions

1973 **ML:**
Meta Language

1985 **Larry Paulson:**
Cambridge LCF

1969 — 1993

1972 **Robin Milner:**
Stanford LCF

1977 **Edinburgh LCF**

1969 **Dana Scott:**
Logic for Computable Functions

1973 **ML:**
Meta Language

1985 **Larry Paulson:**
Cambridge LCF

1969 — 1993

1972 **Robin Milner:**
Stanford LCF

1977 **Edinburgh LCF**

1986 **Isabelle**

1969 **Dana Scott:**
Logic for Computable Functions

1973 **ML:**
Meta Language

1985 **Larry Paulson:**
Cambridge LCF

Tobias Nipkow:
Isabelle/HOL 1993

1969 — 1993

1972 **Robin Milner:**
Stanford LCF

1977 **Edinburgh LCF**

1986 **Isabelle**

System Architecture

Standard ML implementation language

System Architecture

Isabelle/Pure generic proof assistant

Standard ML implementation language

System Architecture

Isabelle/HOL Higher-Order Logic

Isabelle/Pure generic proof assistant

Standard ML implementation language

System Architecture

Isabelle/jEdit jEdit based interface

Isabelle/Scala connects ML to JVM

Isabelle/HOL Higher-Order Logic

Isabelle/Pure generic proof assistant

Standard ML implementation language

A Small Trusted Kernel (aka the LCF-approach)

- abstract type `thm` for theorems
- small set of trusted functions that create theorems
- no other way to create theorems

A Small Trusted Kernel (aka the LCF-approach)

- abstract type `thm` for theorems
- small set of trusted functions that create theorems
- no other way to create theorems

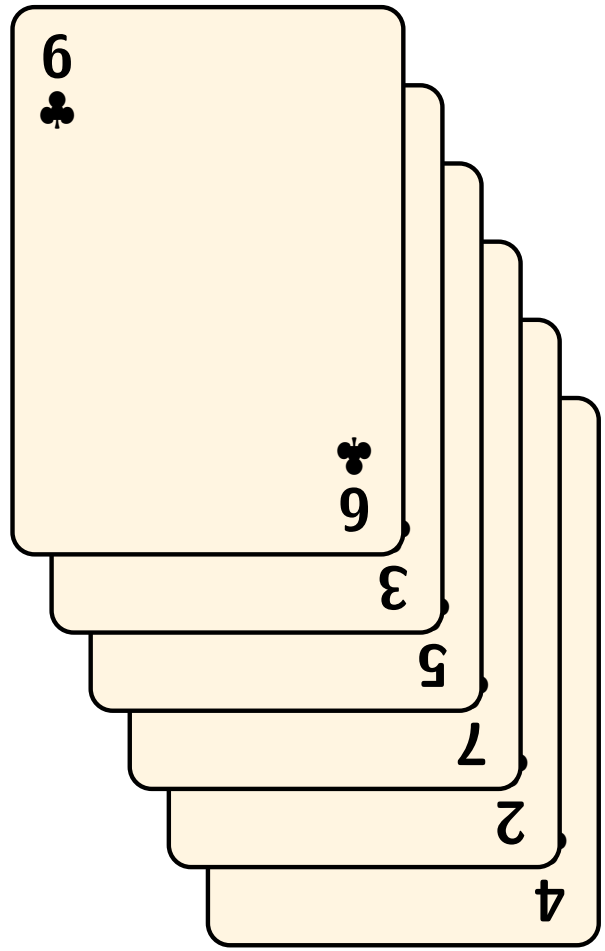
```
sig
  type thm
  val assume : form -> thm
  val implies_intr : form -> thm -> thm
  val implies_elim : thm -> thm -> thm
                     :
end
```

$$\frac{}{\phi \vdash \phi} \qquad \frac{\Gamma \vdash \psi}{\Gamma \backslash \{\phi\} \vdash \phi \rightarrow \psi} \qquad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \phi \rightarrow \psi}{\Gamma \vdash \psi}$$

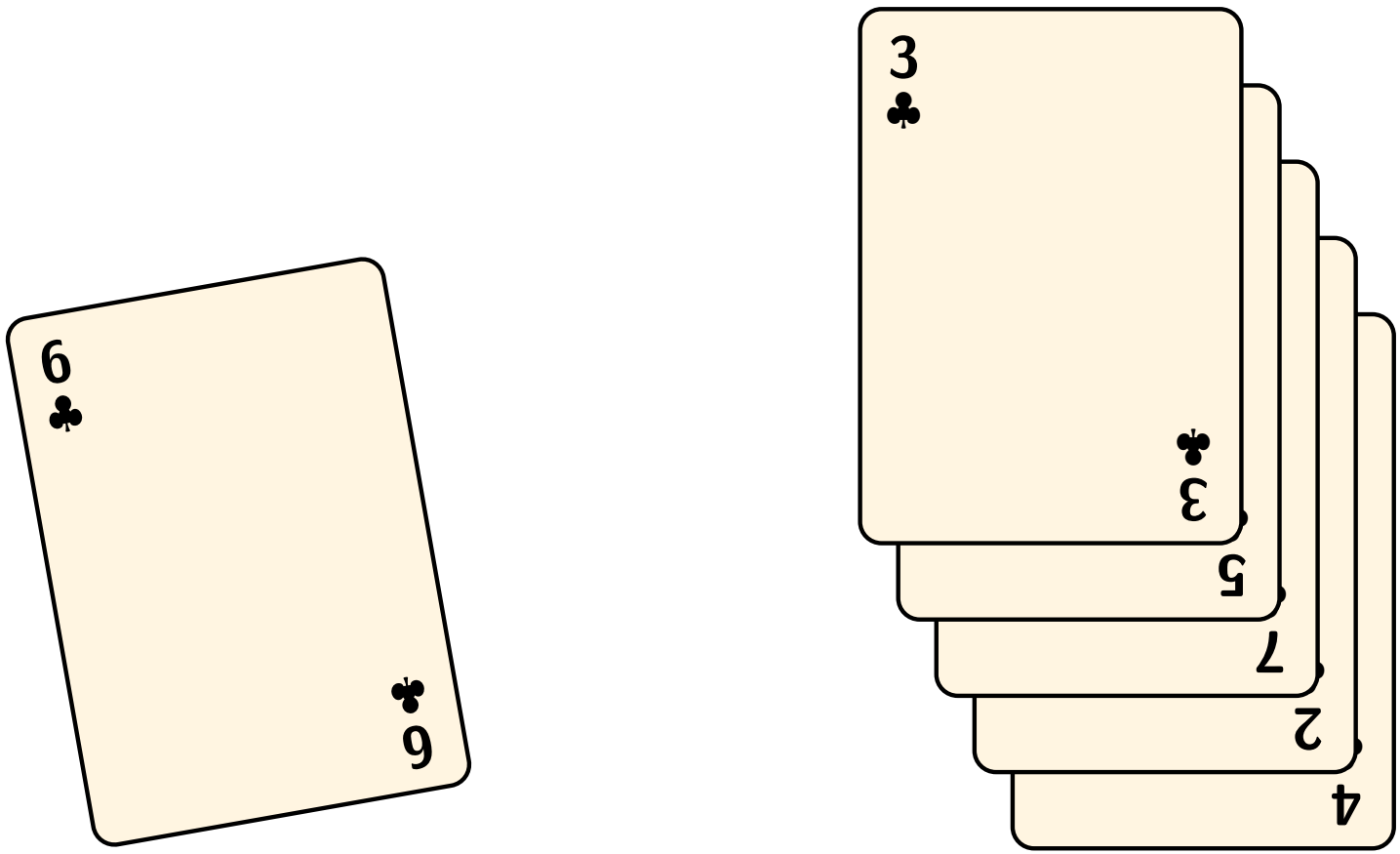
Higher-Order Logic

- HOL = Functional Programming + Logic
- data types (**`datatype`**)
- recursive functions (**`fun`**)
- logical operators ($\wedge$, $\vee$, $\longrightarrow$, $\forall$, $\exists$, ...)

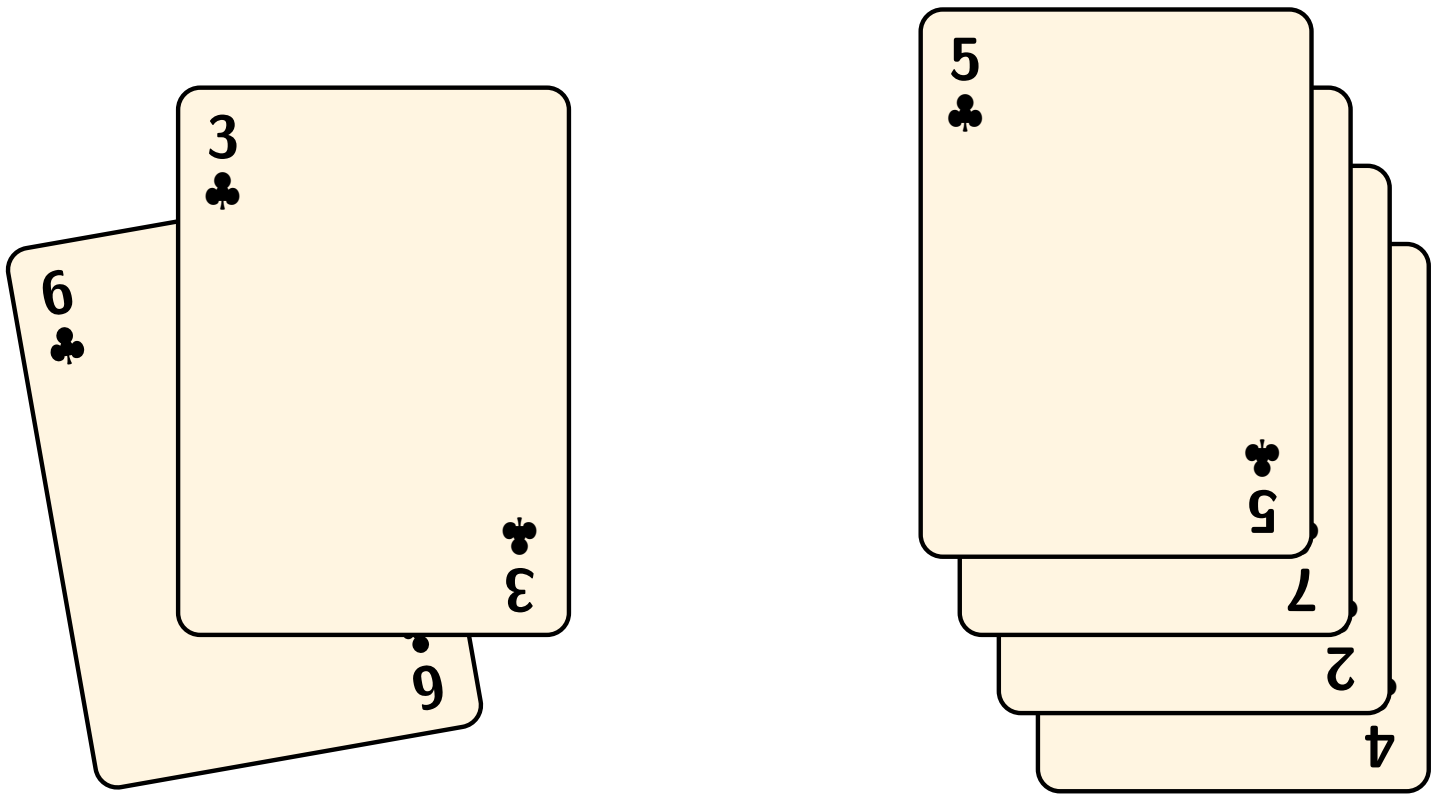
Example: Insertion Sort



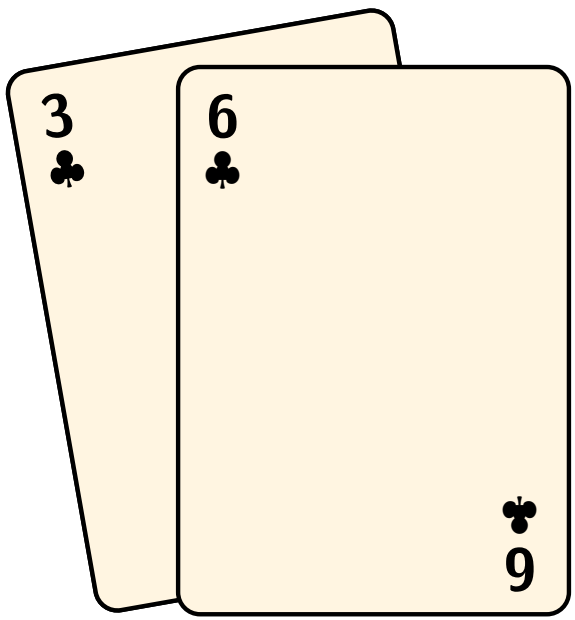
Example: Insertion Sort

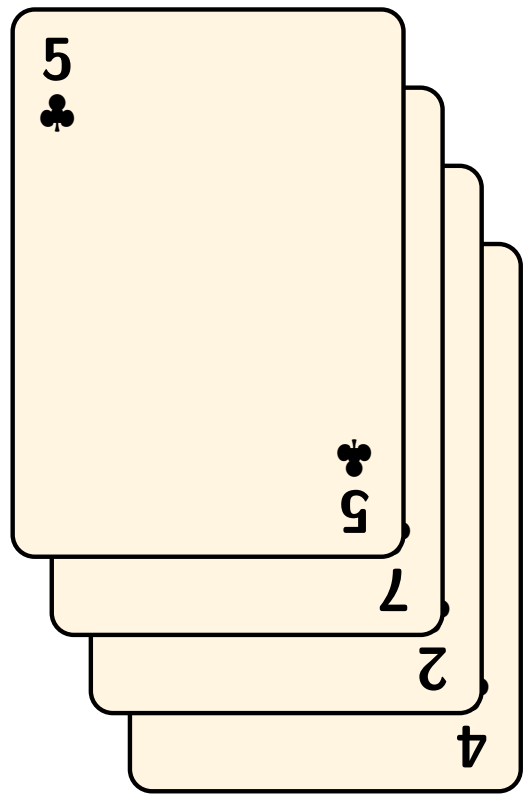


Example: Insertion Sort

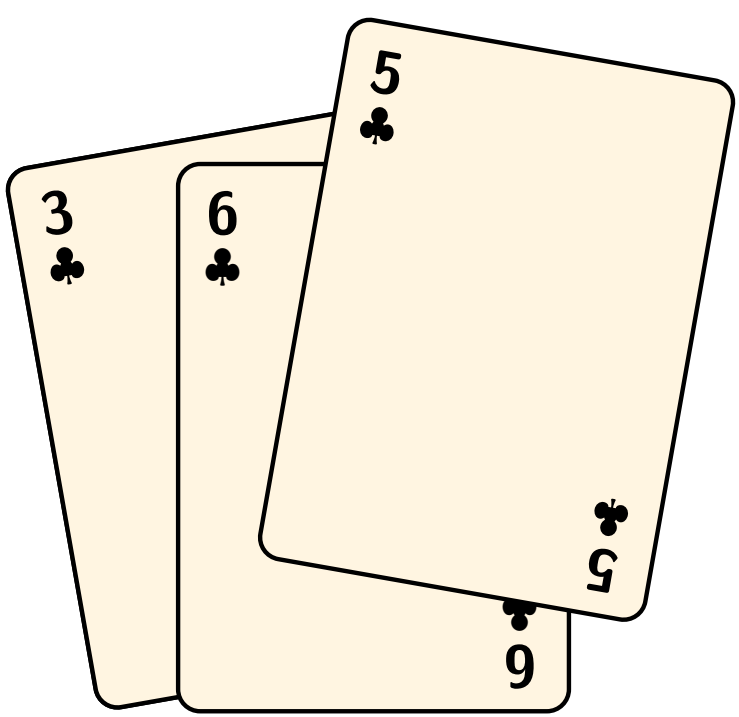


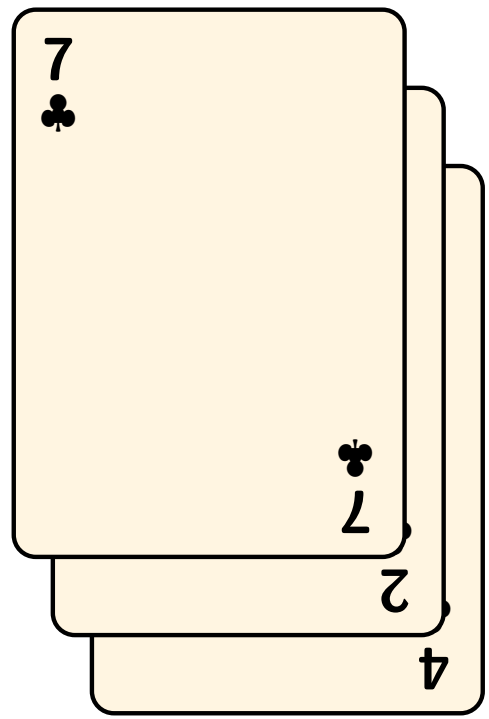
Example: Insertion Sort



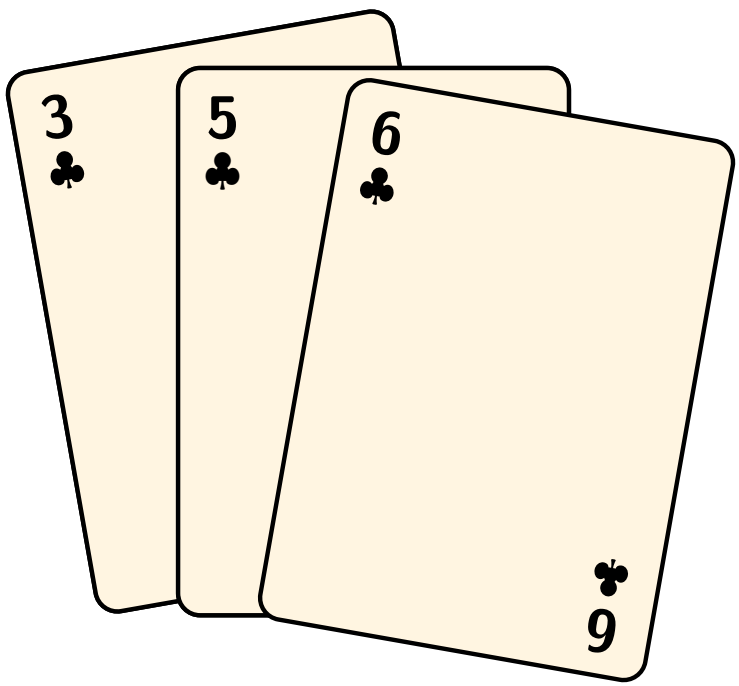


Example: Insertion Sort

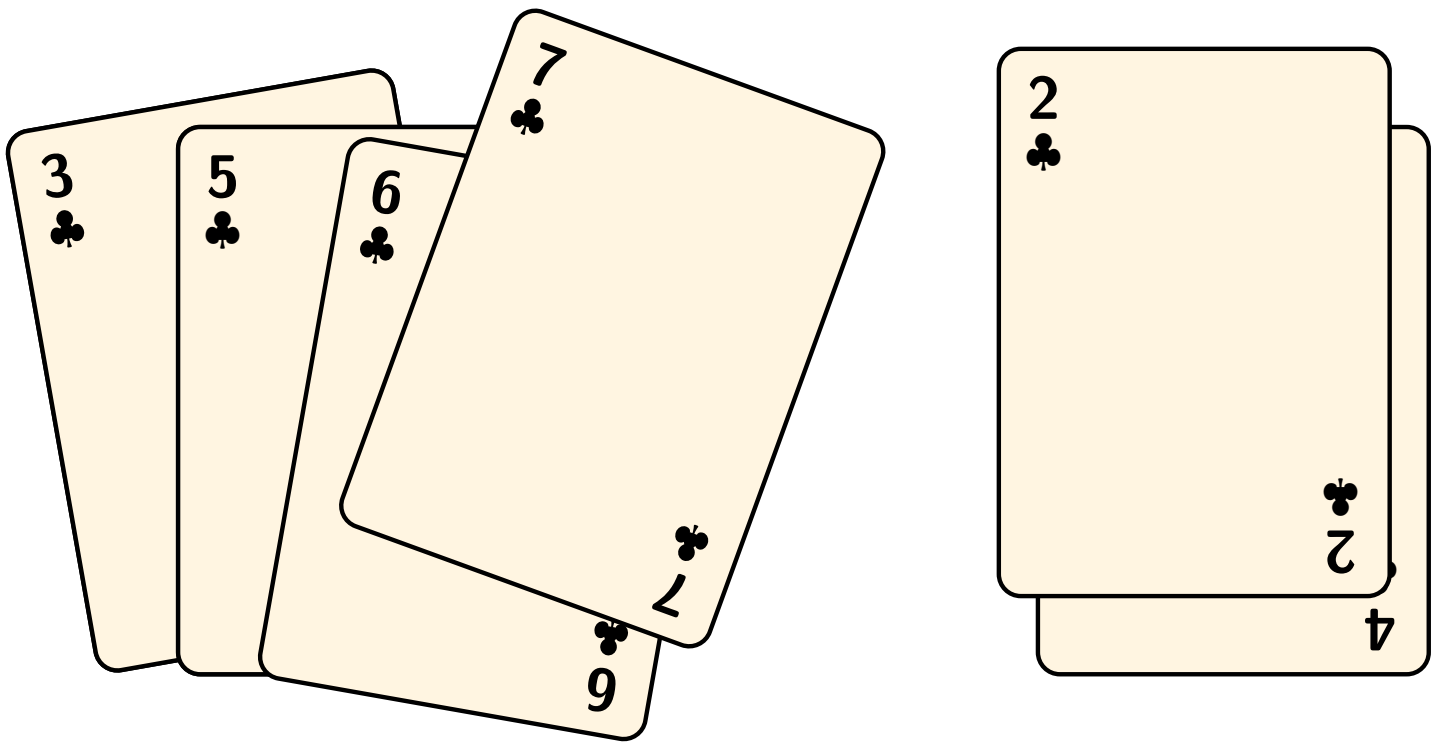




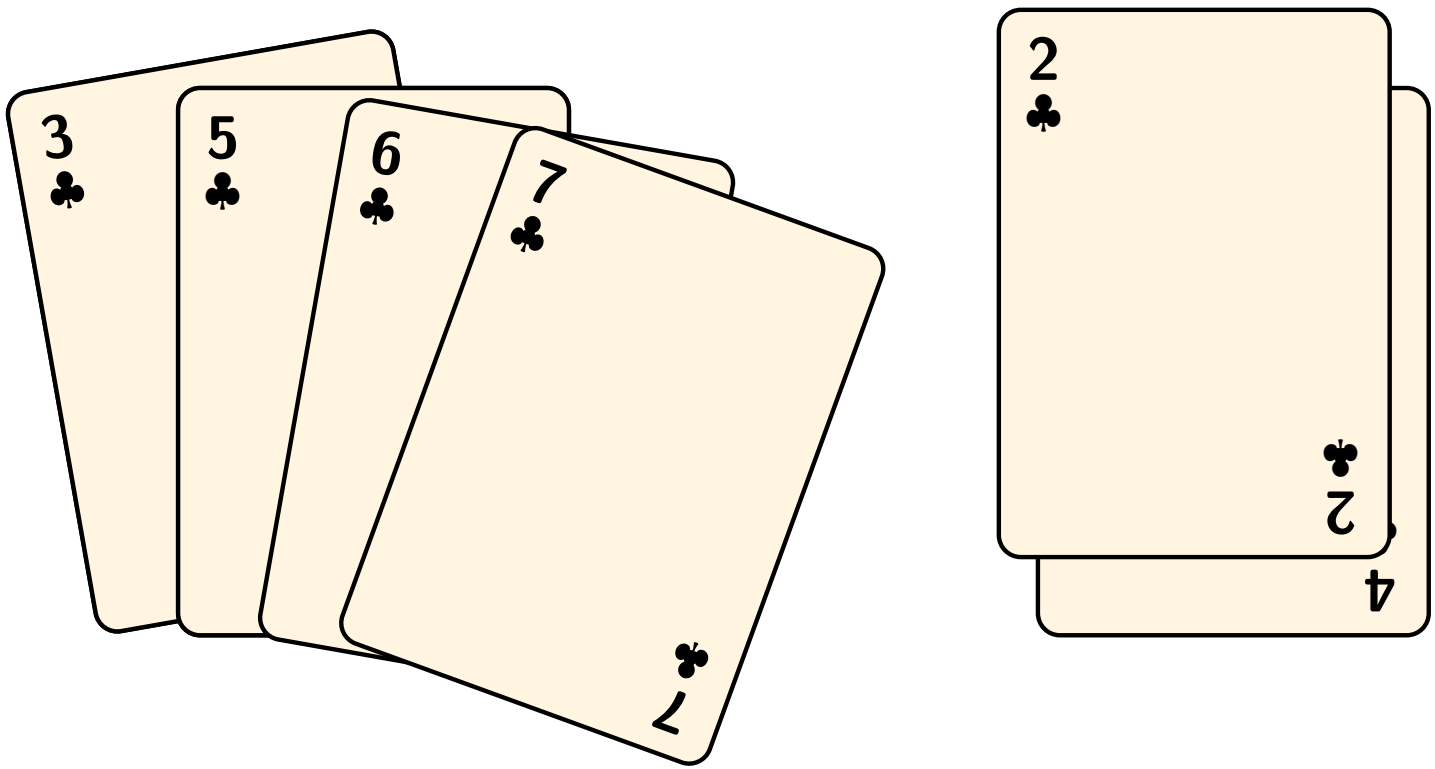
Example: Insertion Sort



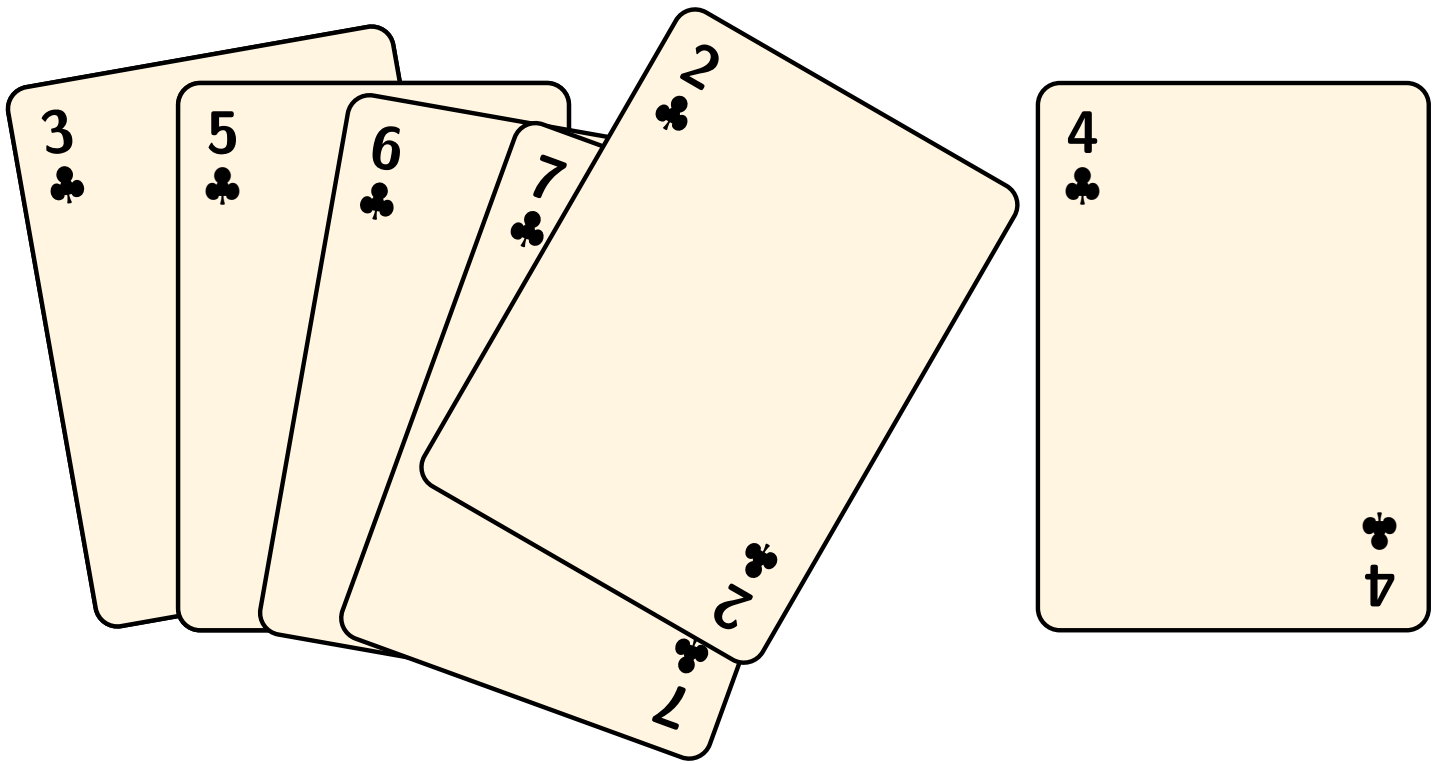
Example: Insertion Sort



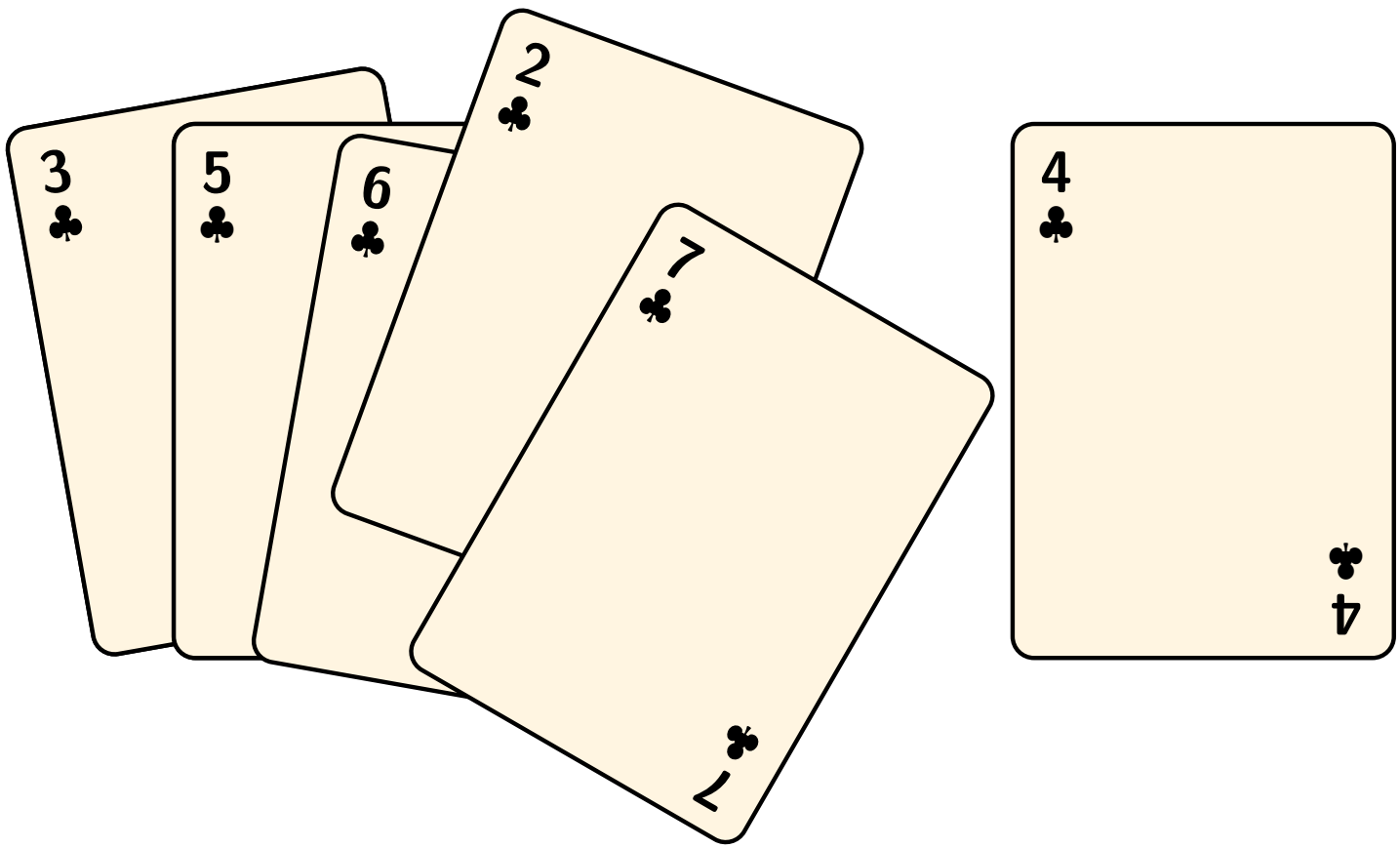
Example: Insertion Sort



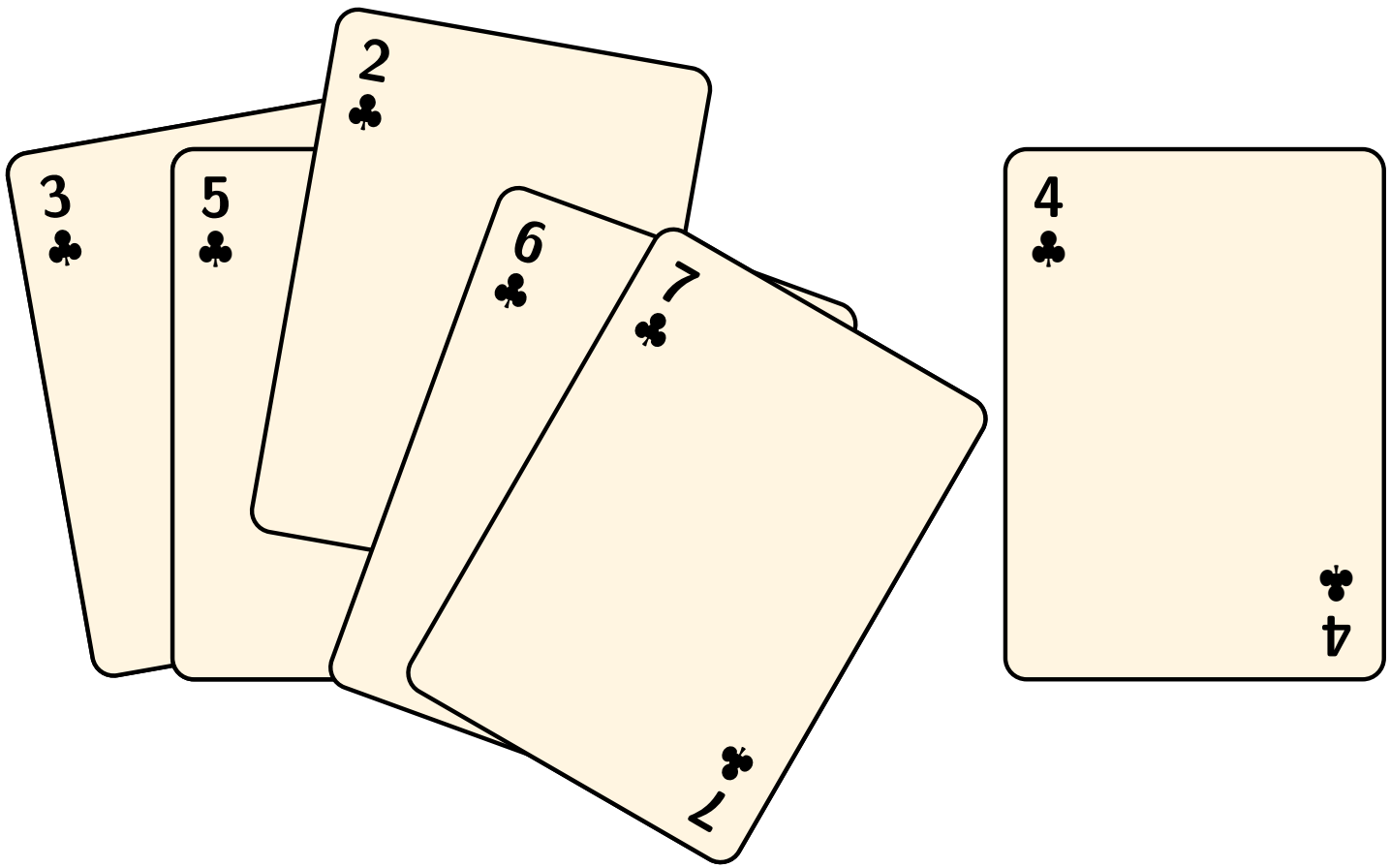
Example: Insertion Sort



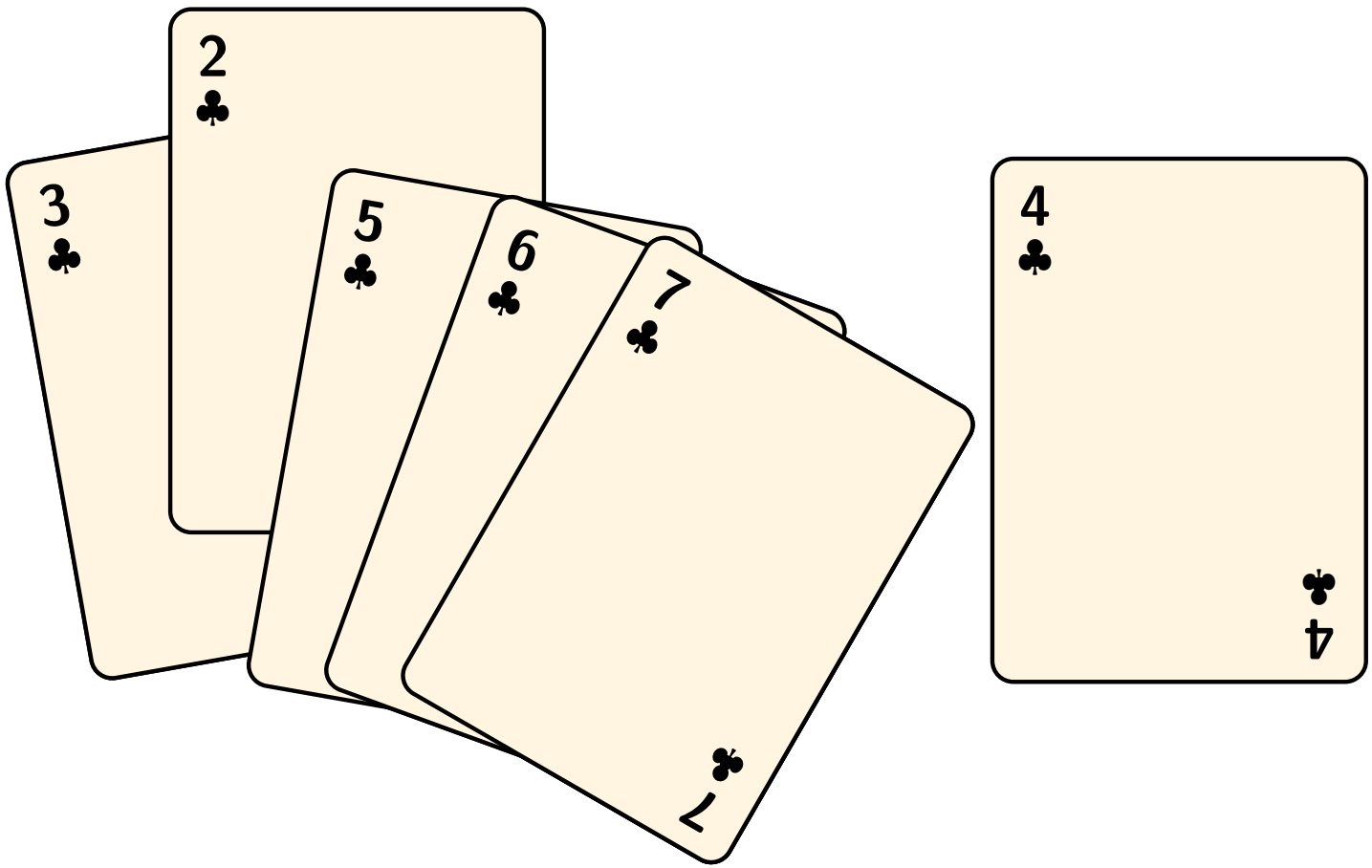
Example: Insertion Sort



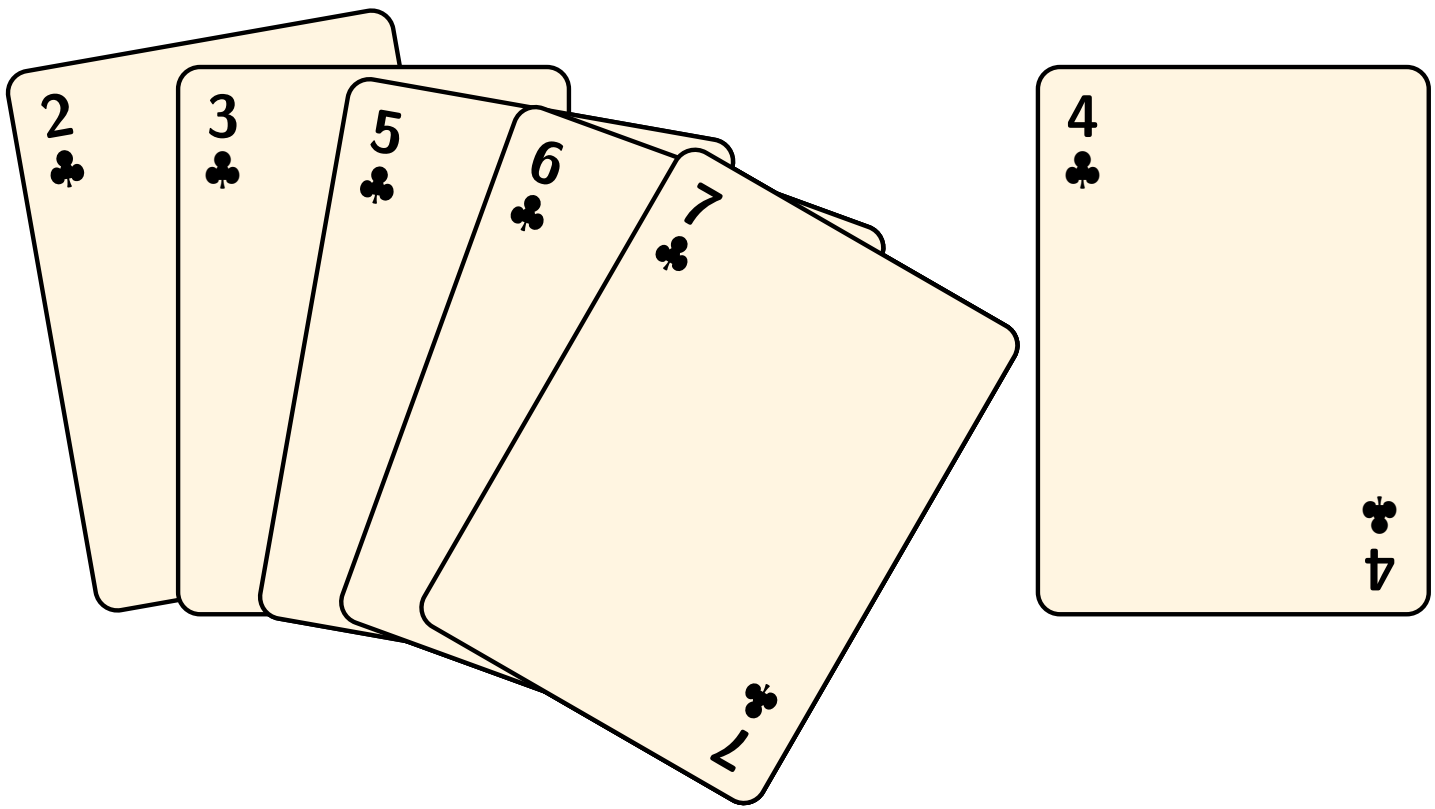
Example: Insertion Sort



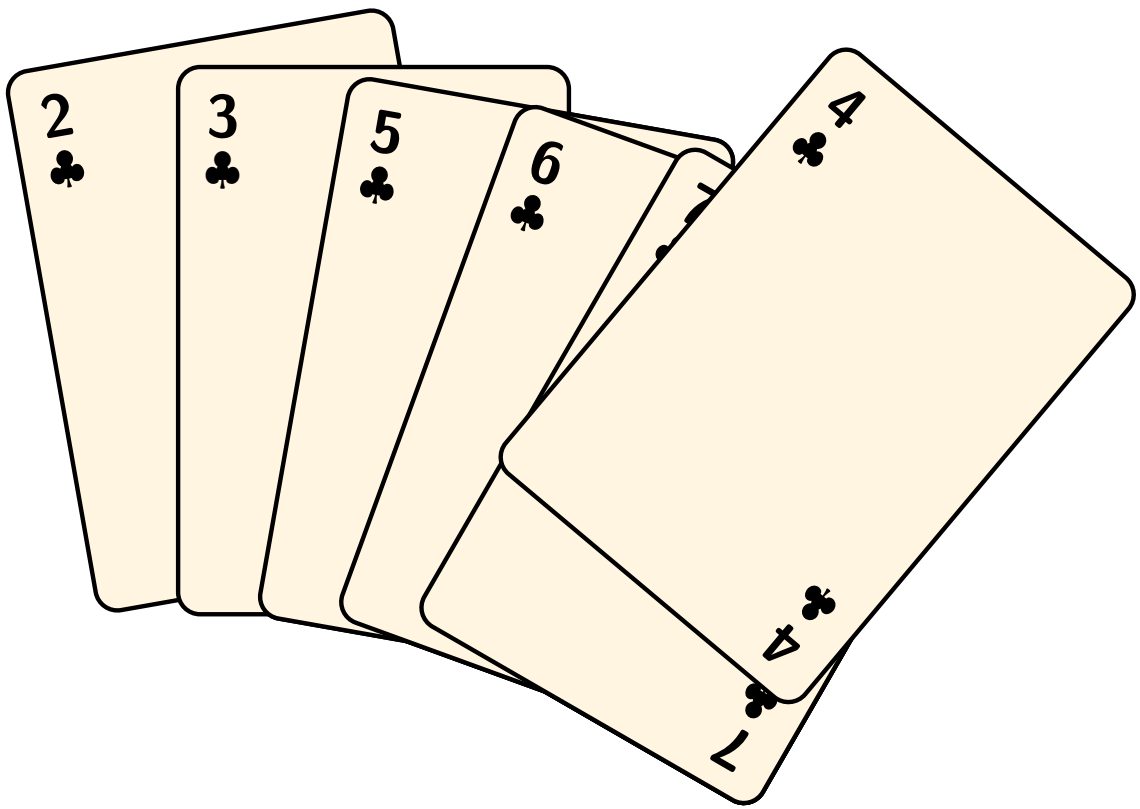
Example: Insertion Sort



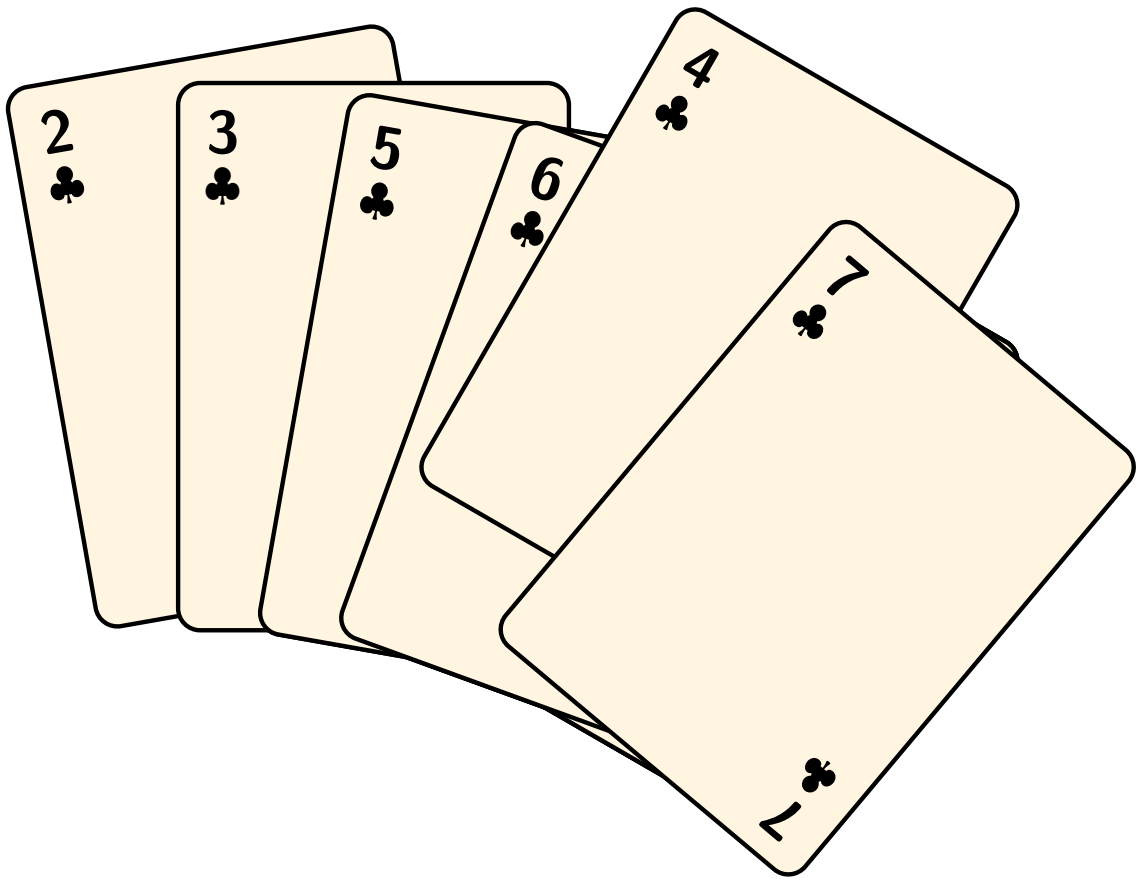
Example: Insertion Sort



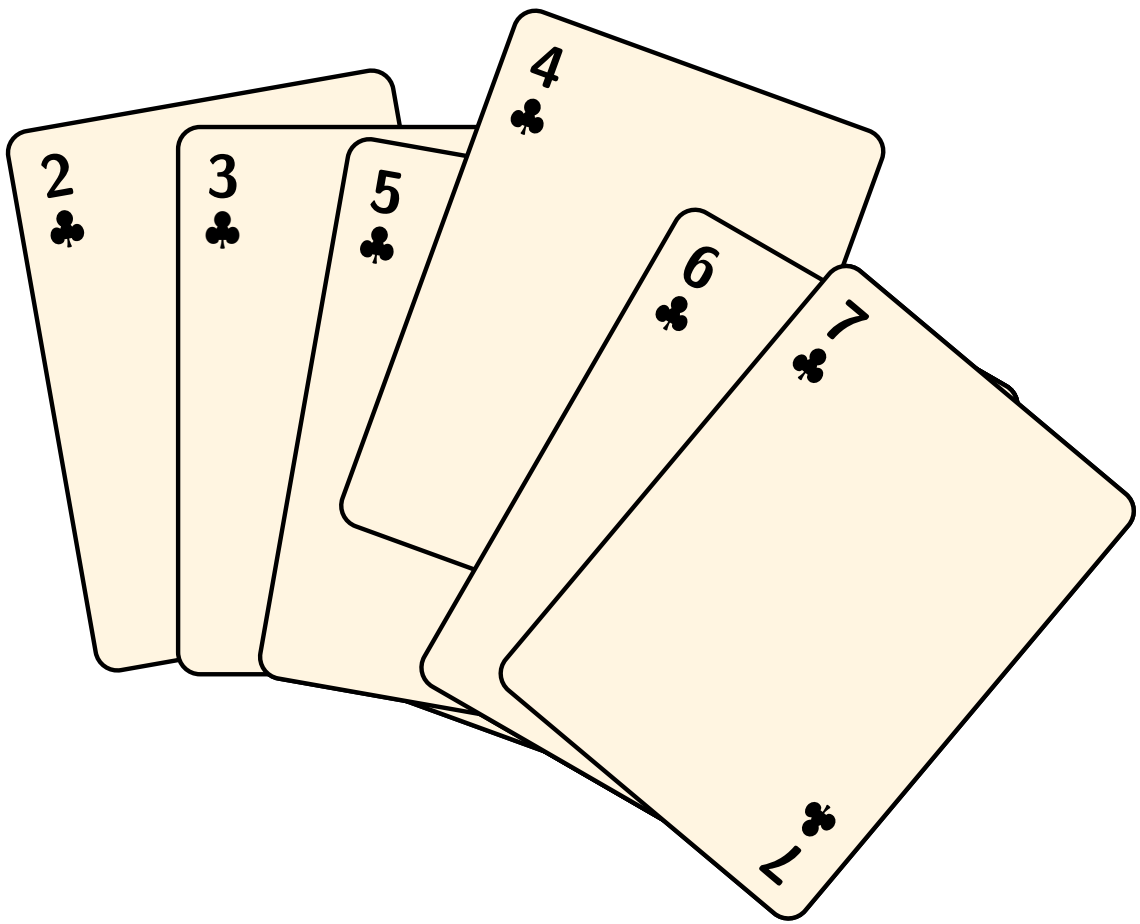
Example: Insertion Sort



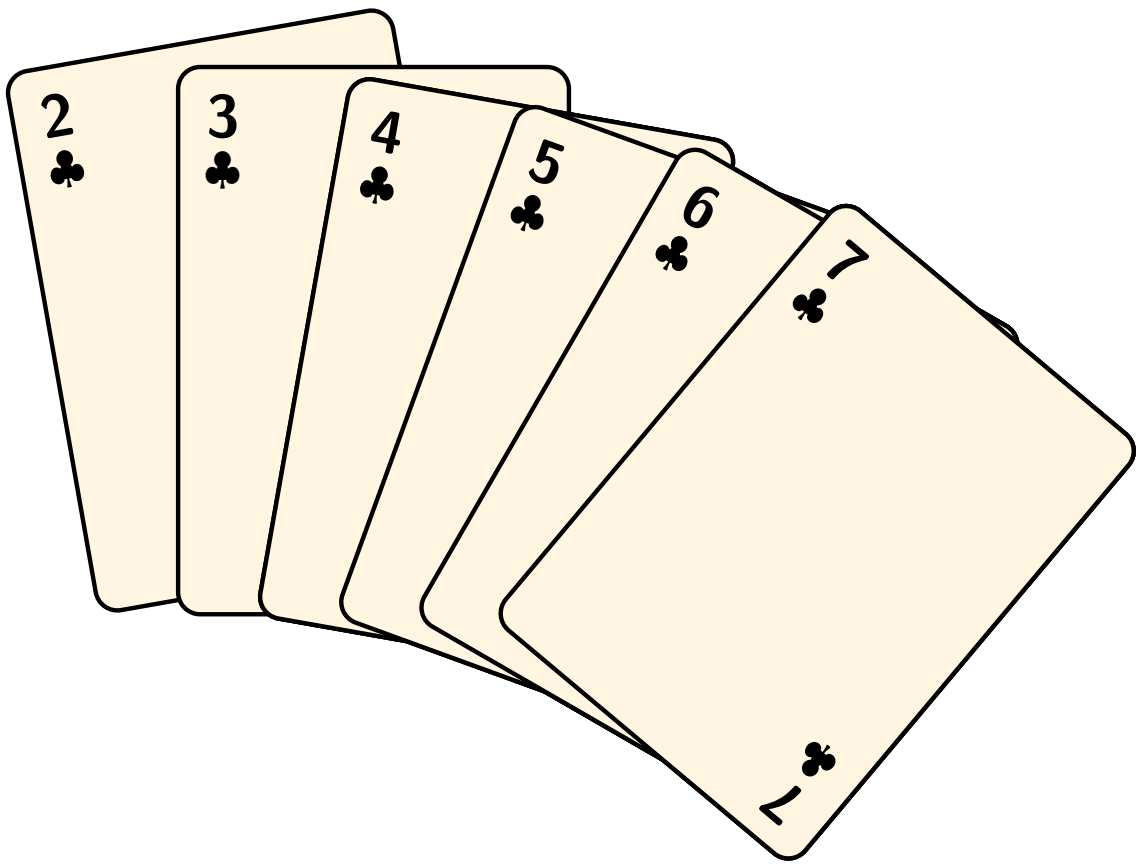
Example: Insertion Sort



Example: Insertion Sort



Example: Insertion Sort



A Functional Implementation

- inserting an element into a sorted list

```
let rec insert x = function
| []      ->  [x]
| y::ys   ->  if x <= y then x::y::ys
              else y :: insert x ys
```

A Functional Implementation

- inserting an element into a sorted list

```
let rec insert x = function
| []      ->  [x]
| y::ys   ->  if x <= y then x::y::ys
              else y :: insert x ys
```

- sorting by repeatedly inserting elements into the empty list

```
insertion_sort xs = List.fold_right insert xs []
```

where

$$\texttt{List.fold\_right}\ f\ [x_1, x_2, \ldots, x_n]\ b = f\ x_1\ (f\ x_2\ (\ldots (f\ x_n\ b)\ldots))$$

Examples

Theory File: `Week08.thy`

- the "coincidence lemma"
- computing CNFs
- Tseitin's transformation
- binary search trees

Ad

Isabelle in Innsbruck:

- IsaFoR project: **Isa***belle* **F***ormalization* **o***f* **R***ewriting* ($\sim$750,000 LOC)
- bachelor projects
- master projects
- PhD theses