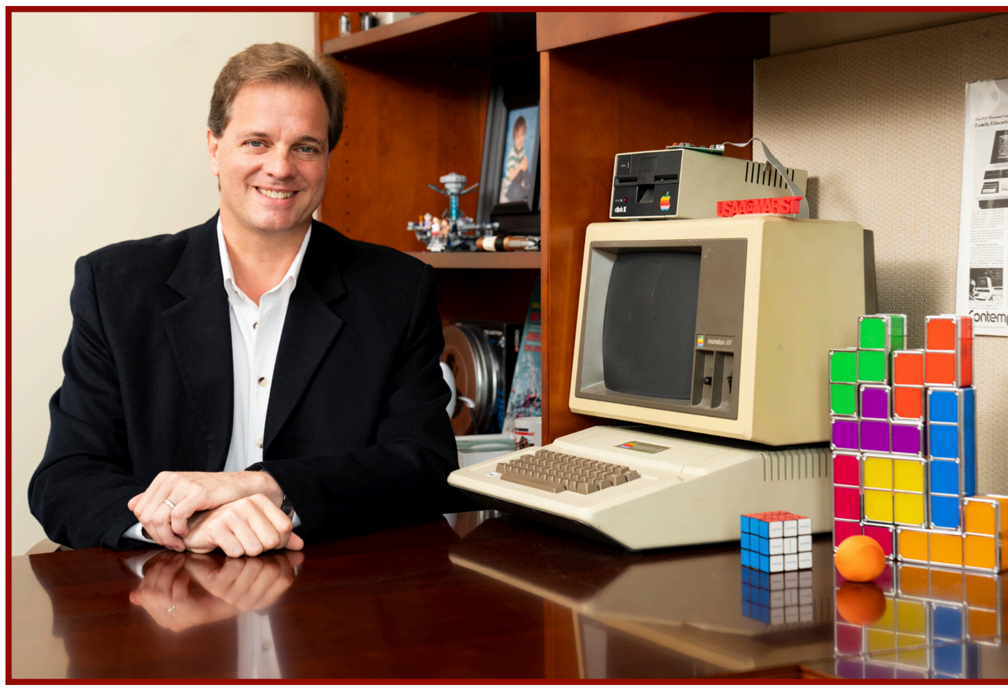
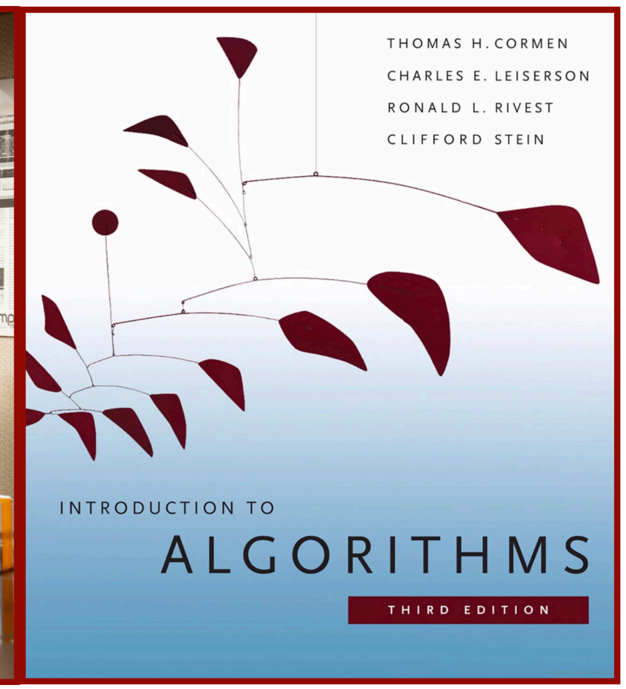
Sorting - part one





Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

Total Order / Linear Order

From Appendix B in our CLRS text:

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a \; R \; b$ or $b \; R \; a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***.

Let's consider the relation (R) of $\leq$

A total order on $\leq$ is a binary relation that satisfies …

- totality - either $a \leq b$ or $b \leq a$ or both.
- transitivity - if $a \leq b$ and $b \leq c$ then $a \leq c$.
- anti-symmetry - if $a \leq b$ and $b \leq a$ then $a = b$.

Total Order / Linear Order

From Appendix B in our CLRS text:

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a \, R \, b$ or $b \, R \, a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***. For example, the relation "$\leq$" is a total order on the natural numbers, but the "is a descendant of" relation is not a total order on the set of all people, since there are individuals neither of whom is descended from the other.

Example: natural numbers:

A total order on ≤ is a binary relation that satisfies …

- totality $0 \leq 1 \leq 2 \leq 3 \leq 4 \ldots$
- transitivity $1 \leq 2$ and $2 \leq 3$ so $1 \leq 3$
- anti-symmetry the only way $a \leq b$ and $b \leq a$ is if $a = b$
  i.e., $1 \leq 1$ and $1 \leq 1$ or $2 \leq 2$ and $2 \leq 2$

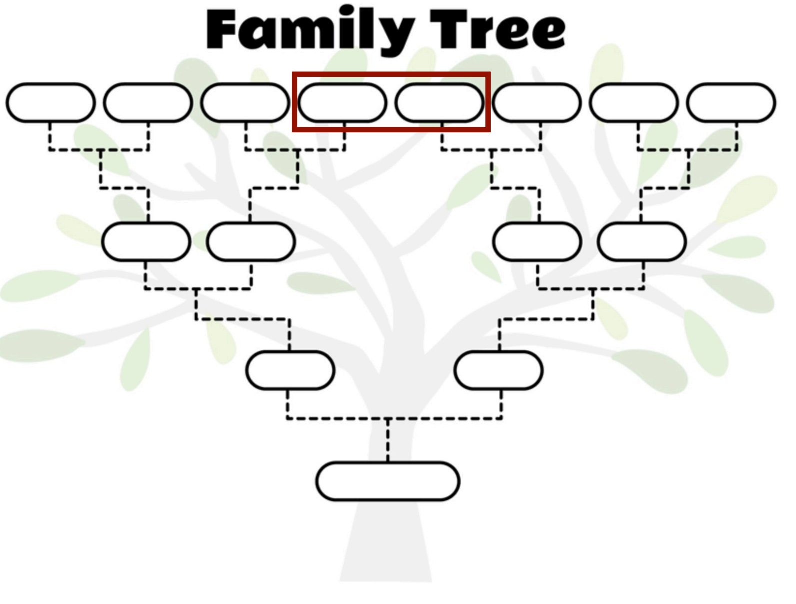
Total Order / Linear Order

From Appendix B in our CLRS text:

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a\ R\ b$ or $b\ R\ a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***. For example, the relation "$\leq$" is a total order on the natural numbers, but the "is a descendant of" relation is not a total order on the set of all people, since there are individuals neither of whom is descended from the other.

Counter-example: the "is a descendant of" relationship

- totality - Not everybody is related, so this violates totality.



Total Order / Linear Order

From Appendix B in our CLRS text:

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a\ R\ b$ or $b\ R\ a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***. For example, the relation "$\leq$" is a total order on the natural numbers, but the "is a descendant of" relation is not a total order on the set of all people, since there are individuals neither of whom is descended from the other.

Counter-example: "Rock, Paper, Scissors" (also, sports outcomes)

- transitivity - scissors < stone and stone < paper, but scissors $\nless$ paper so this violates transitivity. In fact, scissors > paper.

Total Order / Linear Order

From Appendix B in our CLRS text:

A relation R on a set A is a ***total relation*** if for all $a, b \in A$, we have $a\ R\ b$ or $b\ R\ a$ (or both), that is, if every pairing of elements of A is related by R. A partial order that is also a total relation is a ***total order*** or ***linear order***. For example, the relation "$\leq$" is a total order on the natural numbers, but the "is a descendant of" relation is not a total order on the set of all people, since there are individuals neither of whom is descended from the other.

Counter-example: the "predator, prey" relationships

- No anti-symmetry because equal ferocity but different species.
- I.e., we **have** anti-symmetry **unless** $a \leq b$ and $b \leq a$ and $a \neq b$. *Remember:* the only way $a \leq b$ and $b \leq a$ is if $a = b$

Permutations

Order matters

set size	*permu-tations*	*examples*
1	1	{ (a) }
2	2	{ (a,b), (b,a) }
3	6	{ (a,b,c), (a,c,b), (b,a,c), (b,c,a), (c,a,b), (c,b,a) }
4	24	{ (a,b,c,d), (a,b,d,c), . . . }
5	120	{ (a,b,c,d,e), . . . }
.		
.		
.		
?		

Permutations

Order matters

set size	*permu-tations*	*examples*
1	1	{ (a) }
2	2	{ (a,b), (b,a) }
3	6	{ (a,b,c), (a,c,b), (b,a,c), (b,c,a), (c,a,b), (c,b,a) }
4	24	{ (a,b,c,d), (a,b,d,c), . . . }
5	120	{ (a,b,c,d,e), . . . }
.		
.		
.		
n!		

Shuffle sort / Bogo sort / Monkey sort

```
procedure sort(in out list D)
begin
   boolean done := false;
   while (not done)
      randomly permute D
      if (D is sorted)
         done := true
      end if
   end while
   // D is returned out
end procedure
```

Shuffle sort / Bogo sort / Monkey sort

```
procedure sort(in out list D)
begin
   boolean done := false;
   while (not done)
      randomly permute D
      if (D is sorted)
         done := true
      end if
   end while
   // D is returned out
end procedure
```

In terms of *n*, the number of items in list D...

How many times through the loop until we expect it to be sorted?

How long do we expect each iteration to take?

Shuffle sort / Bogo sort / Monkey sort

```
procedure sort(in out list D)
begin
   boolean done := false;
   while (not done)
      randomly permute D
      if (D is sorted)
         done := true
      end if
   end while
   // D is returned out
end procedure
```

In terms of n, the number of items in list D...

How many times through the loop until we expect it to be sorted? **n!**

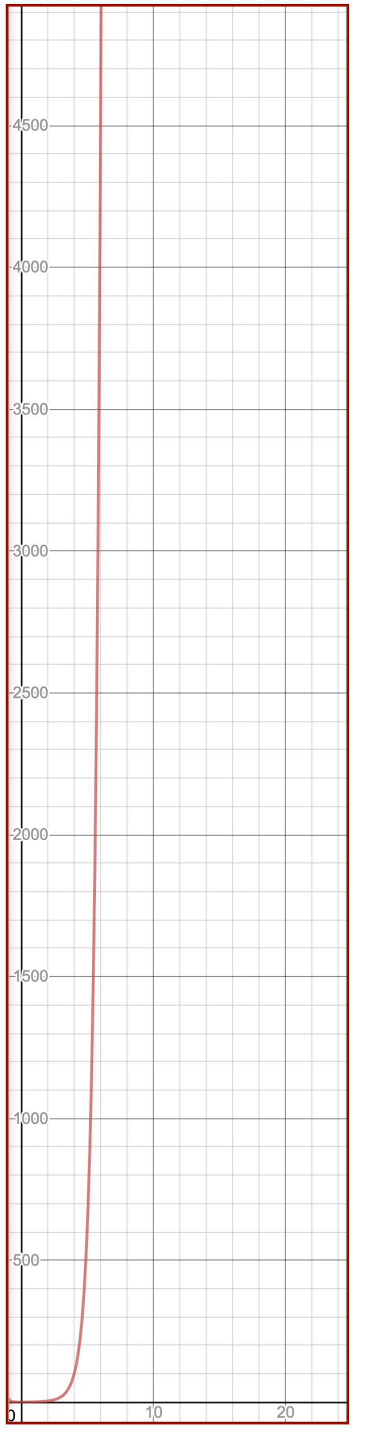
How long do we expect each iteration to take?

permute $= O(n)$

check if sorted $= O(n)$

Total time = time per iteration × number of iterations = $O(n \times n!)$

Shuffle sort / Bogo sort / Monkey sort



This is silly. And terrible.

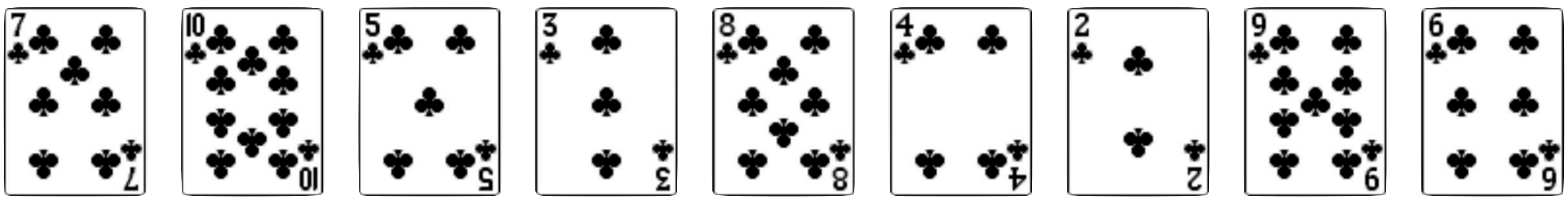
But the worst part is that this is the **expected** case. The worst case scenario is that it never halts because there is no guarantee that we'll ever produce a sorted list through random permutations. In that sense, it's scary.

To put it more technically: O(*scary*)

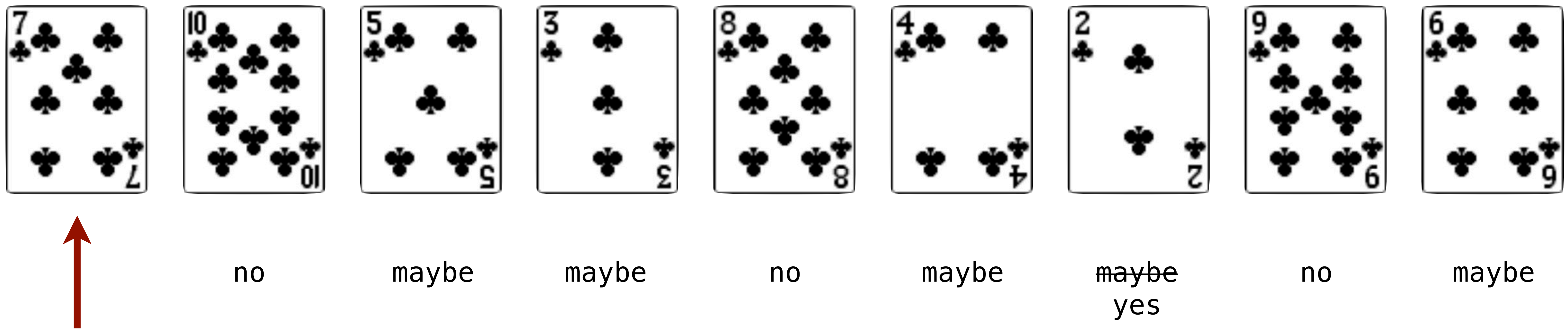
Let's not do this.

Total time = time per iteration × number of iterations = $O(n \times n!)$

Selection Sort



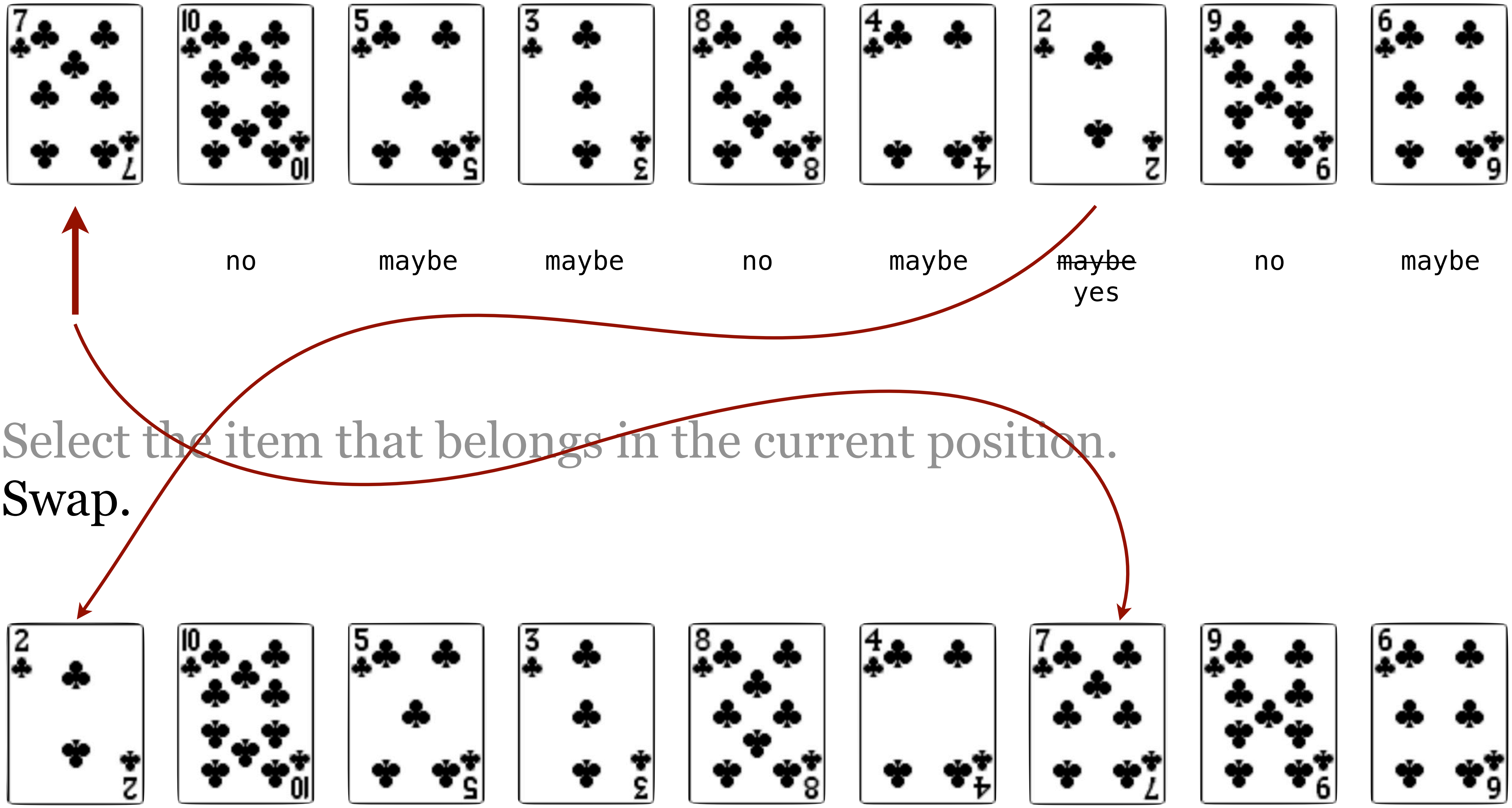
Selection Sort



↑	no	maybe	maybe	no	maybe	~~maybe~~ yes	no	maybe

Select the item that belongs in the current position.

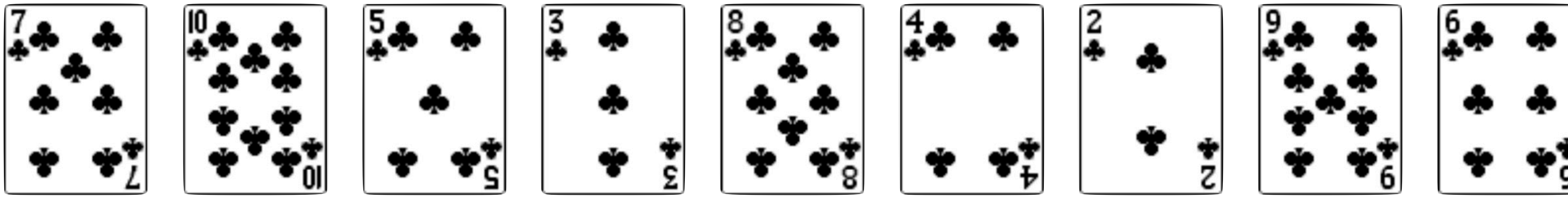
Selection Sort



Select the item that belongs in the current position.

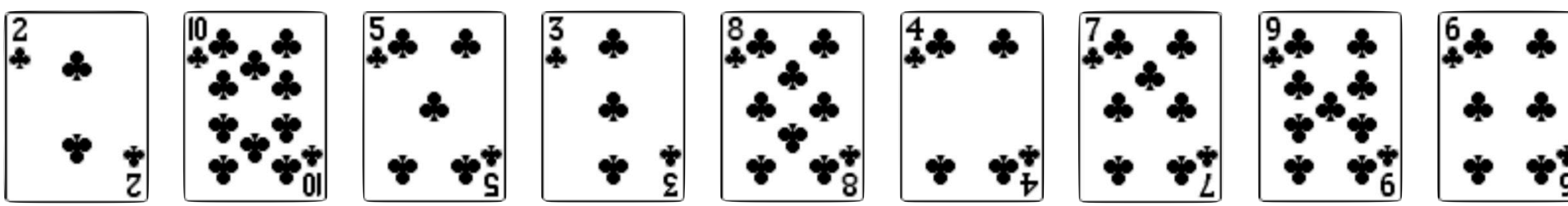
Swap.

Selection Sort



no maybe maybe no maybe ~~maybe~~ yes no maybe

Select the item that belongs in the current position.
Swap.



done maybe ~~maybe~~ yes maybe maybe maybe maybe maybe

The first item is now “sorted”. Continue from the next item.