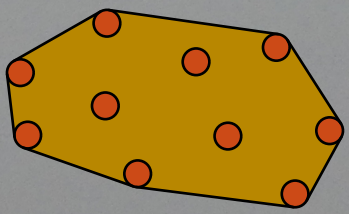
# the **Algorithm Experience**

## CONVEX HULL



# WELCOME!

With this book, you will be able to feel like a computer. You will sort cards from small to large without even knowing what you are doing.
Are you ready?

## What you need

- 7 identical envelopes
- scissors
- a large empty surface
- line indicator
- a stopwatch or timer

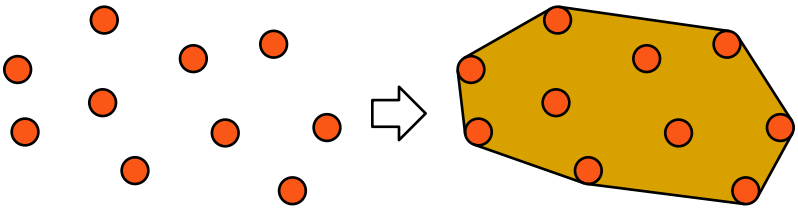


## Contents

Welcome! ..... 1
Sorting ..... 2
Materials ..... 3
Instructions ..... 4
Algorithm A ..... 6
Algorithm B ..... 7
Algorithm C ..... 8
Results ..... 9

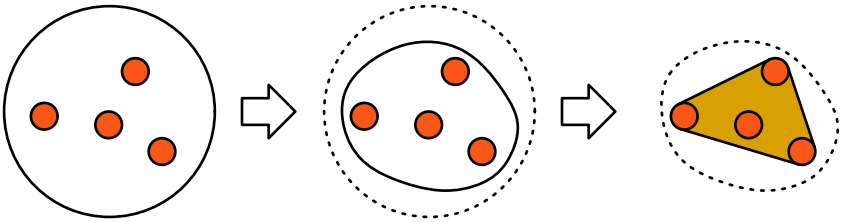
# CONVEX HULL

A convex hull is the smallest convex shape that contains an input.



We can think of it as the shape we would get when releasing a rubber band.



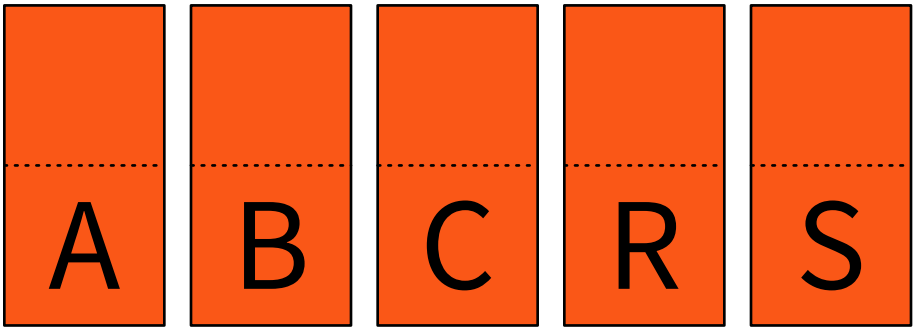


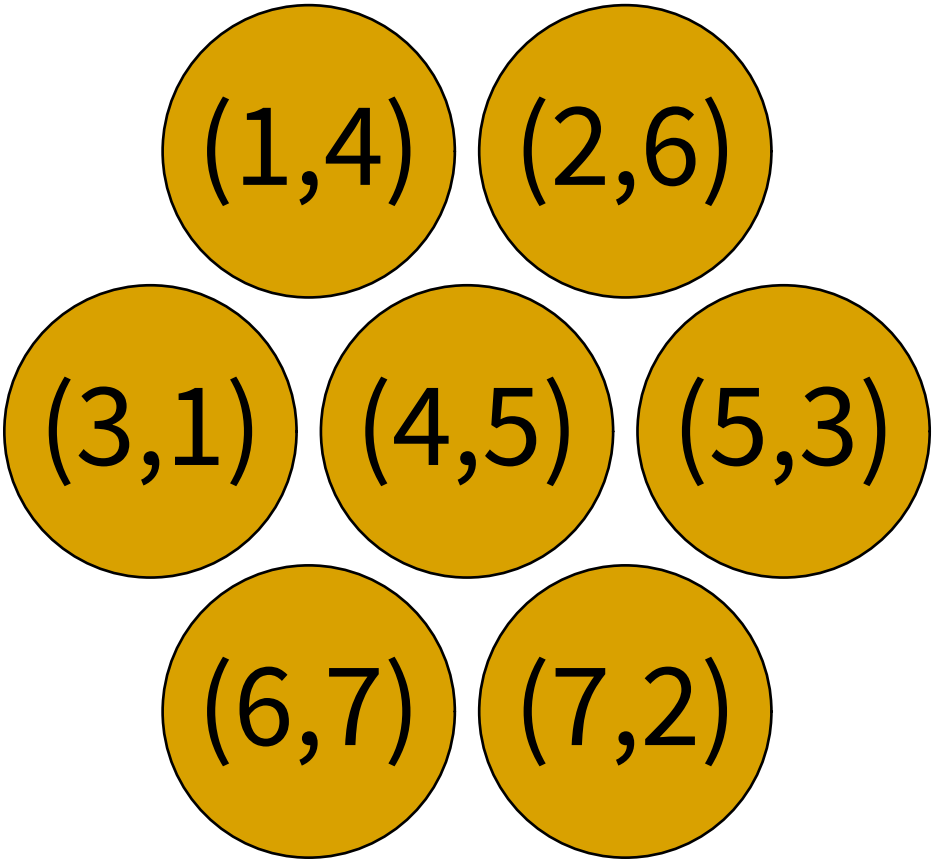
## Checks

To find a convex hull, we need to be able to check whether three points are in clockwise or counterclockwise order.

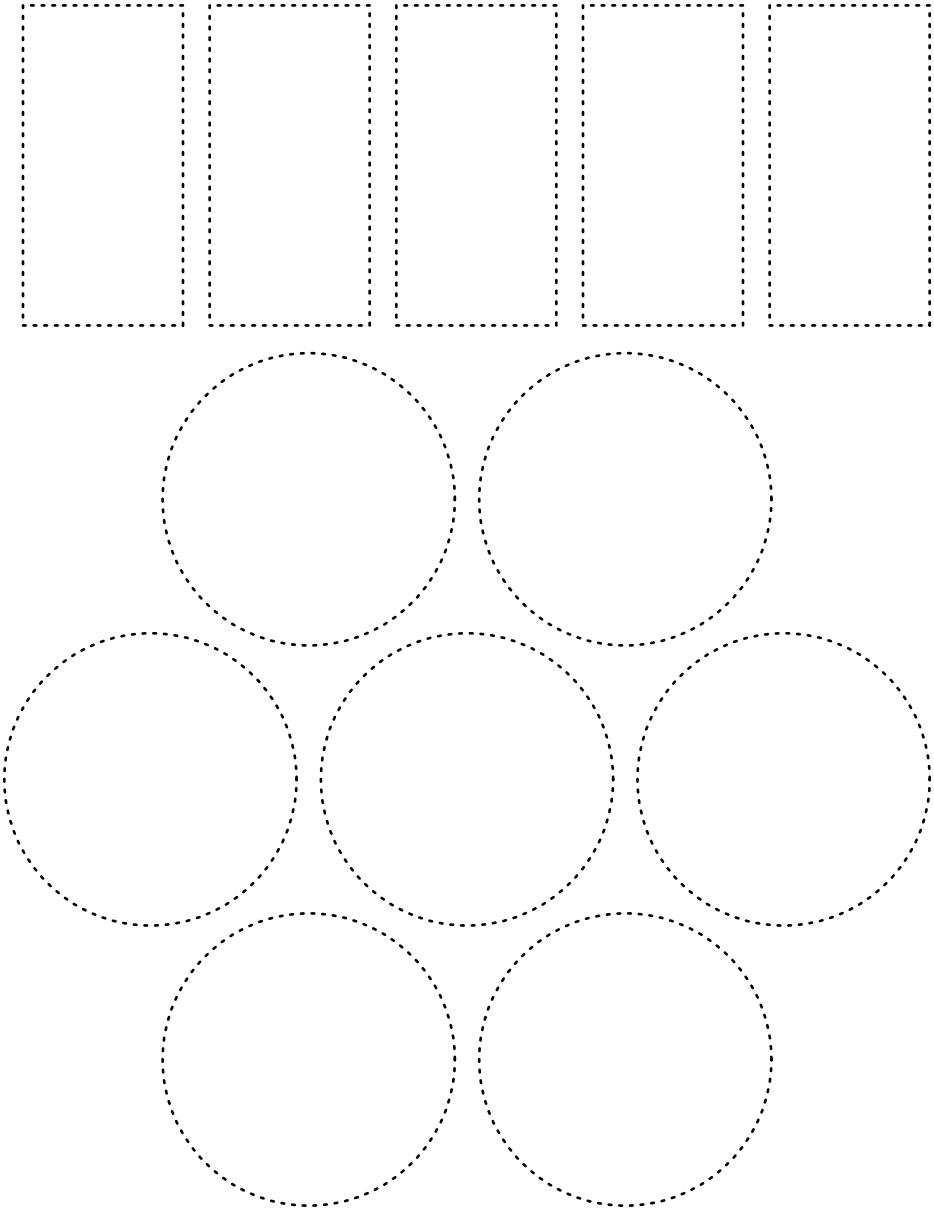
# MATERIALS

Cut these shapes, fold the dotted lines.



4



# INSTRUCTIONS

## Before the experience

- cut out the yellow point cards
- put each card into an envelope
- shuffle the envelopes
- cut and fold the orange stack markers
- pick one of the algorithms to execute
- point the marker to the first line
- start the timer

## During the experience

- follow the instructions in the algorithm
- keep track of where you are in the algorithm with the marker
- do exactly what the algorithm tells you to do (even if you think it is silly)

## After the experience

- stop the timer
- check if the result is correct
- write down your time on page 9
- try again with another algorithm

CONVEX HULL

# ALGORITHM 1




**input**: stack S
make three copies A, B, C of S
take a from A
take b from B
take c from C
**check**: is a b c ↻?
**yes**: next c
**no**: restore C
next b
is C empty?
put $\overline{a}\,\overline{b}$ on stack R
restore B and C
next a
is B empty?
restore B and C
next a
is A empty?
**output**: stack R

CONVEX HULL

# ALGORITHM 2


**input**: stack S

sort S from left to right

take p from S

put p on R

take a, b, c from R

**check**: is a b c ↻?

**yes**: put a, b, c back on R

next p

**no**: put a, c back on R

next b

not enough points on R?

next p

is S empty?

**output**: stack R

collect all points in S again

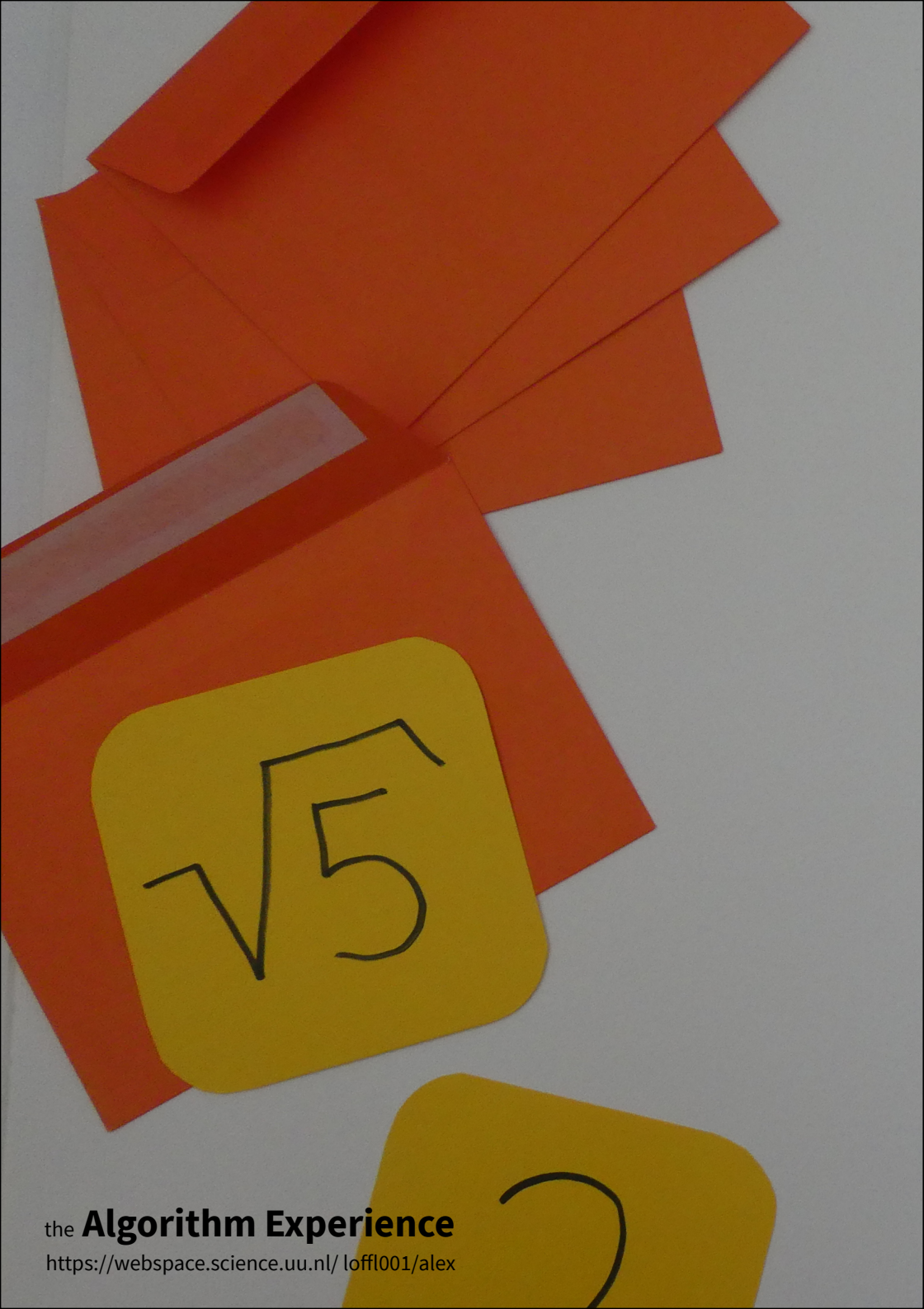
replace all y coordinates by -y

**repeat**

# SCORES

Keep track of your times here.

| date | name | algorithm | time |
| --- | --- | --- | --- |
| | | | |




the **Algorithm Experience**

https://webspace.science.uu.nl/ loffl001/alex