
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PENNSYLVANIA 1900

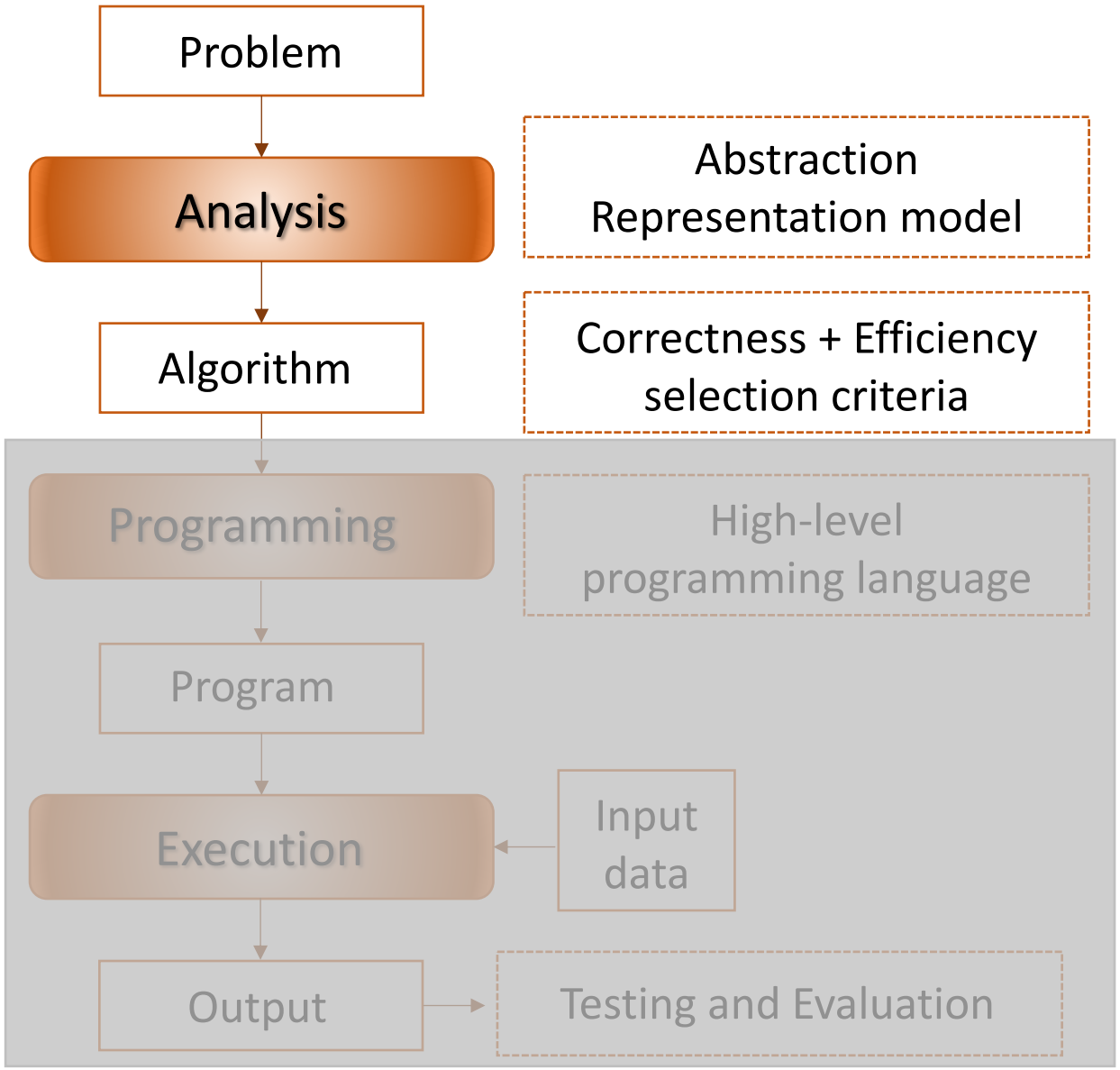
15-110 Principles of Computing – F23

Lecture 2: Algorithms!

Teacher:
Gianni A. Di Caro

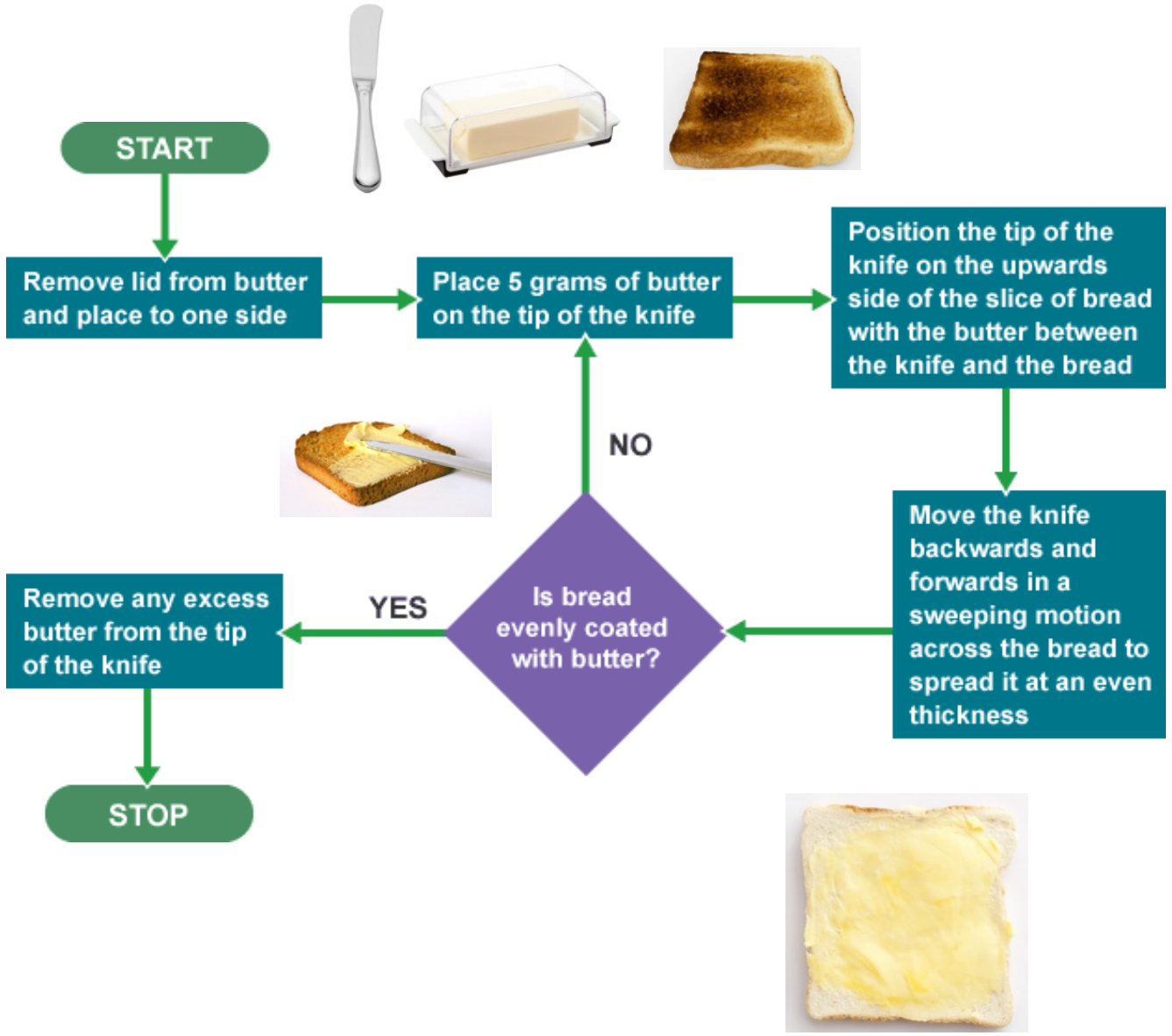
جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

Computational problem solving


Problem
Analysis
Abstraction
Representation model
Algorithm
Correctness + Efficiency
selection criteria
Programming
High-level
programming language
Program
Execution
Input
data
Output
Testing and Evaluation

- **Algorithm:**
  - ✓ A finite list / sequence of instructions that describe a **computation**
  - ✓ when the instructions are executed on a provided set of inputs, the computation proceeds *step by step* through a set of well-defined states (configurations)
  - ✓ eventually, it ends, with some outputs being produced
- **Program:**
  - ✓ Algorithm encoding using a language that the computer understands
  - ✓ *> 700 programming languages!*
  - ✓ Primitive constructs, syntax, static semantics, semantics

Algorithms: common traits


START
Remove lid from butter and place to one side
Place 5 grams of butter on the tip of the knife
Position the tip of the knife on the upwards side of the slice of bread with the butter between the knife and the bread
Move the knife backwards and forwards in a sweeping motion across the bread to spread it at an even thickness
Is bread evenly coated with butter?
NO
YES
Remove any excess butter from the tip of the knife
STOP

- ❖ Identify **essential elements** for solving the problem (*Abstraction* step)
- ✓ **Start up** actions / conditions
- ✓ Actions to execute at **each step**
- ✓ **Inspect** the situation to **make choices**
- ✓ Choices create **decision branches**
- ✓ **Repeat** sub-sets of actions
- ✓ Condition to **stop computation**
- ➢ **Save** intermediate results

A *selection* problem: Choose a snack with the lowest intake calories


Annie's
Organic Bunny Fruit Snacks BERRY PATCH
Annie's
Organic Bunny Fruit Snacks SUNRISE STRAWBERRY
Welch's
Fruit Snacks
Annie's
Organic Cheddar Bunnies
Nutrition Facts
Serving Size 1 pouch
Amount Per Serving
Calories 70
% Daily Value*
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Nutrition Facts
Serving Size 1 pouch
Amount Per Serving
Calories 70
% Daily Value*
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Nutrition Facts
Nutrition Facts
Serving Size 1 packet (28g)
Amount Per Serving
Calories 140 Calories from Fat 60
% Daily Value*
Total Fat 7g 11%
Saturated Fat 1g 5%
Trans Fat 0g
Cholesterol 0mg 0%
Sodium 240mg 10%
Total Carbohydrate 18g 6%
Dietary Fiber 0g 0%
Sugars 0g
Protein 3g
Vitamin A 0% • Vitamin C 0%
Calcium 0% • Iron 2%
BEST INGREDIENTS:
Organic Wheat Flour, Organic Expeller-pressed Sunflower Oil, Salt, Organic Cheddar Cheese (organic pasteurized milk, cheese culture, salt, enzymes), Organic Annatto Extract (for color), Yeast, Organic Paprika, Organic Cultured Whole Milk, Organic Nonfat Milk, Organic Ground Celery Seed, Organic Onion Powder, Natural Vitamin E (to protect flavor).
CONTAINS MILK AND WHEAT INGREDIENTS.
Made on shared equipment that also processes soy.

- ❖ You have to choose among **5 snacks**
- ✓ You want to choose the one with the lowest intake calories

  - ❖ Identify **essential elements** for solving the problem (*Abstraction* step)
  - ✓ **Start up** actions / conditions
  - ✓ Actions to execute at **each step**
  - ✓ **Inspect** the situation to **make choices**
  - ✓ Choices create **decision branches**
  - ✓ **Repeat** sub-sets of actions
  - ✓ Condition to **stop computation**
  - ✓ **Save** intermediate results

- Snack pack and its nutritional facts are the essential elements
- Let's assume the snacks are in a heap in front of you

A *selection* problem: Choose a snack with the lowest intake calories


Annie's
Bunny Fruit Snacks
Nutrition Facts
Serving Size 1 pouch
Calories 70
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Annie's
Bunny Fruit Snacks
Nutrition Facts
Serving Size 1 pouch
Calories 70
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Welch's
Fruit Snacks
Nutrition Facts
Annie's
Organic Cheddar Bunnies
Nutrition Facts
Serving Size 1 packet (28g)
Calories 140 Calories from Fat 60
Total Fat 7g 11%
Saturated Fat 1g 5%
Trans Fat 0g
Cholesterol 0mg 0%
Sodium 240mg 10%
Total Carbohydrate 18g 6%
Dietary Fiber 0g 0%
Sugars 0g
Protein 3g
Vitamin A 0% • Vitamin C 0%
Calcium 0% • Iron 2%
BEST INGREDIENTS:
Organic Wheat Flour, Organic Expeller-pressed Sunflower Oil, Salt, Organic Cheddar Cheese (organic pasteurized milk, cheese culture, salt, enzymes), Organic Annatto Extract (for color), Yeast, Organic Paprika, Organic Cultured Whole Milk, Organic Nonfat Milk, Organic Ground Celery Seed, Organic Onion Powder, Natural Vitamin E (to protect flavor).
CONTAINS MILK AND WHEAT INGREDIENTS.
Made on shared equipment that also processes soy.

- ❖ Identify **essential elements** for solving the problem (*Abstraction* step)
- ✓ **Start up** actions / conditions
- ✓ Actions to execute at **each step**
- ✓ **Inspect** the situation to **make choices**
- ✓ Choices create **decision branches**
- ✓ **Repeat** sub-sets of actions
- ✓ Condition to **stop computation**
- ✓ **Save** intermediate results

1. At random, pick-up a ***first* snack** from the heap and check its calories
2. Write down or memorize its calories (i.e., **save the state** of search)
3. Put the snack aside, in the *selected snack* location (e.g., to your left)
4. At random, pick-up a snack from the heap
5. Read its calories
6. **If** its calories are lower than the previously selected snack, put the current snack in the *selected snack* location and write its calories
7. Remove the previous snack from the *selected snack* location and put it in the *rejected* location (e.g., to your right)
8. **Instead**, if the calories are higher than the previously selected snack, put the current snack in the *rejected* location
9. **Repeat** steps 4 – 8 four times
10. The snack in the *selected snack* location is the one you'll eat!

A *selection* problem: Choose a snack with the lowest intake calories

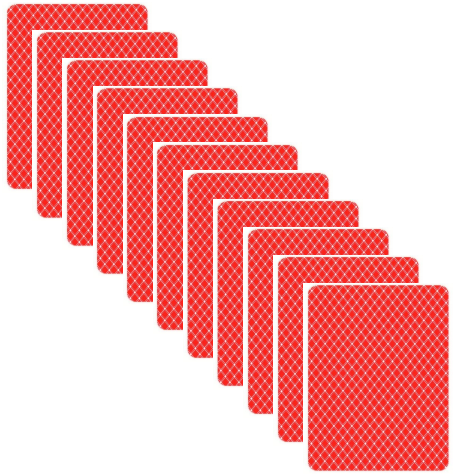

Annie's
Nutrition Facts
Serving Size 1 pouch
Calories 70
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Annie's
Nutrition Facts
Serving Size 1 pouch
Calories 70
Total Fat 0g 0%
Sodium 45mg 2%
Total Carb. 17g 6%
Sugars 11g
Protein 0g
Vitamin C 100%
Welch's
Fruit Snacks
Nutrition Facts
Annie's
Cheddar Bunnies
Nutrition Facts
Serving Size 1 packet (28g)
Calories 140 Calories from Fat 60
Total Fat 7g 11%
Saturated Fat 1g 5%
Trans Fat 0g
Cholesterol 0mg 0%
Sodium 240mg 10%
Total Carbohydrate 18g 6%
Dietary Fiber 0g 0%
Sugars 9g
Protein 3g
Vitamin A 0% • Vitamin C 0%
Calcium 0% • Iron 2%
BEST INGREDIENTS:
Organic Wheat Flour, Organic Expeller-pressed Sunflower Oil, Salt, Organic Cheddar Cheese (organic pasteurized milk, cheese culture, salt, enzymes), Organic Annatto Extract (for color), Yeast, Organic Paprika, Organic Cultured Whole Milk, Organic Nonfat Milk, Organic Ground Celery Seed, Organic Onion Powder, Natural Vitamin E (to protect flavor).
CONTAINS MILK AND WHEAT INGREDIENTS.
Made on shared equipment that also processes soy.

- ❖ You have to choose among **5 snacks**
- ✓ You want to choose the one with the lowest intake calories

- ❖ Typically, there's not a SINGLE solution to a problem!
- ✓ **Different variants** are possible for the previous algorithm

  - ➢ Don't remove the snack from the initial heap.
    → Need to **save/memorize** the information about the best snack so far to retrieve it at the end
    - Is it EFFICIENT?
  - ➢ Order (how?) the snacks according to increasing calories and select the first in the ordered list
    - Is it EFFICIENT?
  - ➢ ...

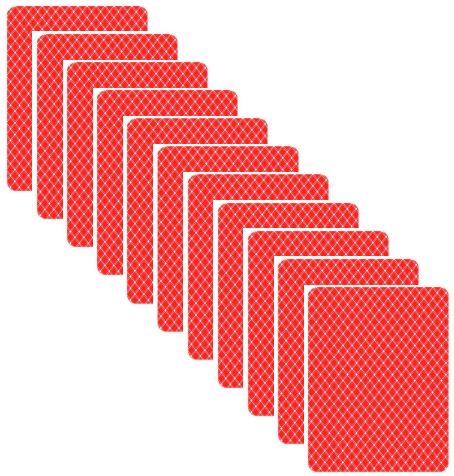
A simple *search* problem: Find the card!



- You are given a set of cards (covered) as show in the figure
- Cards are uniquely numbered from 1 to 100, but of course they aren't necessarily placed in that order!

➢ You must find the card with number 100

A simple search for *max/min* problem



- You are given a set of cards (covered) as show in the figure
- Cards are uniquely numbered from 1 to 100, but:
  - You don't know it,
  - You don't know how many cards are there
  - Cards aren't necessarily placed in the 1-100 order!

➢ You must tell the **highest** and the **lowest** card values in the set

✓ You can *write* down things / Memory!

❖ WHAT IF you can only inspect the cards, but not put them aside? (e.g., cards are on a computer screen!)

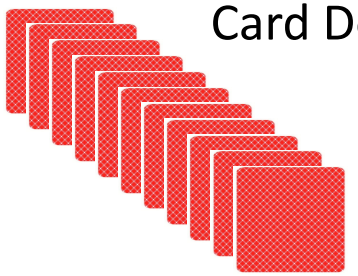
A simple search for *max/min* problem

A detailed step-by-step solution

1. Pick up the first card from the deck pile
2. Record down the number and remove the card from the deck (put it in done pile)
3. Assign the number to min value and to max value

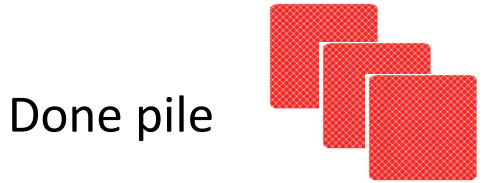
4. Pick up the next card from the deck
5. Look at the number, n, and remove the card from the deck
6. If the number is higher that current max value: max value becomes n
7. If the number is lower that current min value: min value becomes n

8. Repeat 4-7 **until no more cards in deck**
9. Read/Output min value and max value
10. Stop


Card Deck

Min value: XX

Max value: YY


Done pile

A simple search for *max/min* problem: Solution based on sorting

Another possible solution, that has an issue: how do we do sort? → Need another algorithm ☺

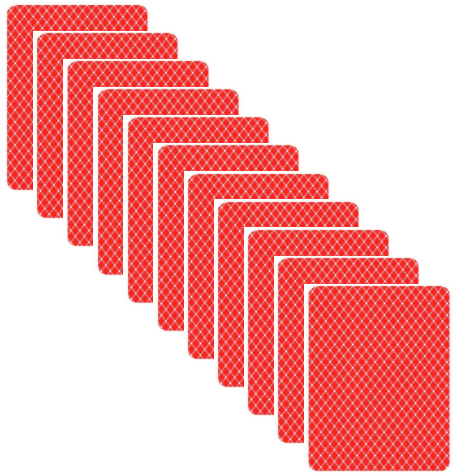
1.) Pick up the first card, pick up the pencil and note down the number that is written on the card on the piece of paper that is on the left side .

2.) repeat the first step by picking up the NEXT card and note down the value.

3.) repeat steps 1-2 until you run out of cards

4.) compare the numbers that you have written down by **sorting** them in ascending order. Identify the lowest and the highest values.

A *sorting* problem (and you know the numbers)



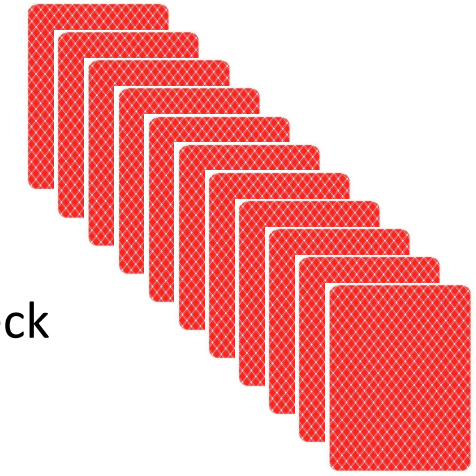
- You are given a set of cards (covered) as show in the figure
- Cards are uniquely numbered from 1 to 100, but cards aren't necessarily placed in the 1-100 order!
- You know that cards are from 1 to 100

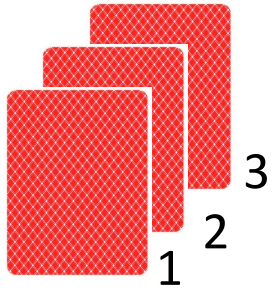
➢ You must **sort** the cards in the 1 → 100 order

Find an algorithm that explicitly makes use of the information that card numbers are between 1 and 100

A *sorting* problem (and you know the numbers)

Unsorted deck




3
2
1

New, sorted card pile

1. Check if the first card from unsorted deck is 1:
    (a) If yes: Put the card at the end of the new pile and move to step 2
    (b) If no: Place the card at the end of the unsorted deck. Go back to step 1.

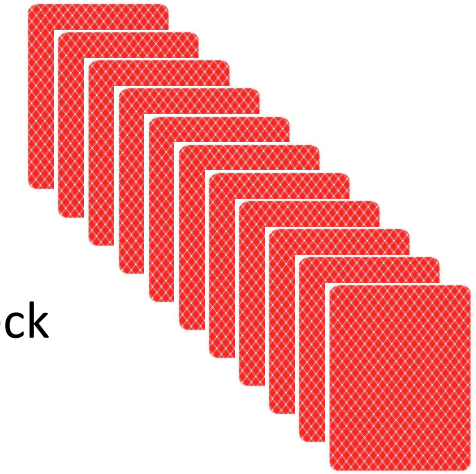
2. Check if the first card from unsorted deck is 2:
    (a) If yes: Put the card at the end of the new pile
    (b) If no: Place the card at the end of the unsorted deck. Go back to step 2.

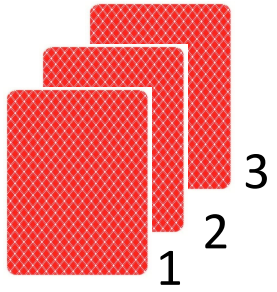
…

100. Check if the first card from unsorted deck is 100:
    (a) If yes: Put the card at the end of the new pile and STOP
    (b) If no: Place the card at the end of the unsorted deck. Go back to step 100.

A *sorting* problem (and you know the numbers)

Unsorted deck




3
2
1

New, sorted card pile

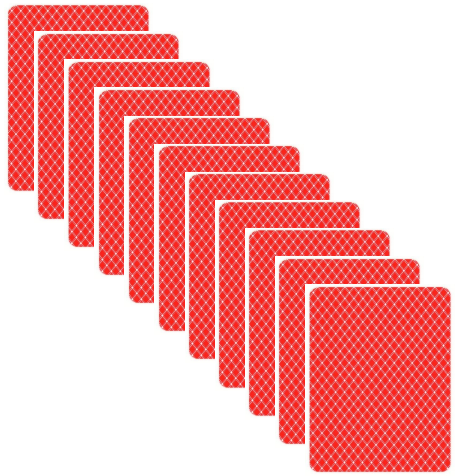
For all numbers from 1 to 100, repeat the following steps:

(a) Check if the first card has that number:

1. If yes: Put the card at the end of the new card pile.
2. If no: Place the card at the end of the unsorted deck.

- How do you sort the cards in the **reverse** order?

A more general *sorting* problem: you don't know the numbers



- You are given a set of cards (covered) as show in the figure
- Same problem as before, but now you don't which is the range of values of the cards, you don't know how many cards are there

➢ You must **sort** the cards in increasing order

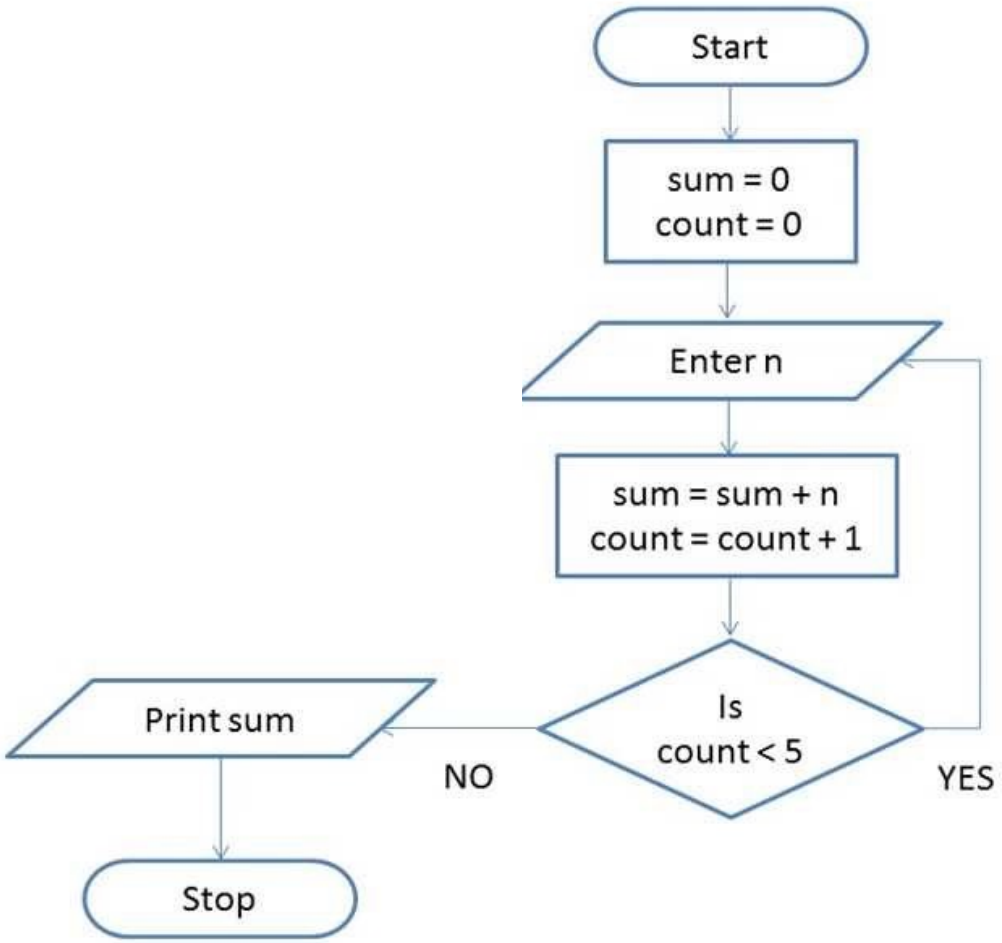
One possible solution:

1. Pick up first card from deck
2. Add the card to sorted pile
3. Pick up first card from deck
4. If card value greater than top card on sorted pile
   - Then add card on top of sorted pile
5. Instead, if card value is lower that bottom card on sorted pile
   - Then add card to the bottom of sorted pile
6. If neither 4 or 5 conditions are satisfied, *insert* card in sorted pile
7. Repeat 3-6 until no cards in card deck

Insert:

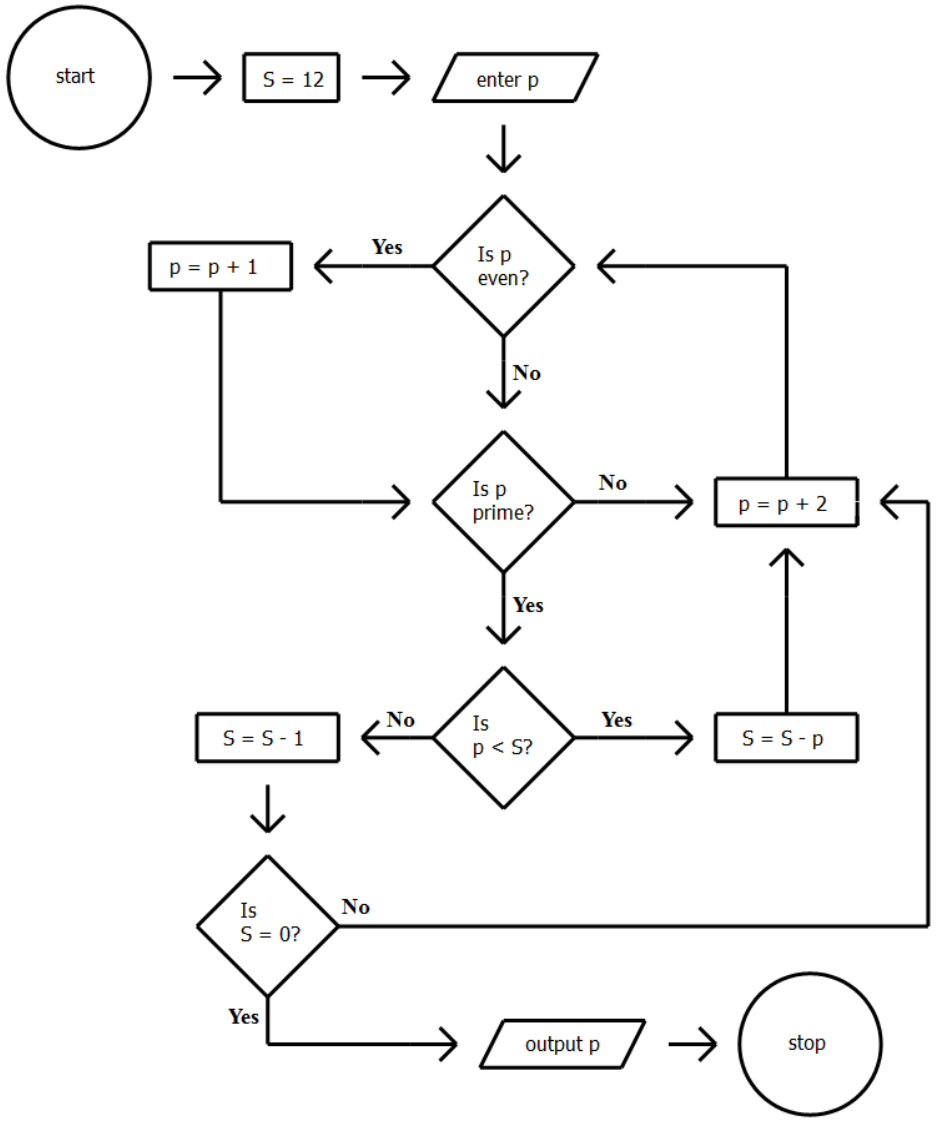
1. If distance from bottom is less than distance from top, start from bottom
2. Otherwise, start from top
3. Inspect the first two cards from start position
4. If card value is lower than first and higher than second, insert card after first
5. Otherwise, set first card as new start position
6. Repeat 3-5 until the new card is inserted

More on algorithms: Decode a flow chart


Start
sum = 0
count = 0
Enter n
sum = sum + n
count = count + 1
Is
count < 5
NO
YES
Print sum
Stop

What is this flow chart doing?

More on algorithms: Decode a flow chart


start
S = 12
enter p
Is p even?
Yes
p = p + 1
No
Is p prime?
No
p = p + 2
Yes
Is p < S?
No
S = S - 1
Yes
S = S - p
Is S = 0?
No
Yes
output p
stop

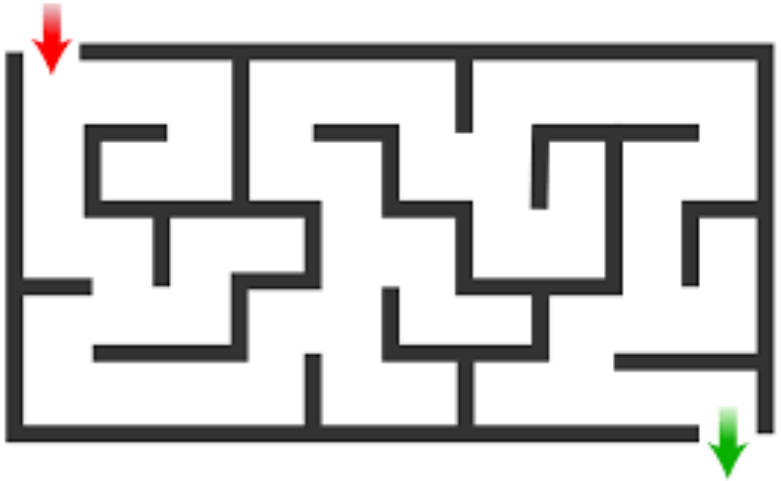
What is this flow chart doing?

What is the output for p=3?

More problems: Navigating out of a maze

Getting out of a maze!

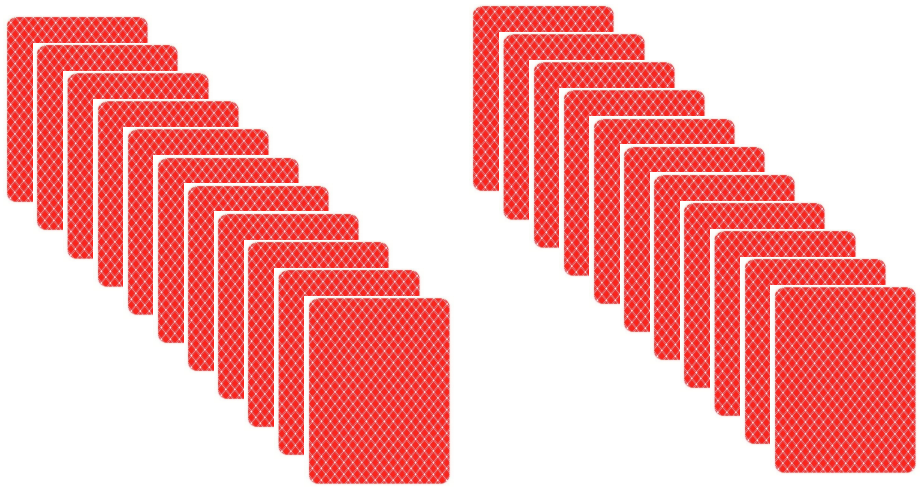
- Describe the step-by-step algorithm (sequence of instructions) to get out of the maze (from the red to the green open accesses)
  - You don’t know distances and don’t know how measure them, but you have a sensor that let you know when you’re hitting a wall
  - You can move straight forward and rotate in place of the desired angle (clockwise or anti-clockwise)



More problems: Merge 2 sorted piles of cards

Split the job among your friends!

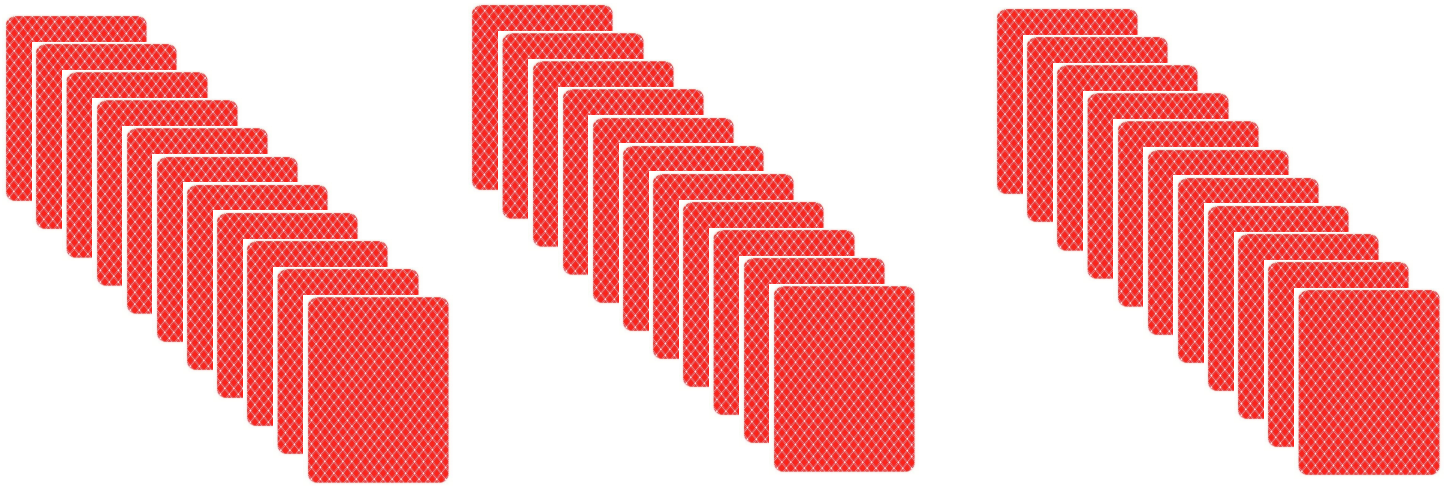
- Still need to sort a pile of n cards, and you don't know their ranges.
- Luckily, you have a friend who can help you in the task. You split the cards in two piles, giving each pile to each one of you
- Both of you know how to sort a pile of cards, such that both of you will return a sorted pile of cards (e.g., sorted in ascending order).
- Now you need to use / merge the sorted piles to create a single sorted pile of cards, which will be your output



More problems: Merge n sorted piles of cards

Split the job among your friends!

- Still need to sort a pile of n cards, and you don't know their ranges.
- Luckily, you have k friends who can help you in the task. You split the cards in k piles, giving each pile to each one of your friends (what if n is not precisely divisible by k?)
- Each friend know how to sort a pile of cards, such that he/she will return a sorted pile of cards (e.g., sorted in ascending order).
- Now you need to use / merge the sorted piles to create a single sorted pile of cards, which will be your output



More problems: Pair the socks in a heap!

Pair the socks!

- Taking care of organizing clothes at home, sometimes can be a very frustrating job! Imagine you have an unorganized heap of (many) socks, and you need to pair all the socks and put them aside to bring them to the wardrobe.
- Socks might have different colors, patterns, shape.
- Define two different algorithms that can achieve the task.
- Discuss which one of the two algorithms you think is more efficient (i.e., it would take less time to complete)