Lecture 2:
Data Structure and Algorithms

MEng Robotics and AI

Daniel Tozadore (d.tozadore@ucl.ac.uk)

UCL Computer Science

From previous sessions…

Modelling the Robot/Maze

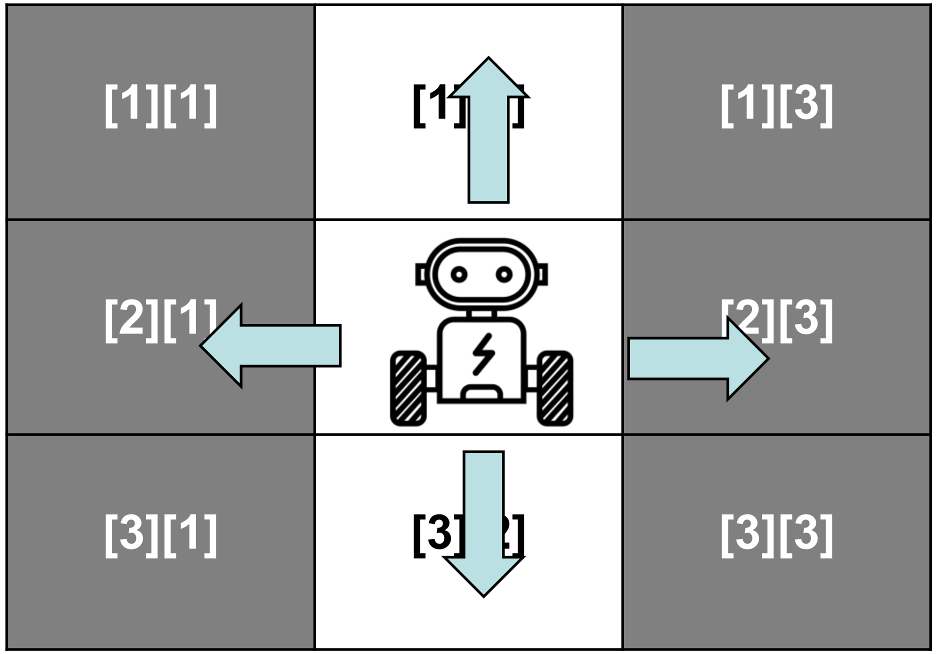
Exercises session 1

1. Create the following scenario in MATLAB
   a. Create a matrix named map
   b. Attribute the obstacle values

2. Create a function to print the map

3. Insert the robot:
   a. Create another matrix [1, 2] to store the robot position
   b. Adapt the function of item 2 to print “2” where the robot is

4. Implement the function “move(direction, robot_pos)” to move the robot
   a. Test your function with values hardcoded defined (meaning declaring the values in your code)



[1][1]	[1][2]	[1][3]
[2][1]		[2][3]
[3][1]	[3][2]	[3][3]

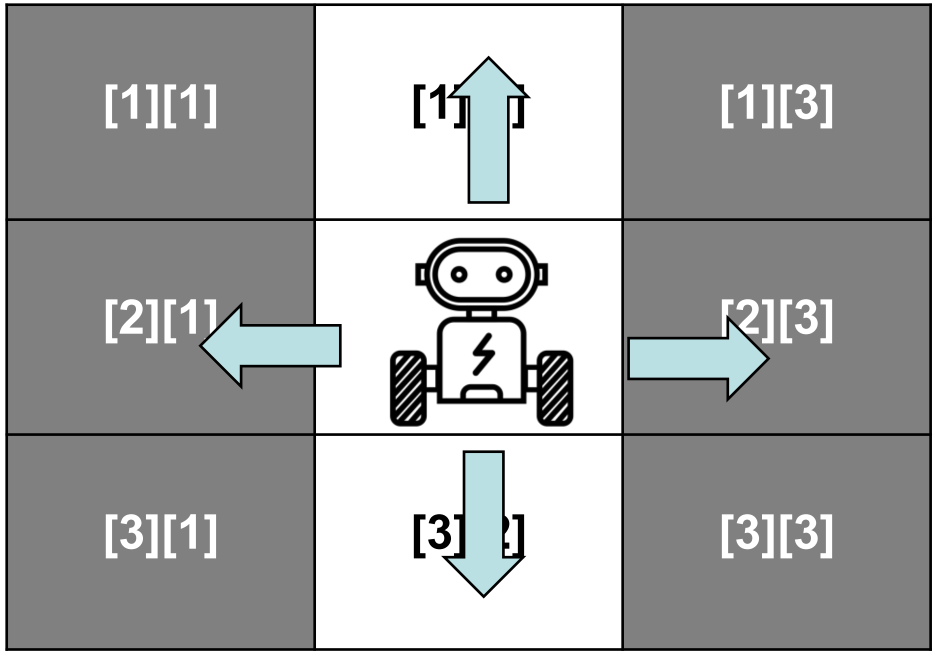
Exercises session 2

1. Implement the verifications in the functions implemented in Session 1:
    a. Verify if the cell is free before moving
    b. Verify whether the next move goes out of the map (edges)

2. Implement a function that given two numbers n and m, create a mesh map of size n*m with all the cells

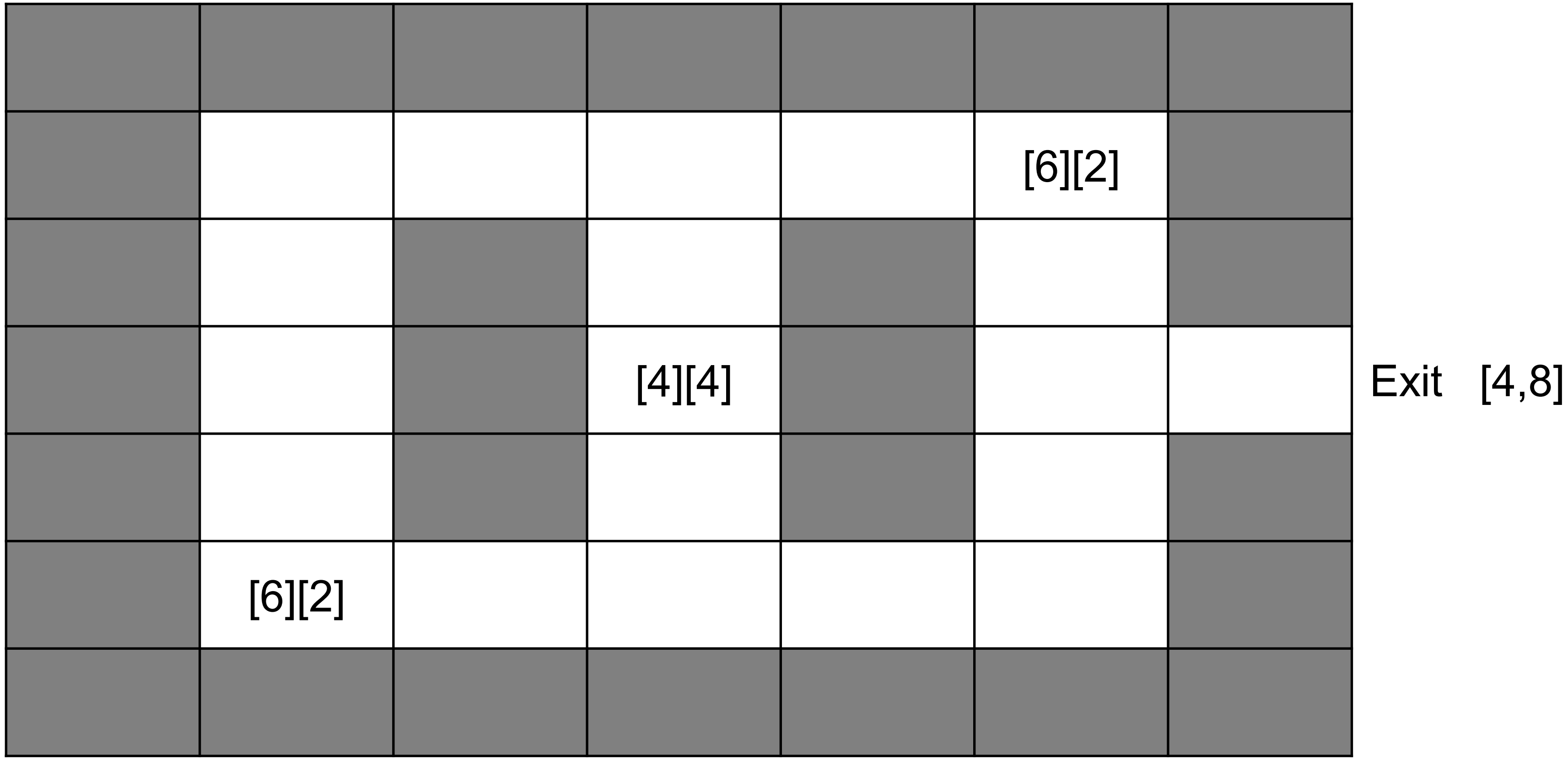
3. Make sure your print_map function handles the new map

4. Adapt your code so you can take the next direction from the keyboard (users insert next direction)



Exercises session 3

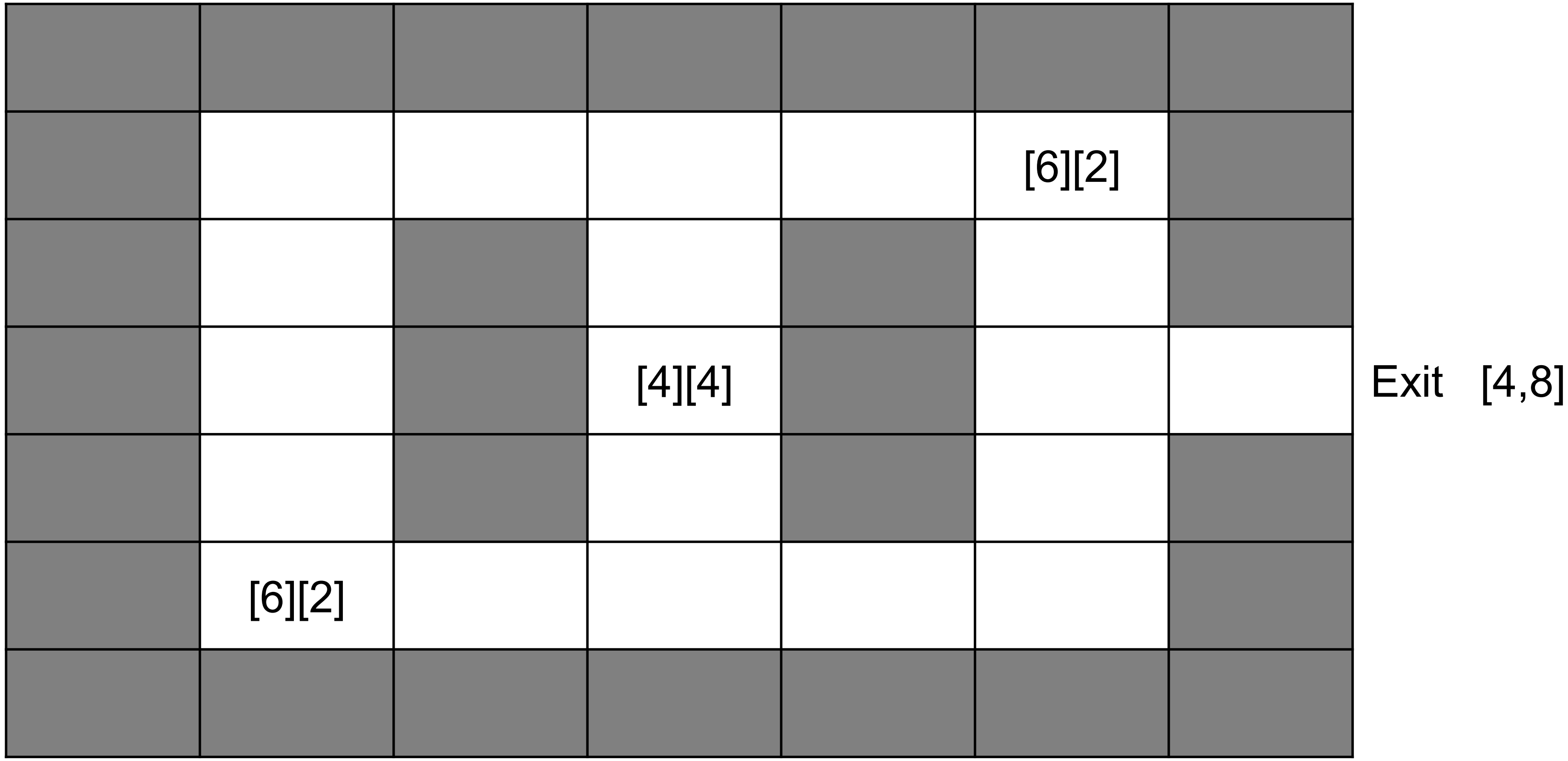
1. Test your code in the following mesh maze (7x7):
   1. Don’t forget to test the condition for out of map (successfully exited)
2. Modify your code to search for the exit autonomously (checking all the directions)
3. Count how many steps your robot took to find the way autonomously
4. Count computational cost of the algorithm you developed
5. Cells to start
   1. [6][2]
   2. [4][4]
   3. [2][6]



6. Work in pairs to handout tomorrow a best-solution for automatically scape the maze

Exercises session 3

1. Test your code in the following mesh maze (7x7):
    1. Don’t forget to test the condition for out of map (successfully exited)
2. Modify your code to search for the exit autonomously (checking all the directions)
3. Count how many steps your robot took to find the way autonomously
4. Count computational cost of the algorithm you developed
5. Cells to start
    1. [6][2]
    2. [4][4]
    3. [2][6]



6. Work in pairs to handout tomorrow a best-solution for automatically scape the maze

Improving the search algorithm

1. How well your measures performed?

2. What part took the longest?

3. How to improve that?

▨	▨	▨	▨	▨	▨	▨
▨					[6][2]	▨
▨		▨		▨		▨
▨		▨	[4][4]	▨		
▨		▨		▨		▨
▨	[6][2]					▨
▨	▨	▨	▨	▨	▨	▨

Improving the search algorithm

1. How well your measures performed?

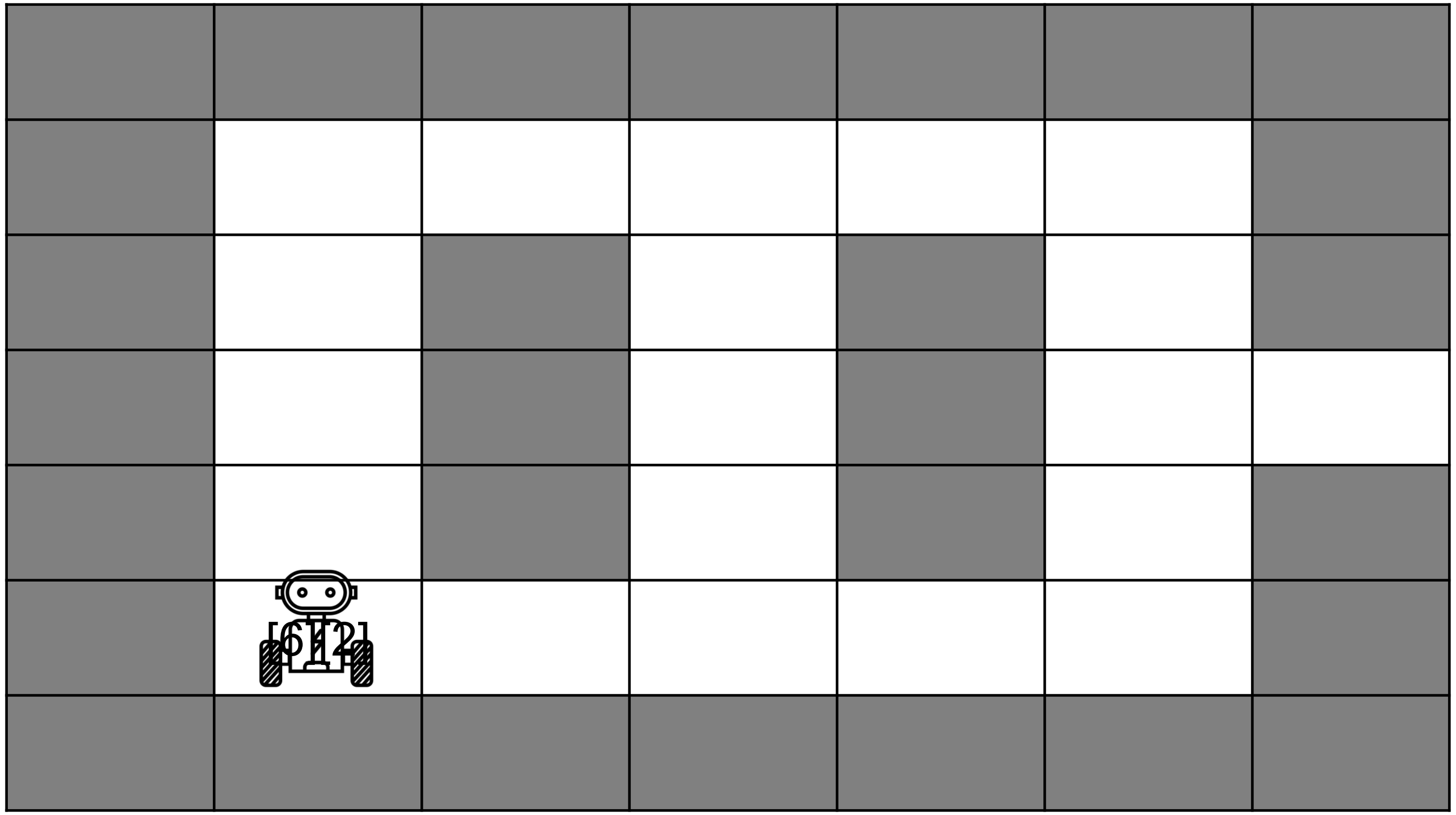
2. What part took the longest?

3. How to improve that?

▨	▨	▨	▨	▨	▨	▨
▨					[6][2]	▨
▨		▨		▨		▨
▨		▨	[4][4]	▨		
▨		▨		▨		▨
▨	[6][2]					▨
▨	▨	▨	▨	▨	▨	▨

Creating a set of visited nodes!

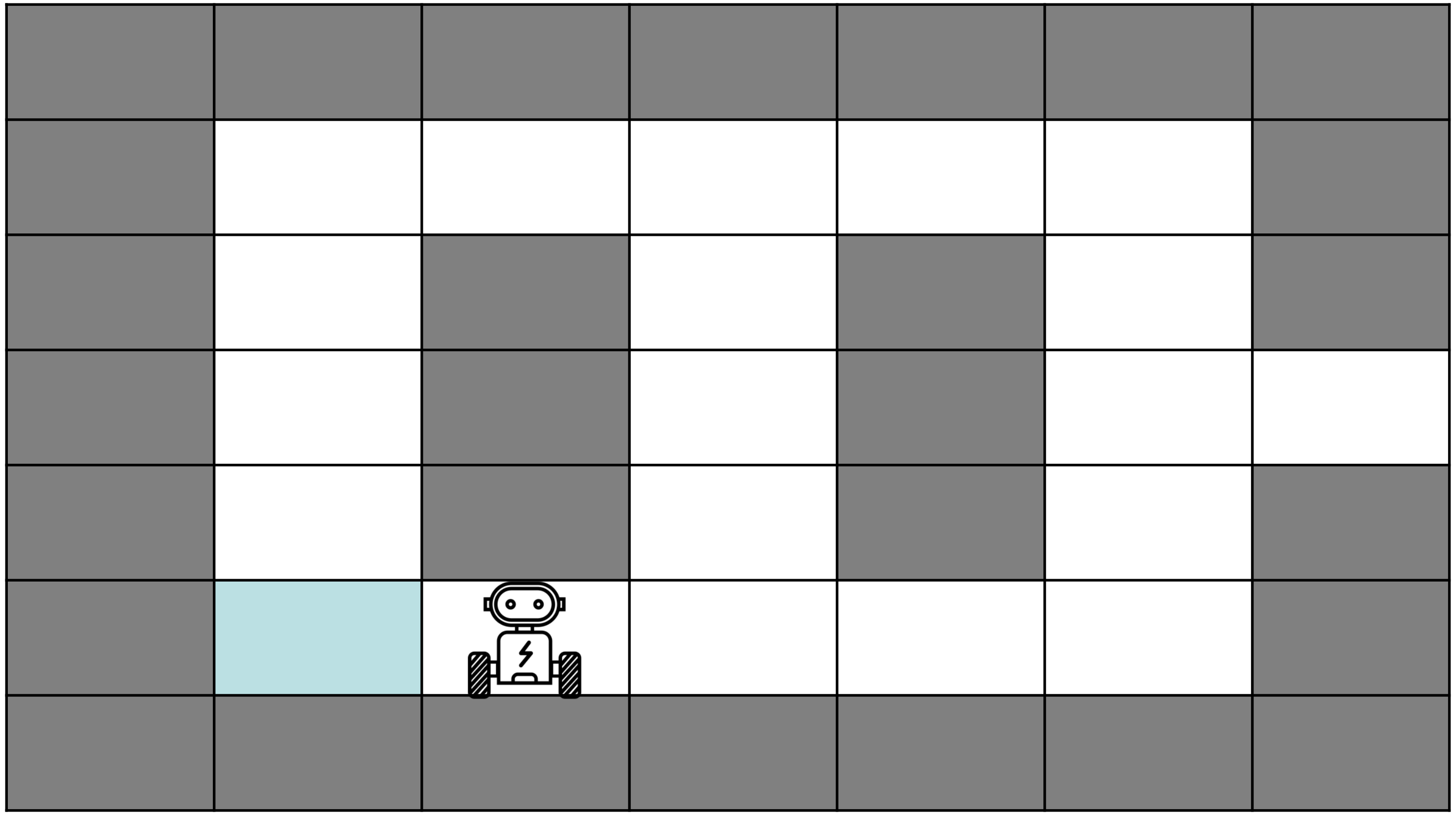
Improving the search algorithm



Visited nodes:

[6,2]

Improving the search algorithm

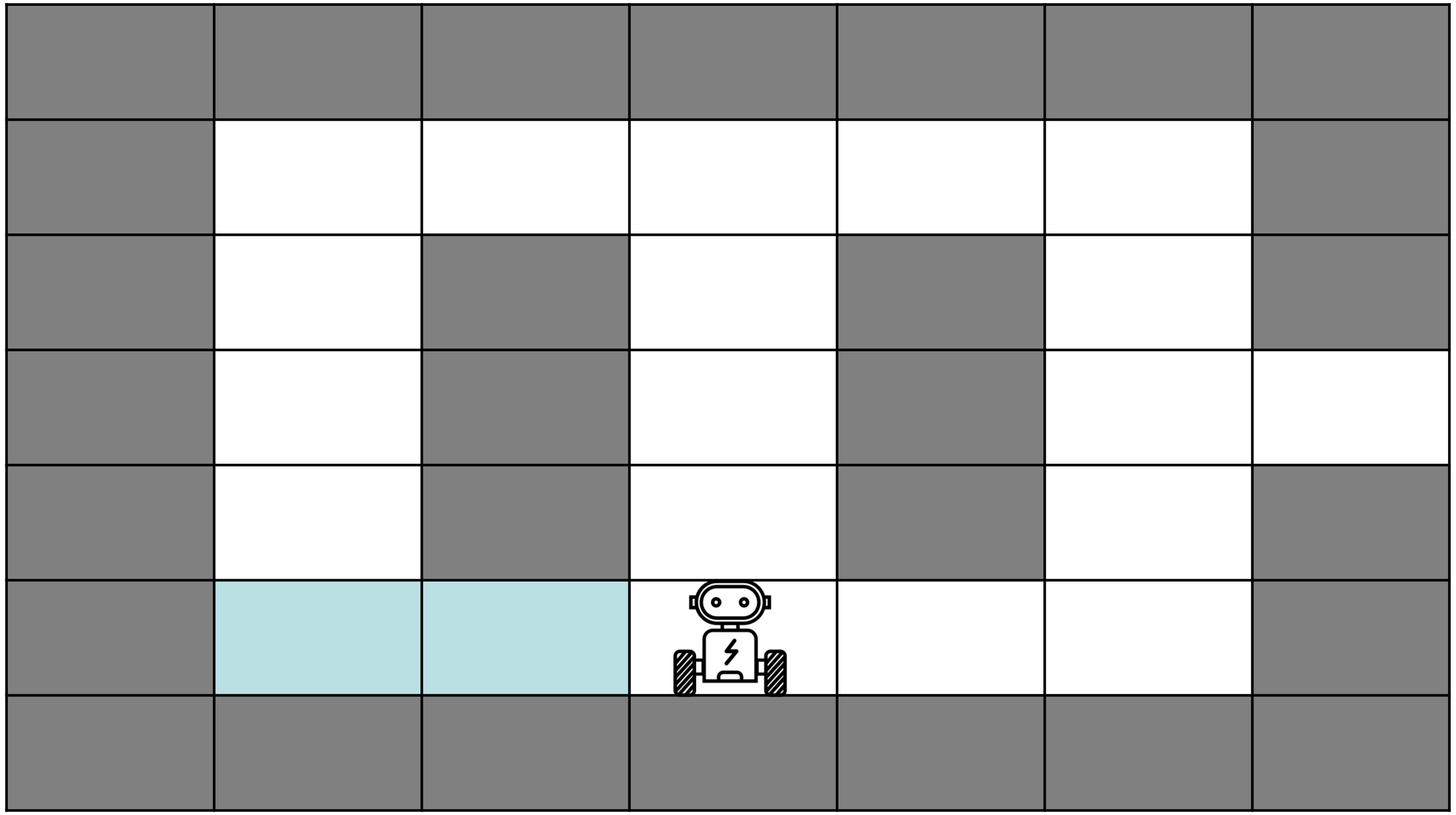


Visited nodes:

[6,2]

[6,3]

Improving the search algorithm



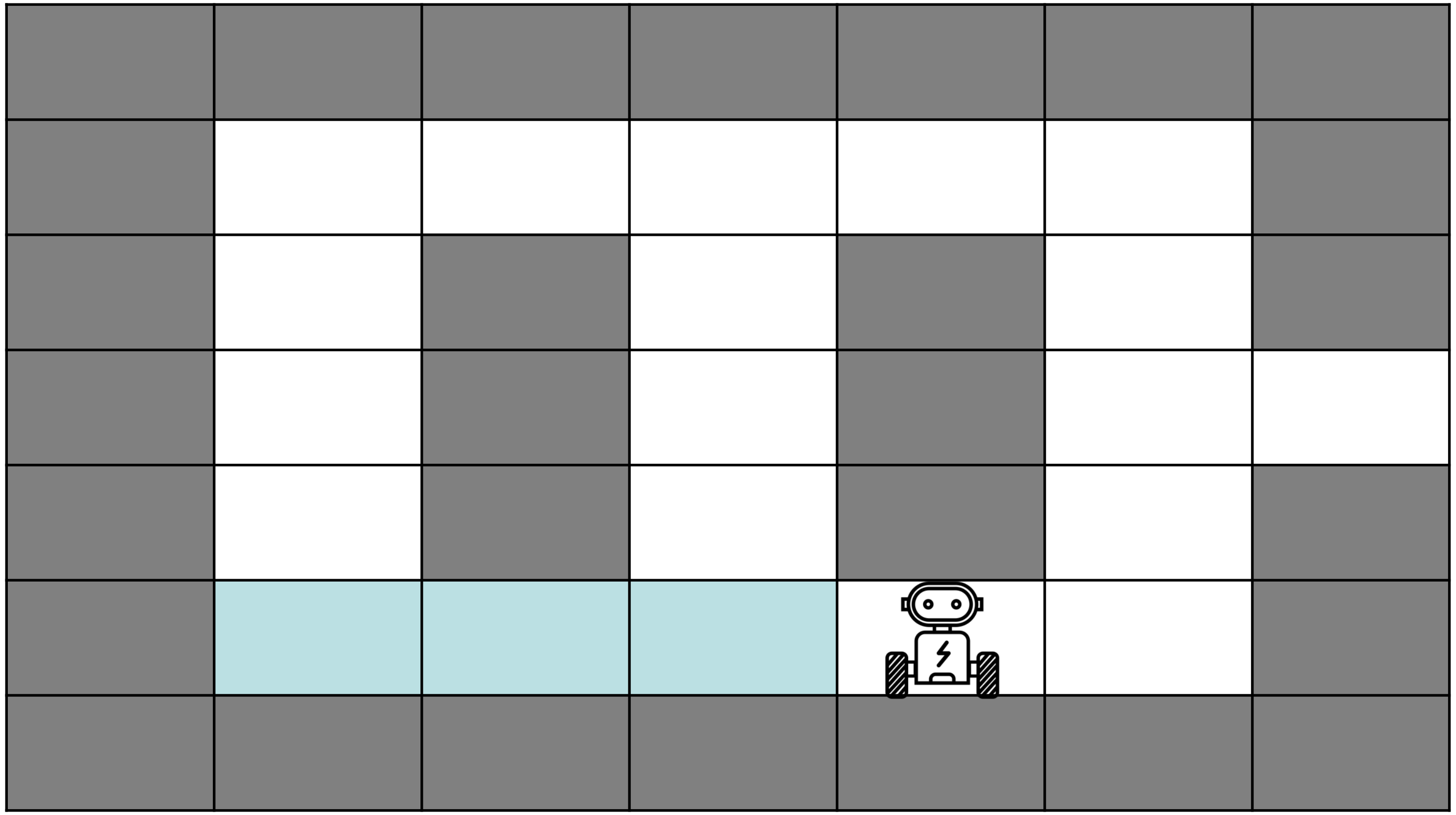
Visited nodes:

[6,2]

[6,3]

[6,4]

Improving the search algorithm



Visited nodes:

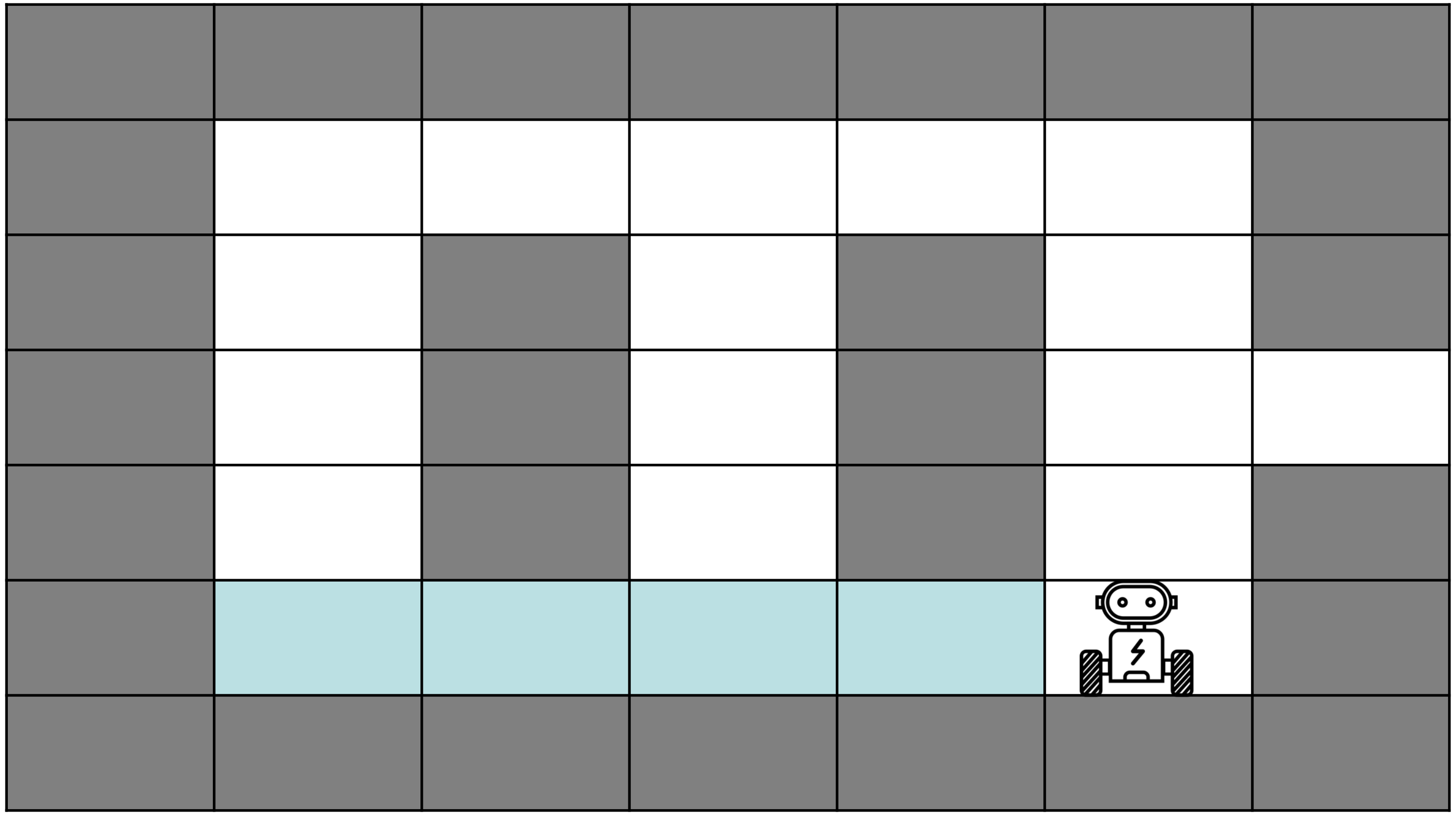
[6,2]

[6,3]

[6,4]

[6,5]

Improving the search algorithm



Visited nodes:

[6,2]

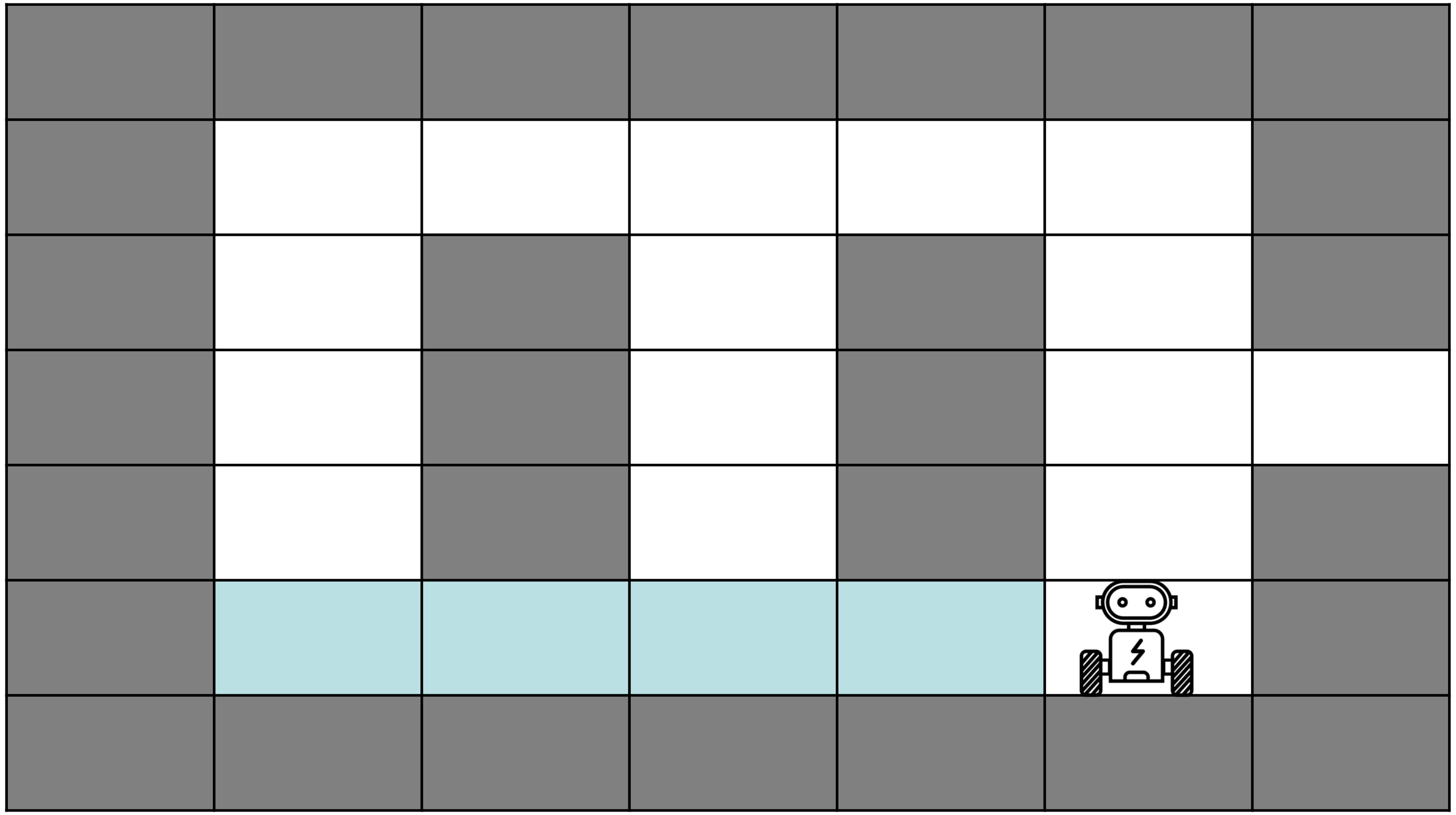
[6,3]

[6,4]

[6,5]

[6,6]

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

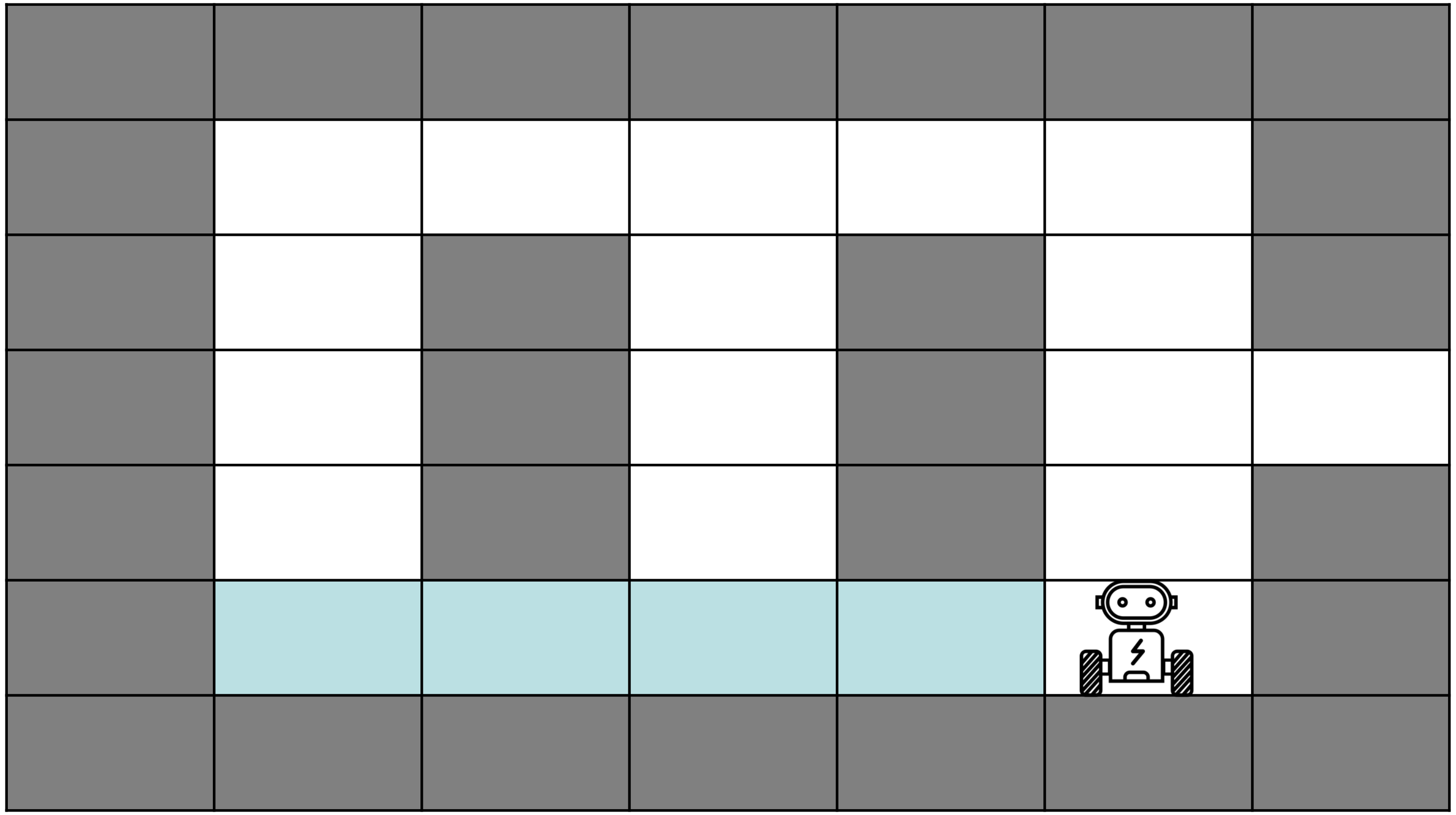
[6,4]

[6,5]

[6,6]

Should we include obstacles in our visited nodes set?

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

[6,4]

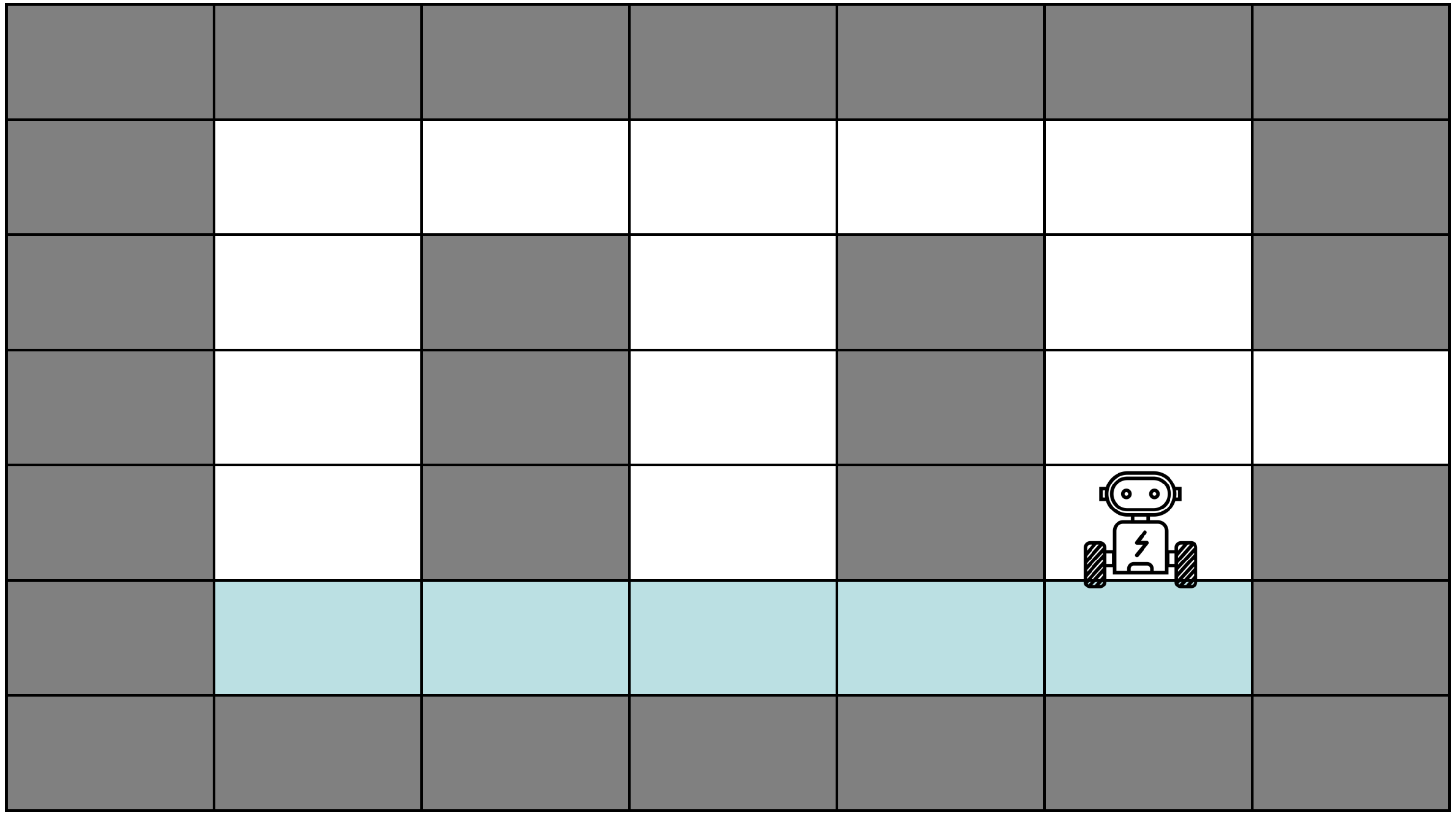
[6,5]

[6,6]

Should we include obstacles in our visited nodes set?

Generally: NO

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

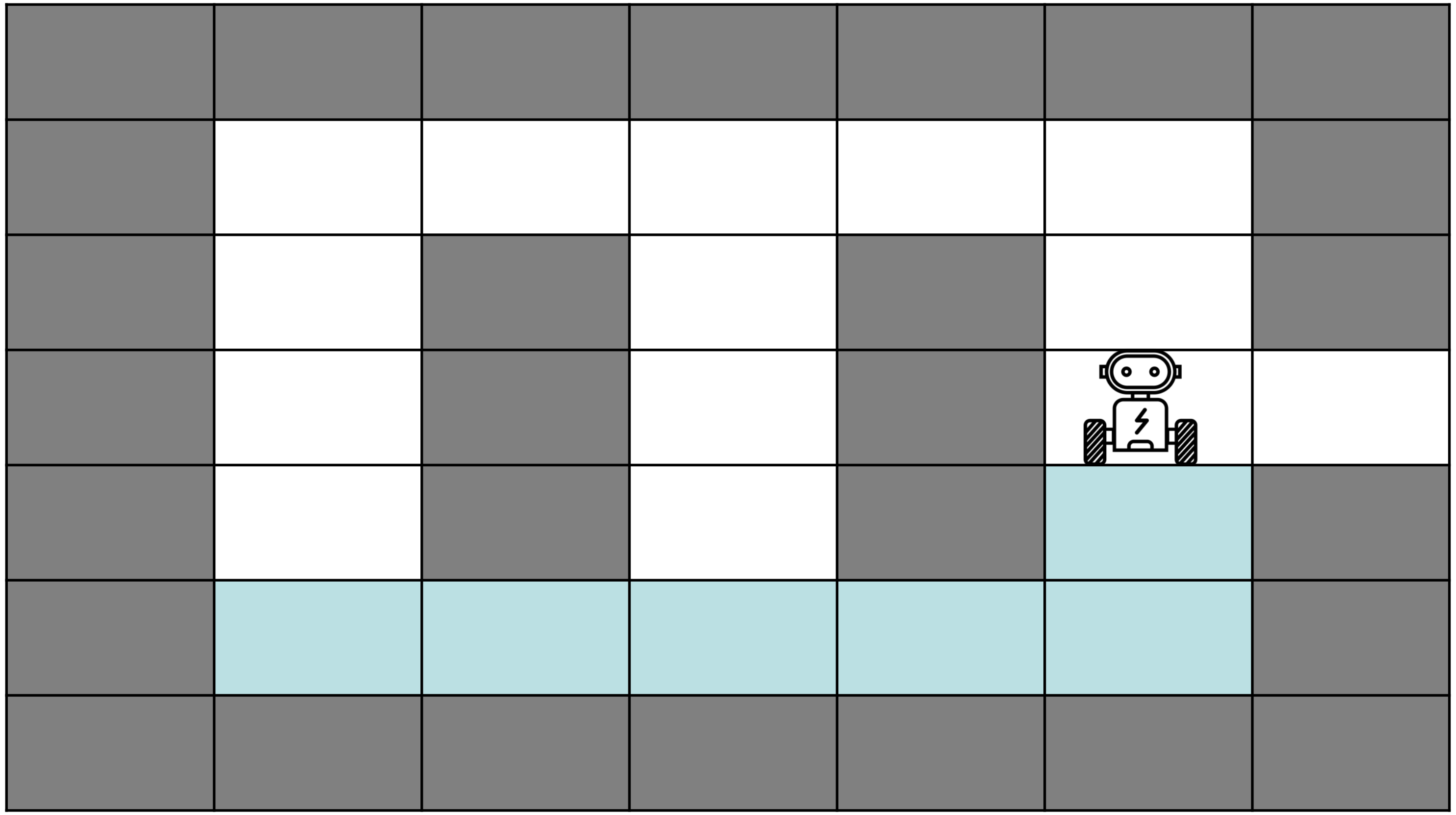
[6,4]

[6,5]

[6,6]

[5,6]

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

[6,4]

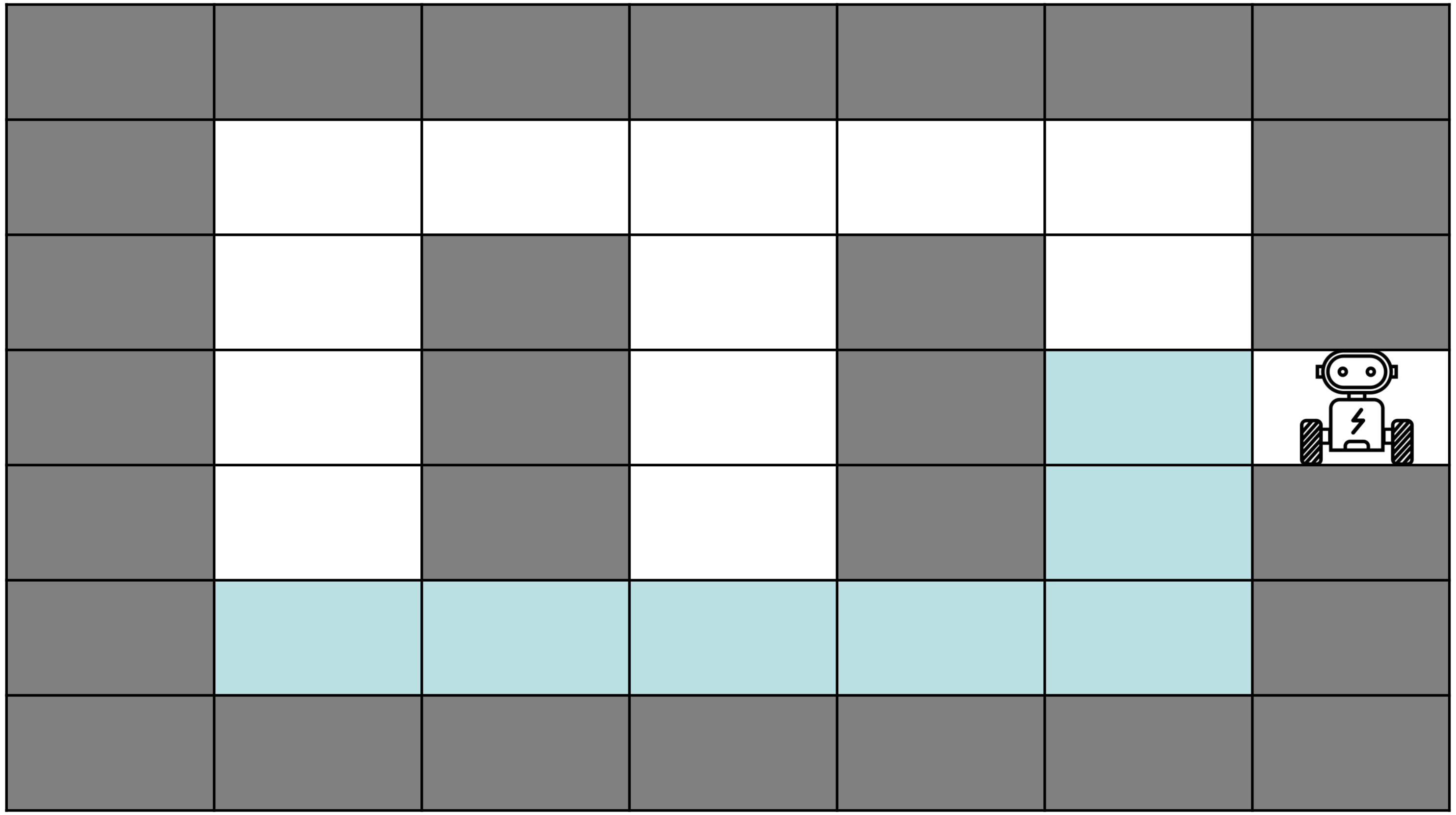
[6,5]

[6,6]

[5,6]

[4,6]

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

[6,4]

[6,5]

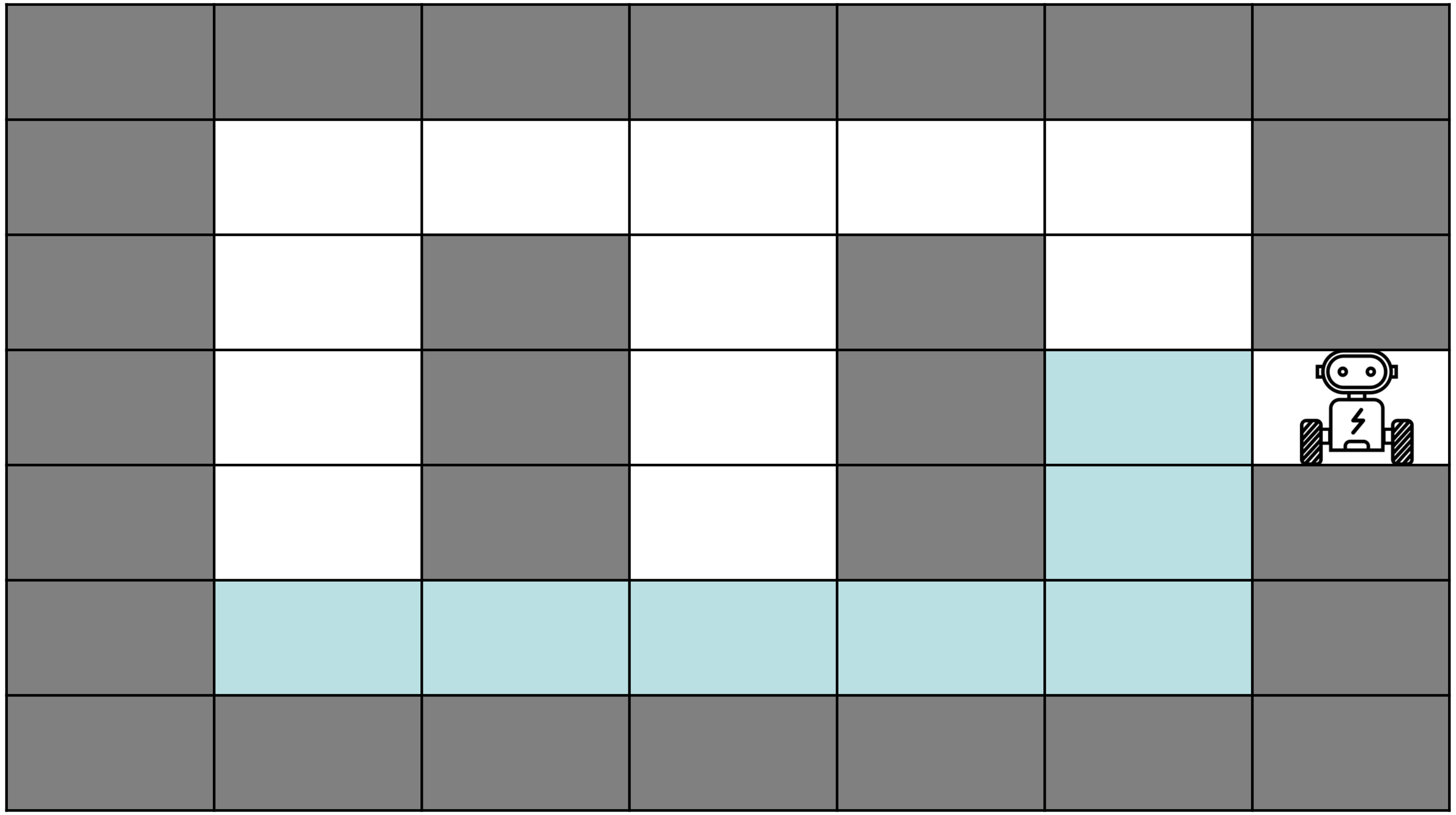
[6,6]

[5,6]

[4,6]

[4,7]

Improving the search algorithm



Visited nodes:

[6,2]

[6,3]

[6,4]

[6,5]

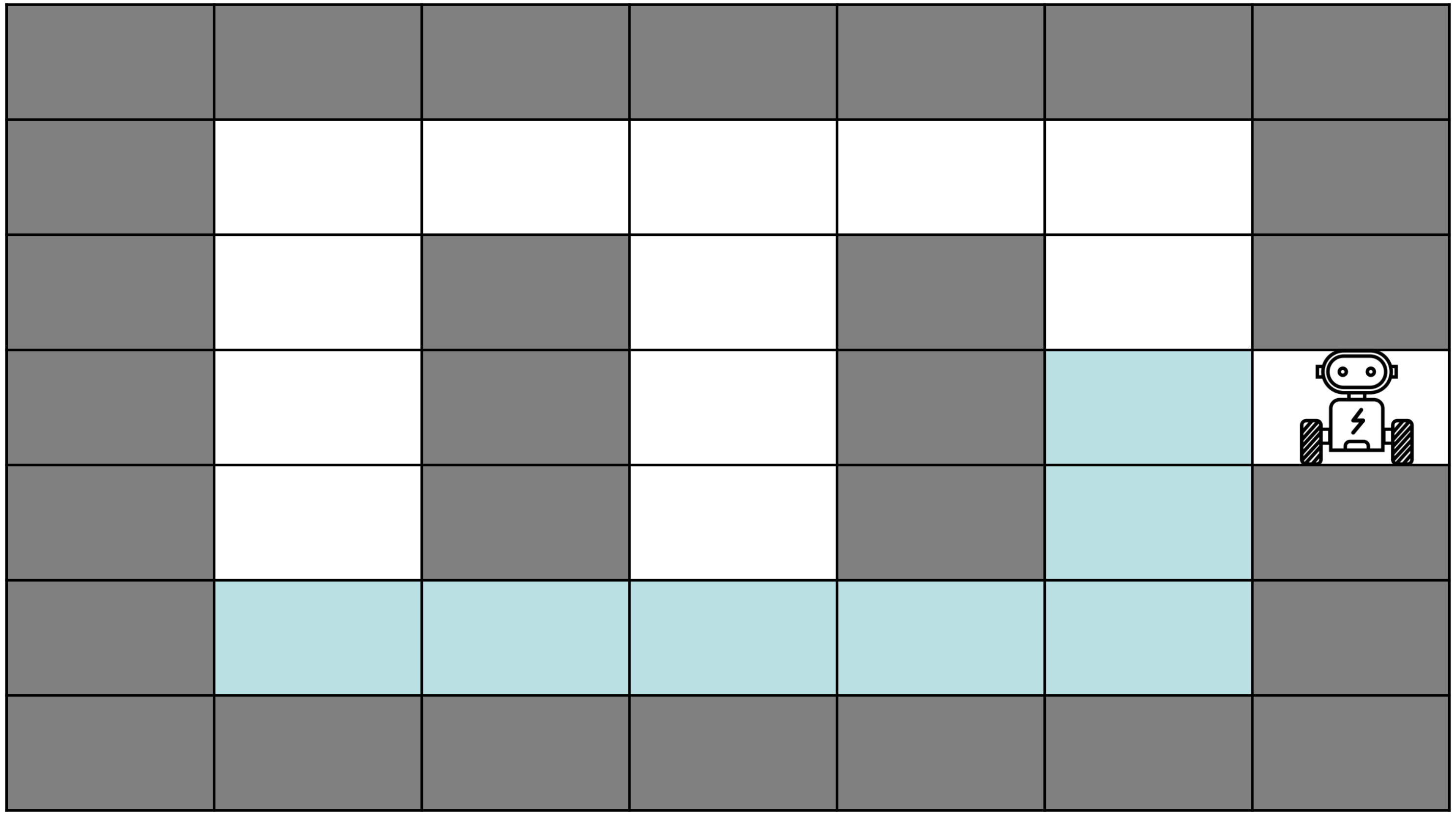
[6,6]

[5,6]

[4,6]

[4,7]

Let’s test!



1. Implement the new algorithm in MATLAB
2. Compare how better it was with the random-direction choice

From exercises session

- Why we need to check which algorithms is best?

From exercises session

Why we need to check which algorithms is best?

- In robotics, we need always to optimize resources

- We need to have measures to take decisions

- It is important to keep the footprint

Three ideas for algorithms’ performance

Three ideas for algorithms' performance

- Time
- Operations
- Lines of code
- Accuracy

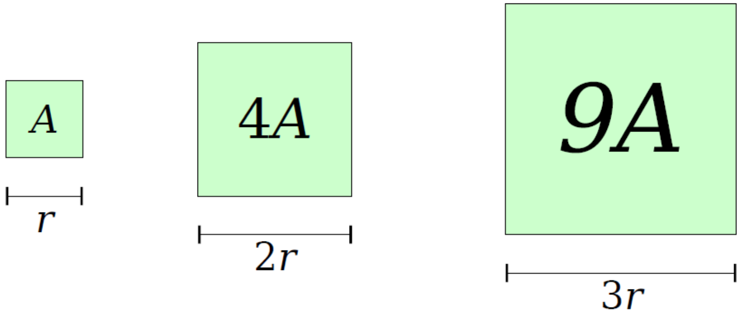
Why they are not enough?

- Language dependency
- Hardware dependency
- …

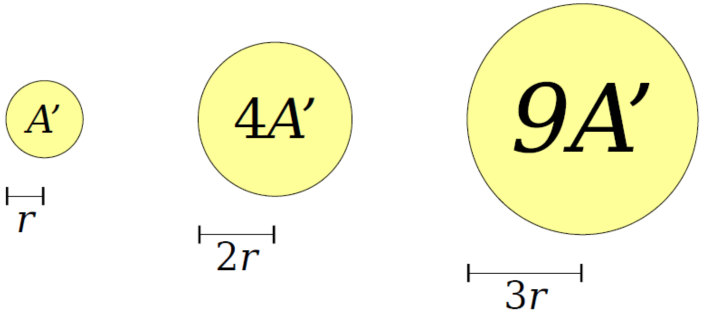
What about scalability?

O(n) – Big O of n

- ***Big-O notation*** is a way of quantifying the rate at which some quantity grows.
- For example:
  - A square of side length r has area O(r^2).
  - A circle of radius r has area O(r^2).

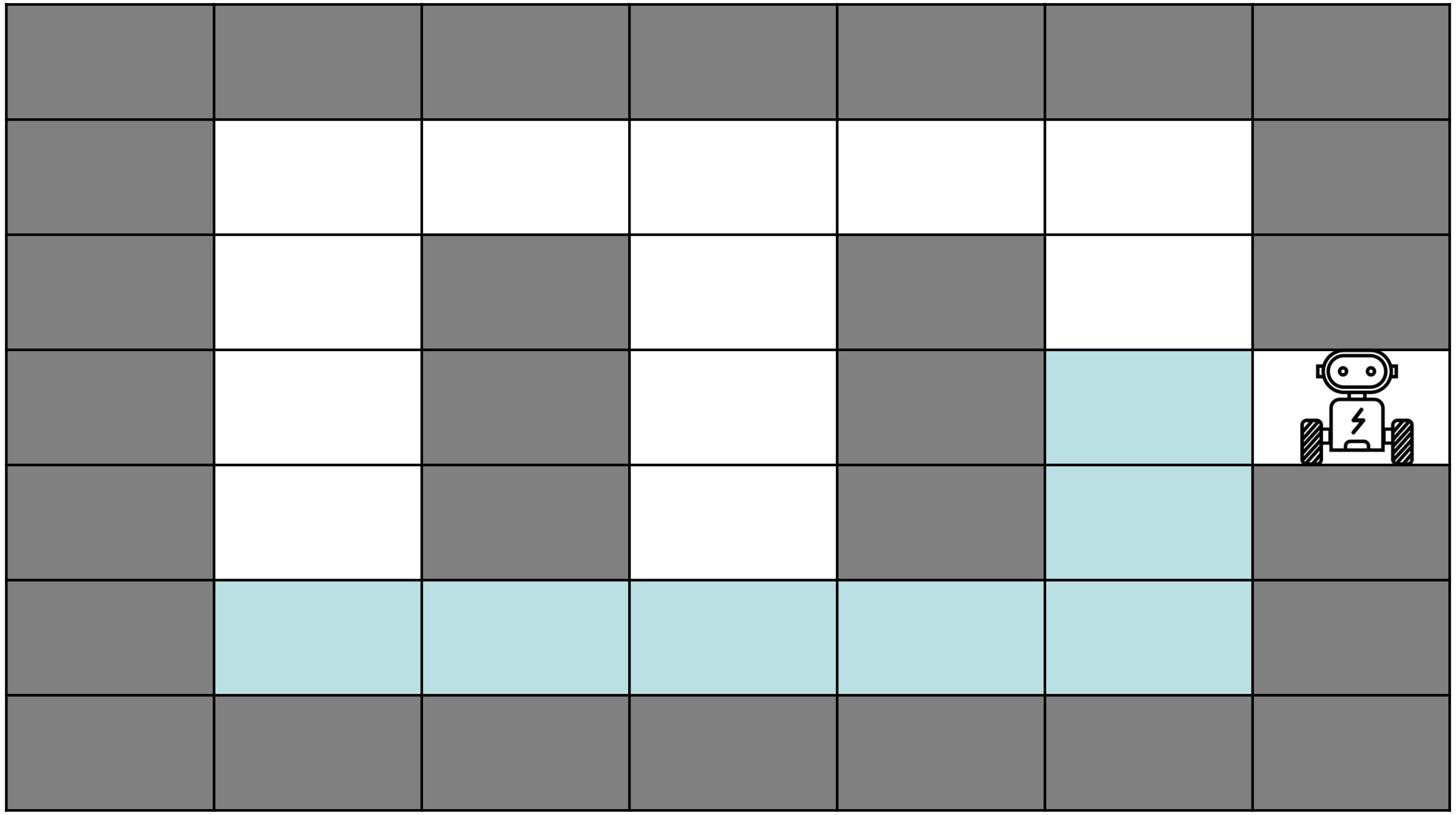


Doubling r increases area 4×.
Tripling r increases area 9×.

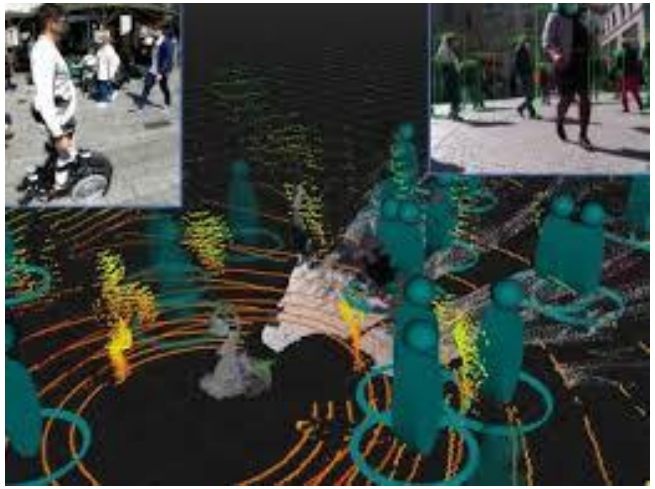


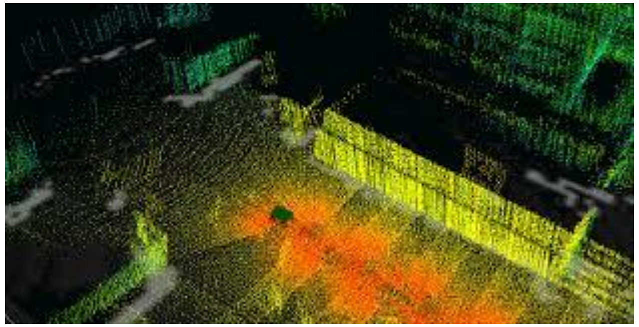
Doubling r increases area 4×.
Tripling r increases area 9×.

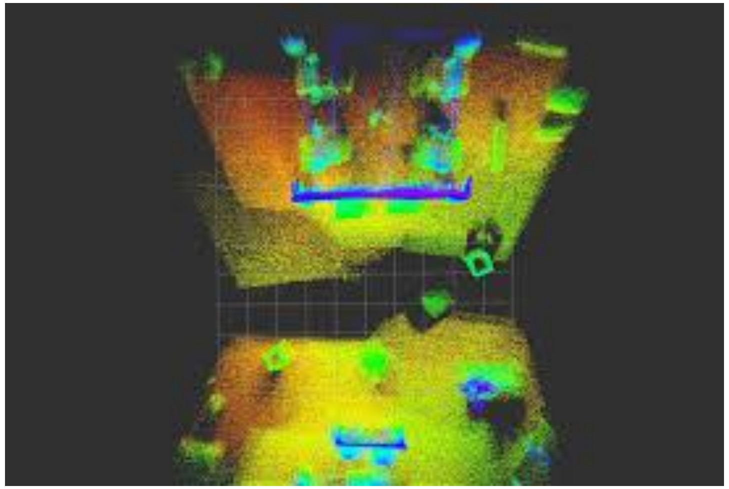
Why does that matter?

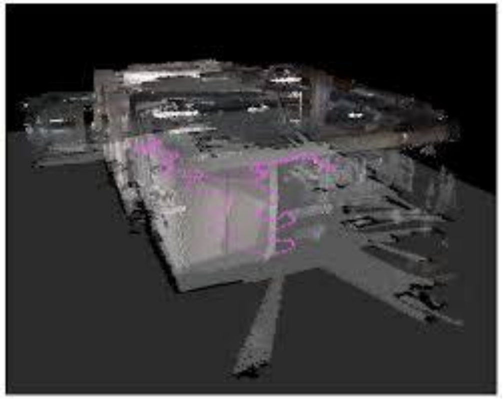


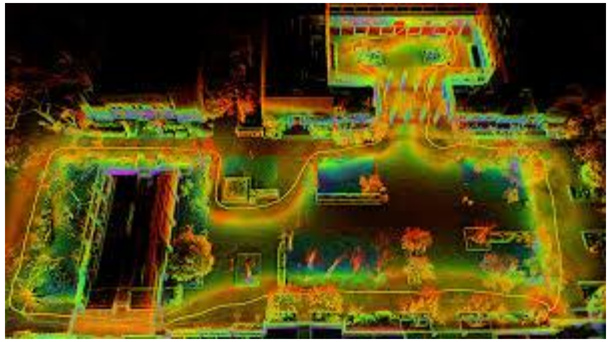
Why does that matter?



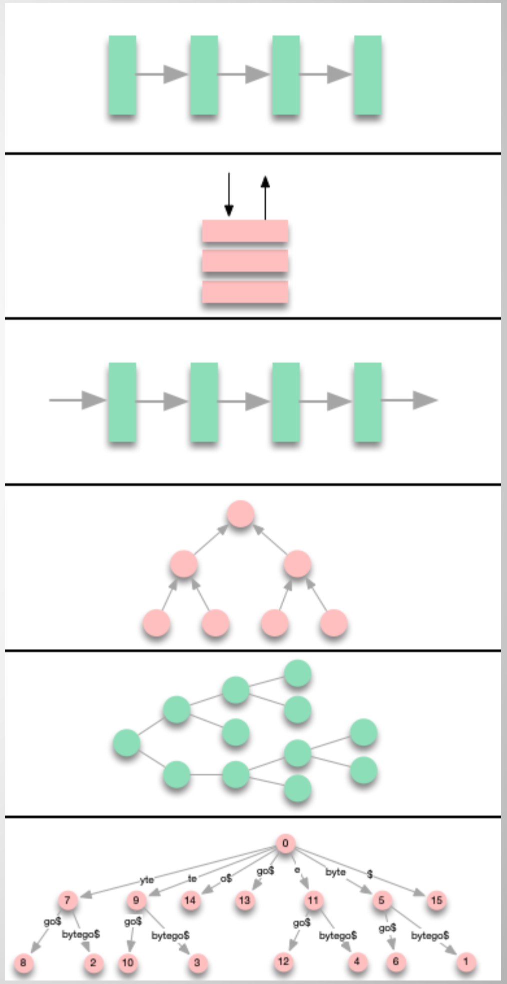








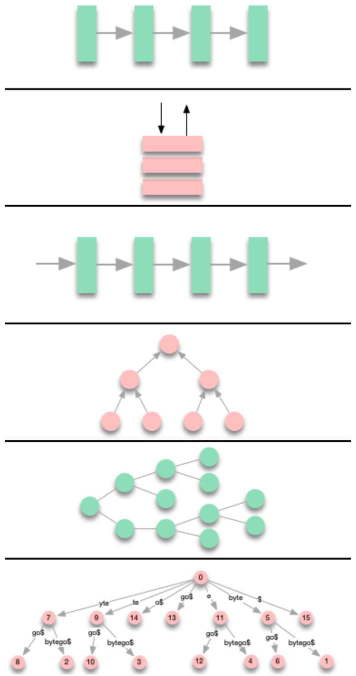
Data Structures



Some definitions

Data Structure

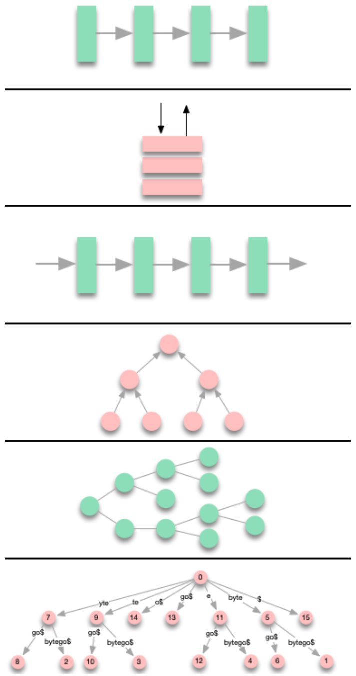
How information is organised



Some definitions

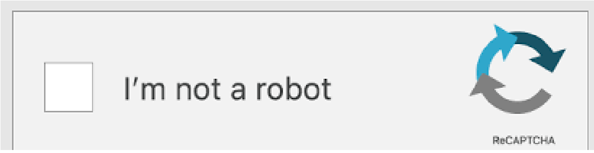
Data Structure

How information is organised



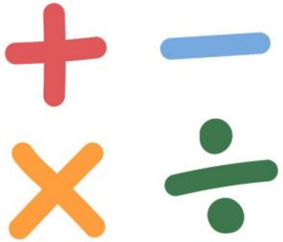
Operations

- Queries (just checks)



□ I’m not a robot

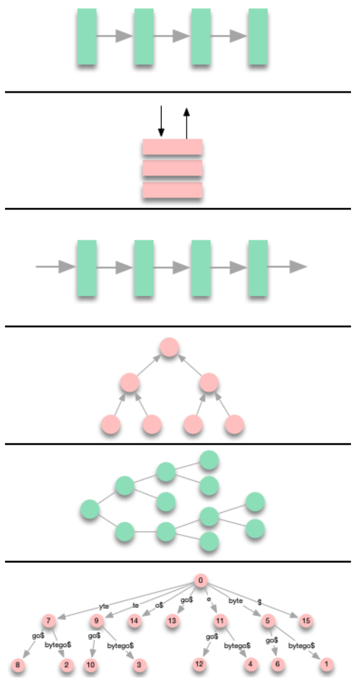
- Modifying operations



Some definitions

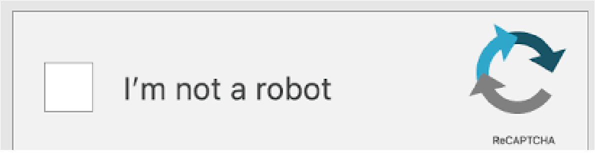
Data Structure

How information is organised

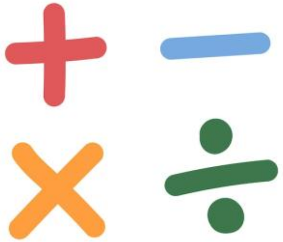


Operations

- Queries (just checks)



- Modifying operations



Algorithms

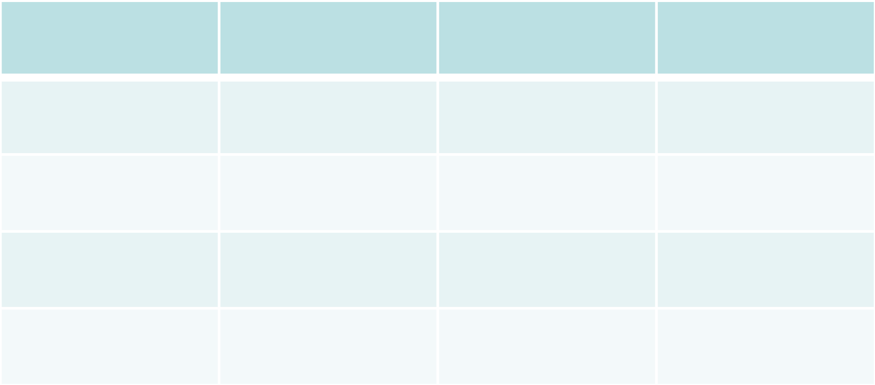
Methods for keeping efficiency in data handling

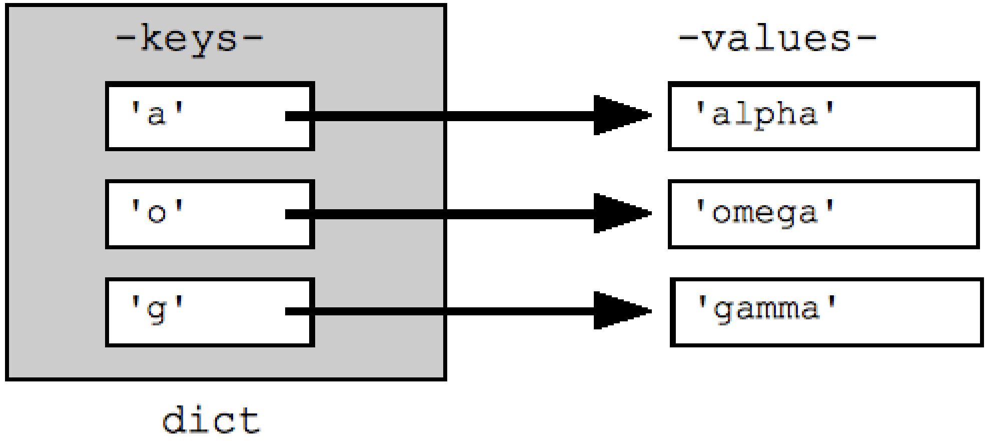
Data Structures

- ?

Data Structures

- Arrays
- Matrices
- Structs (in C)
- Objects (in python), like dictionary..
- …





Static structures!

Static structures

Let’s analyse our visited nodes list for the simple operations:

- Search
- Insert
- Delete

0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1

Static structures

Let’s analyse our visited nodes list for other operations:

- MAX/MIN
- Half…

0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1

Static structures

How about having the array sorted?

0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1
1	4	26	48	51		54	67

Static structures – PROBLEMS?

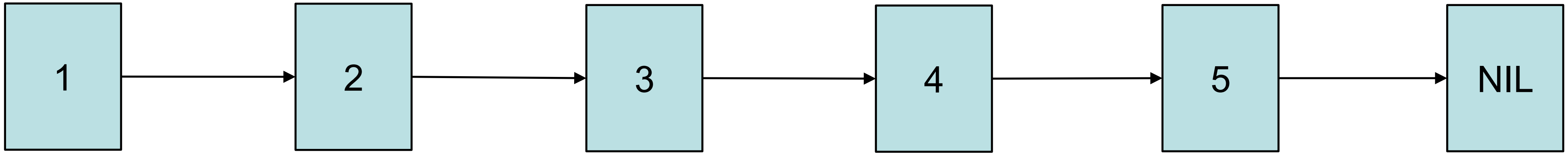
0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1

Linked list

Dynamic structures

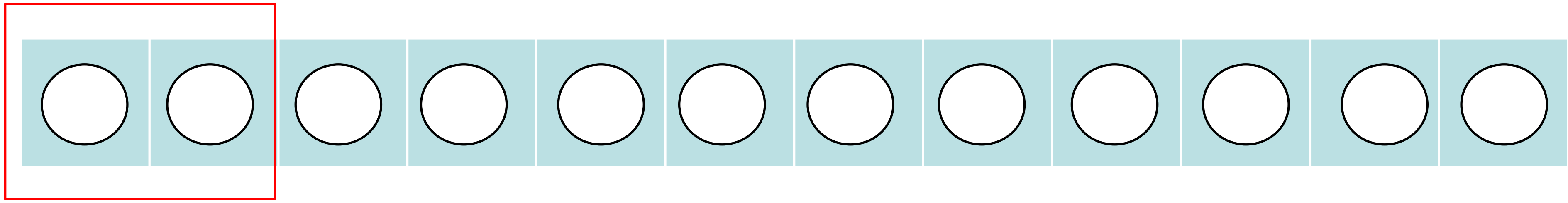
0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1

0	1	2	3	4	….	n-1	n
67	54	51	48	26		4	1



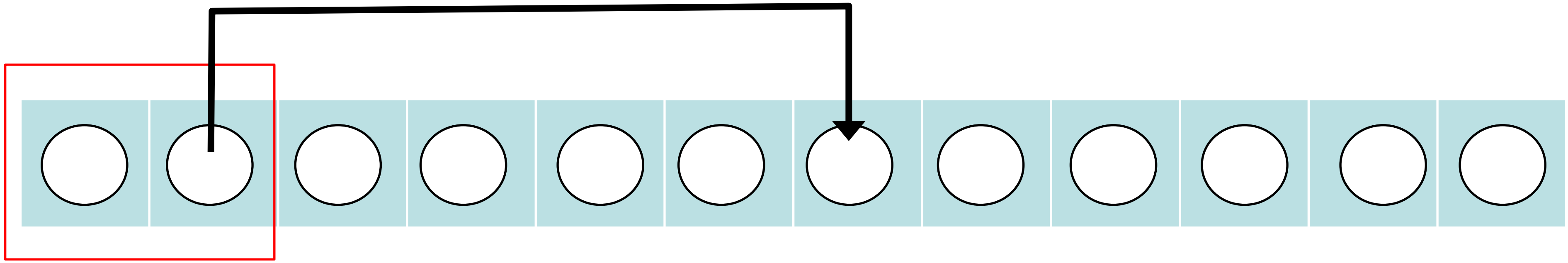
Dynamic Structures

- Using memory space to **point to** the next element



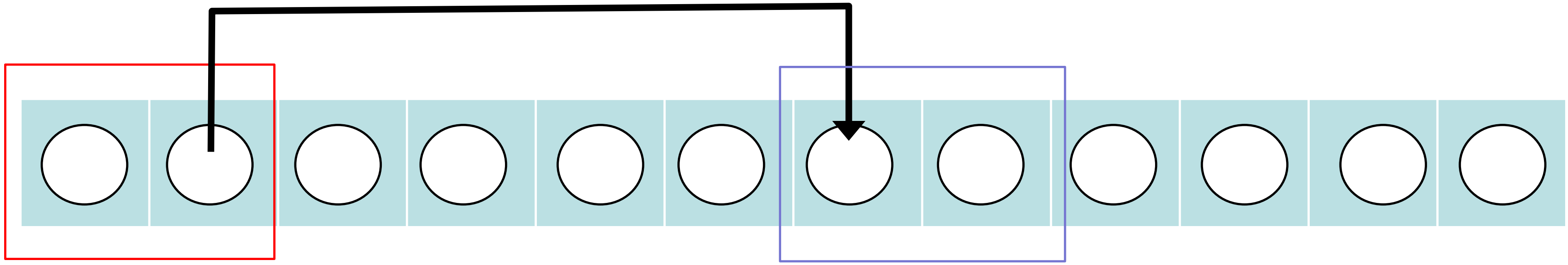
Dynamic Structures

- Using memory space to **point to** the next element



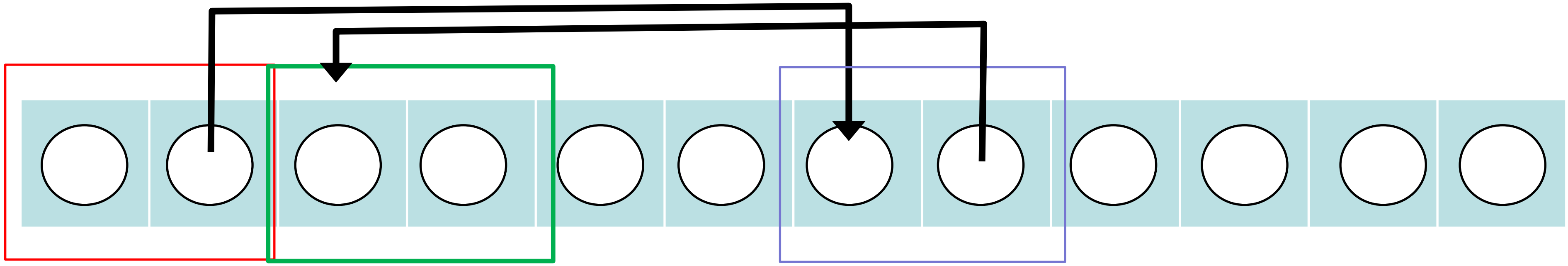
Dynamic Structures

- Using memory space to **point to** the next element



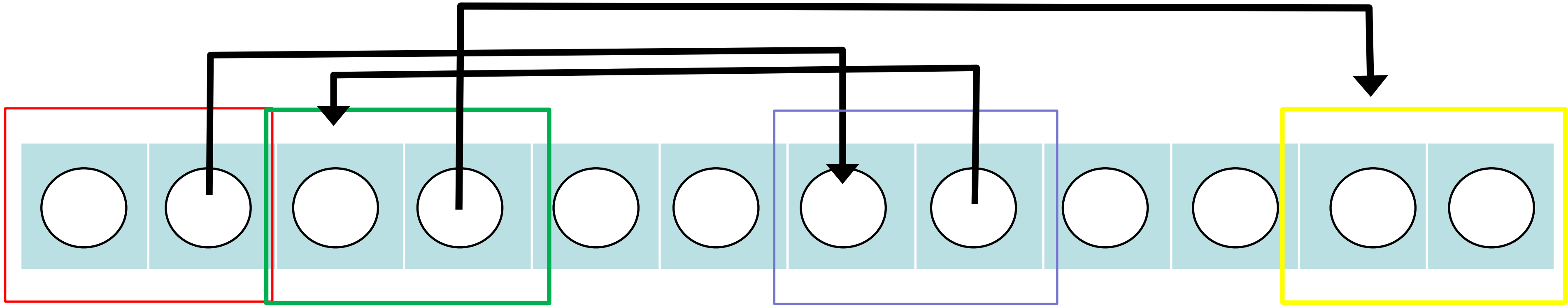
Dynamic Structures

- Using memory space to **point to** the next element



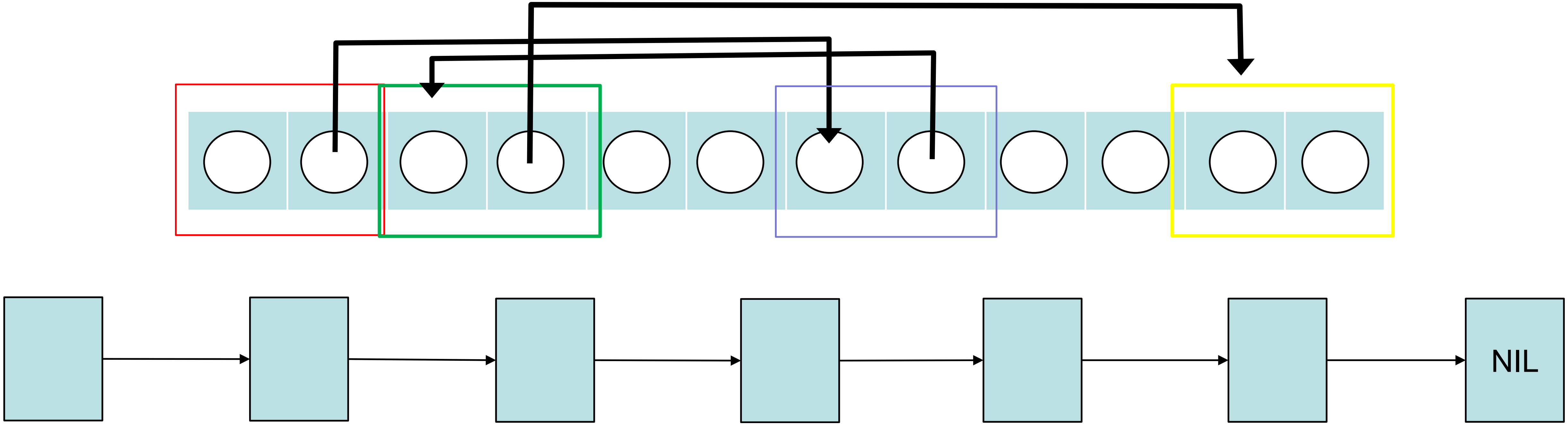
Dynamic Structures

- Using memory space to **point to** the next element

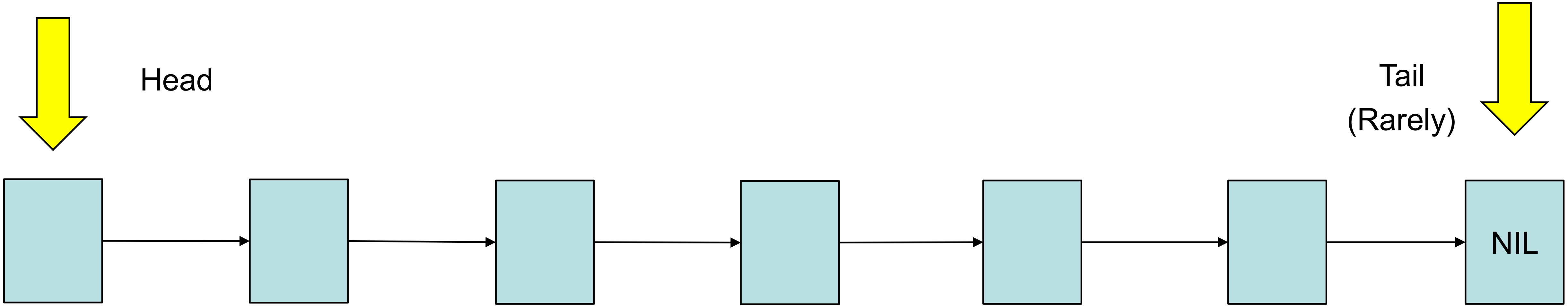


Dynamic Structures

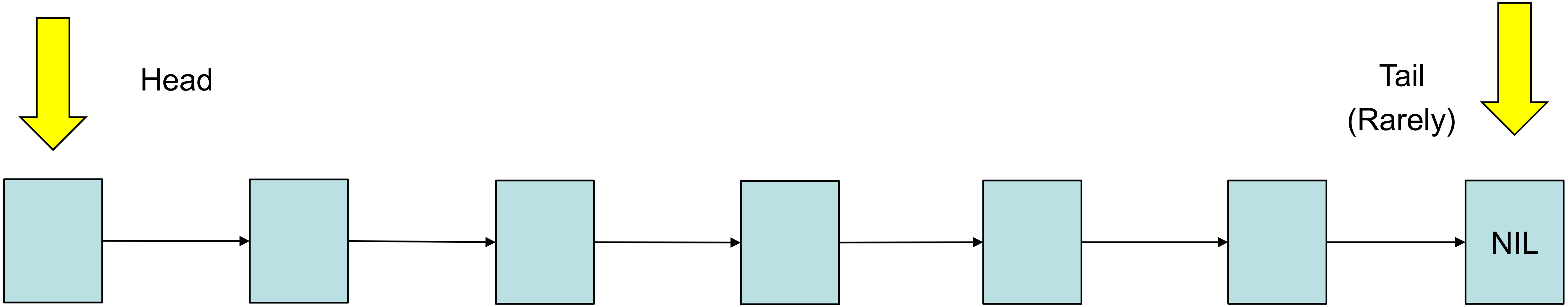
- Using memory space to **point to** the next element



Linked list - Implementation



Linked list - Θ(n)

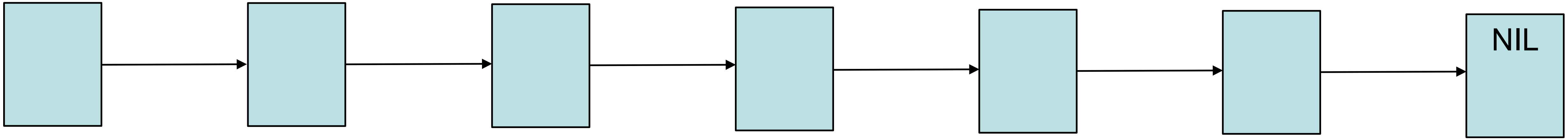


Θ(n) of Insertion?

Θ(n) accessing?

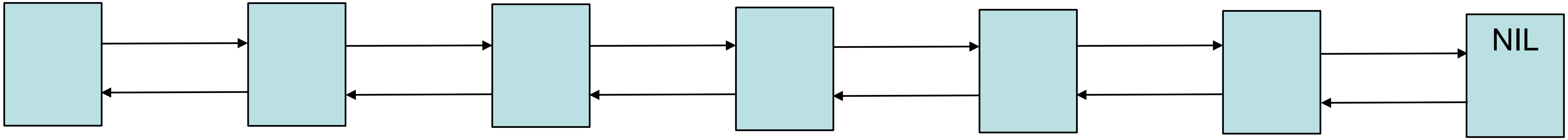
Θ(n) Max/Min?

How to solve accessing issues in lists?

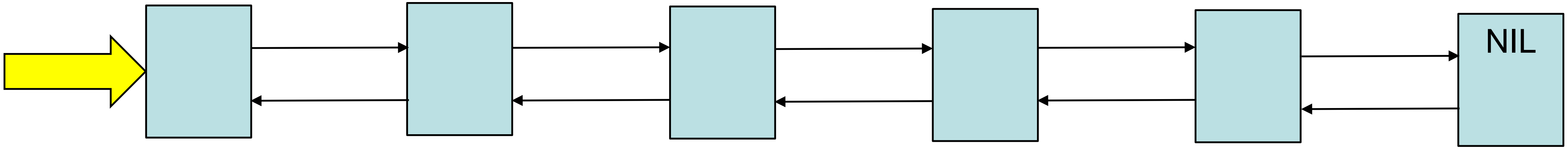


Bring me 3 ideas !

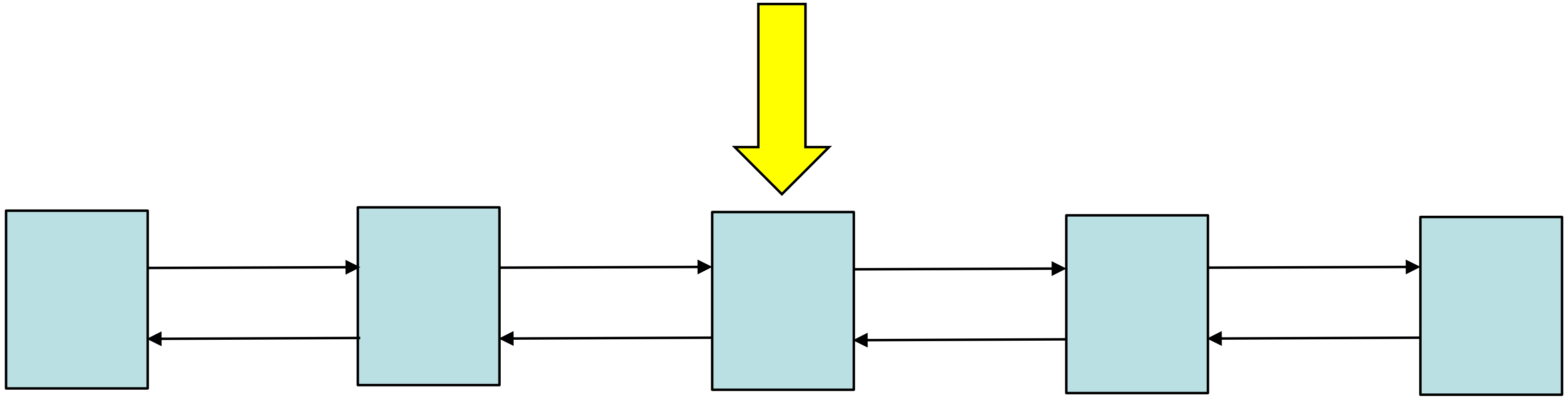
How to solve accessing issues in lists?



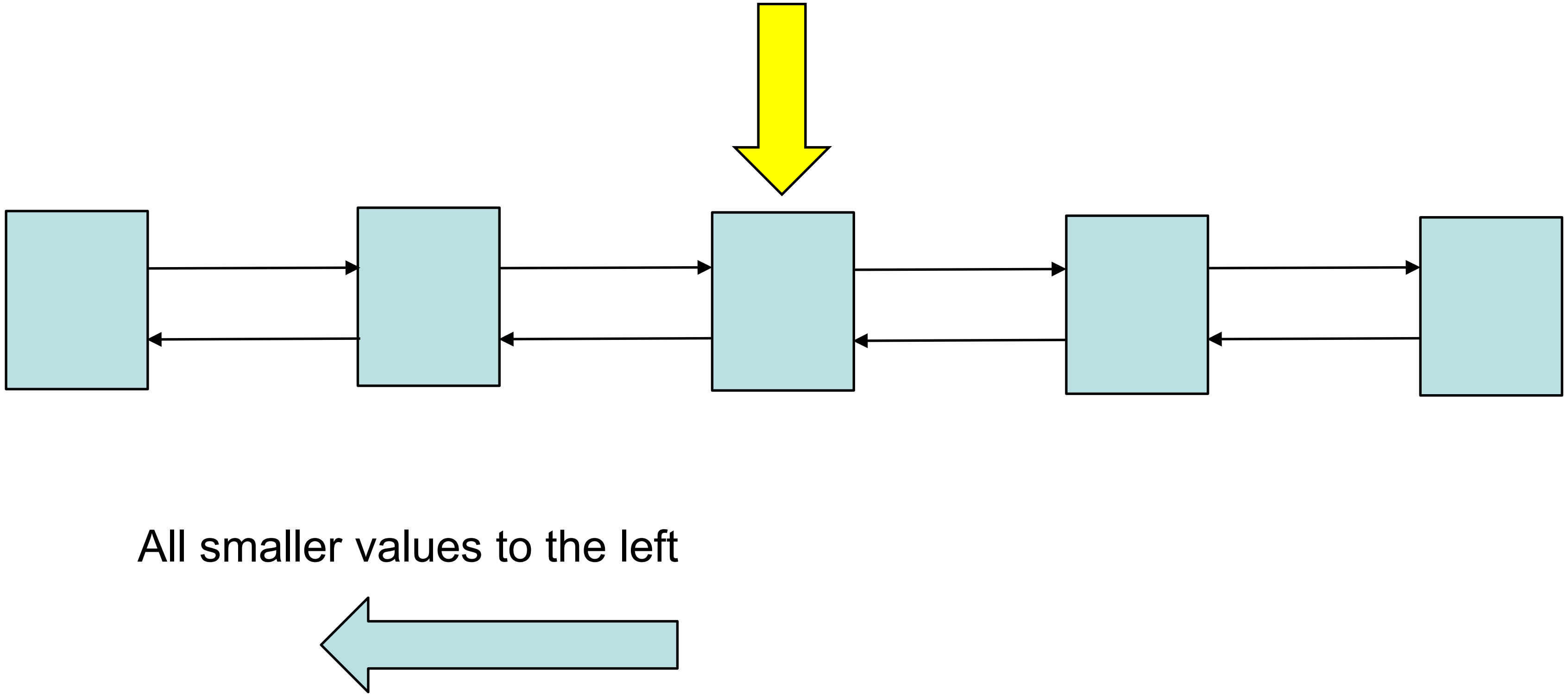
Trees



Trees

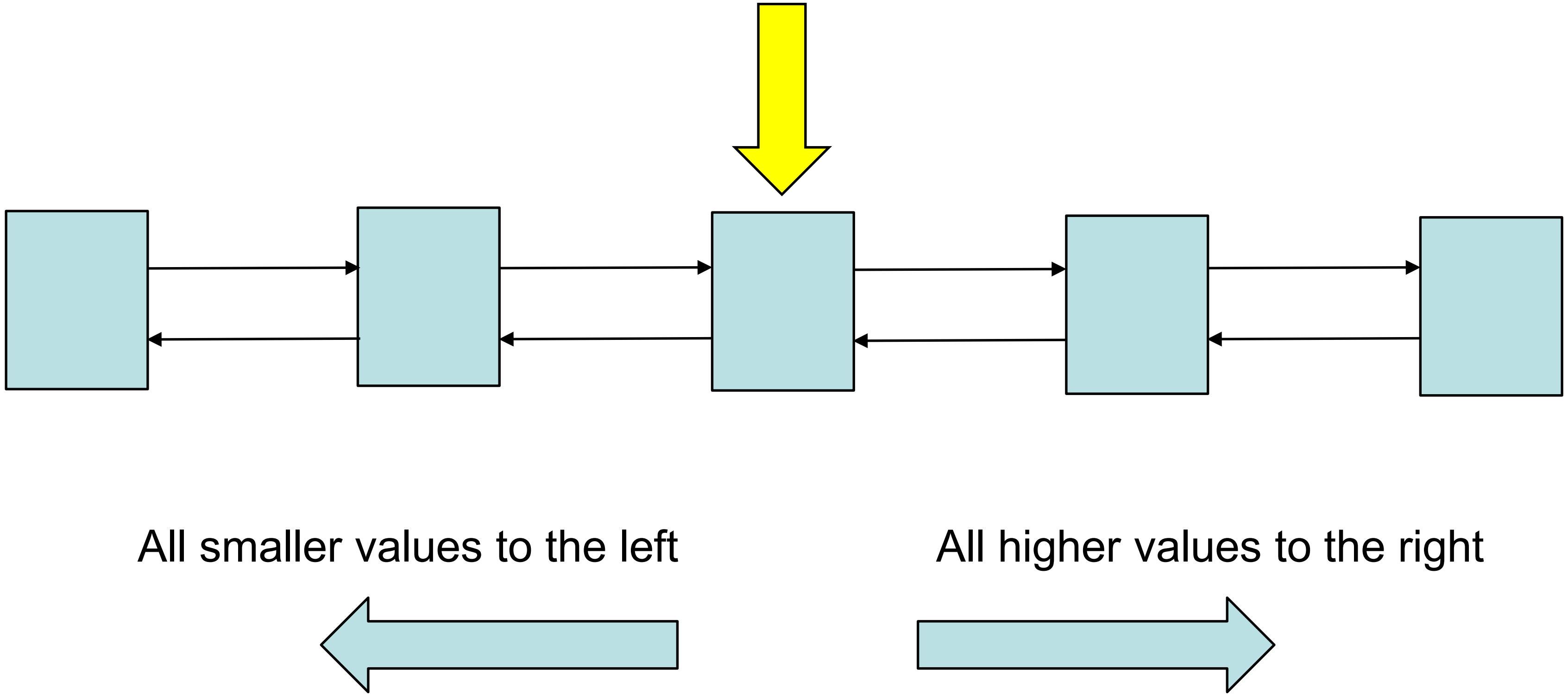


Trees



All smaller values to the left

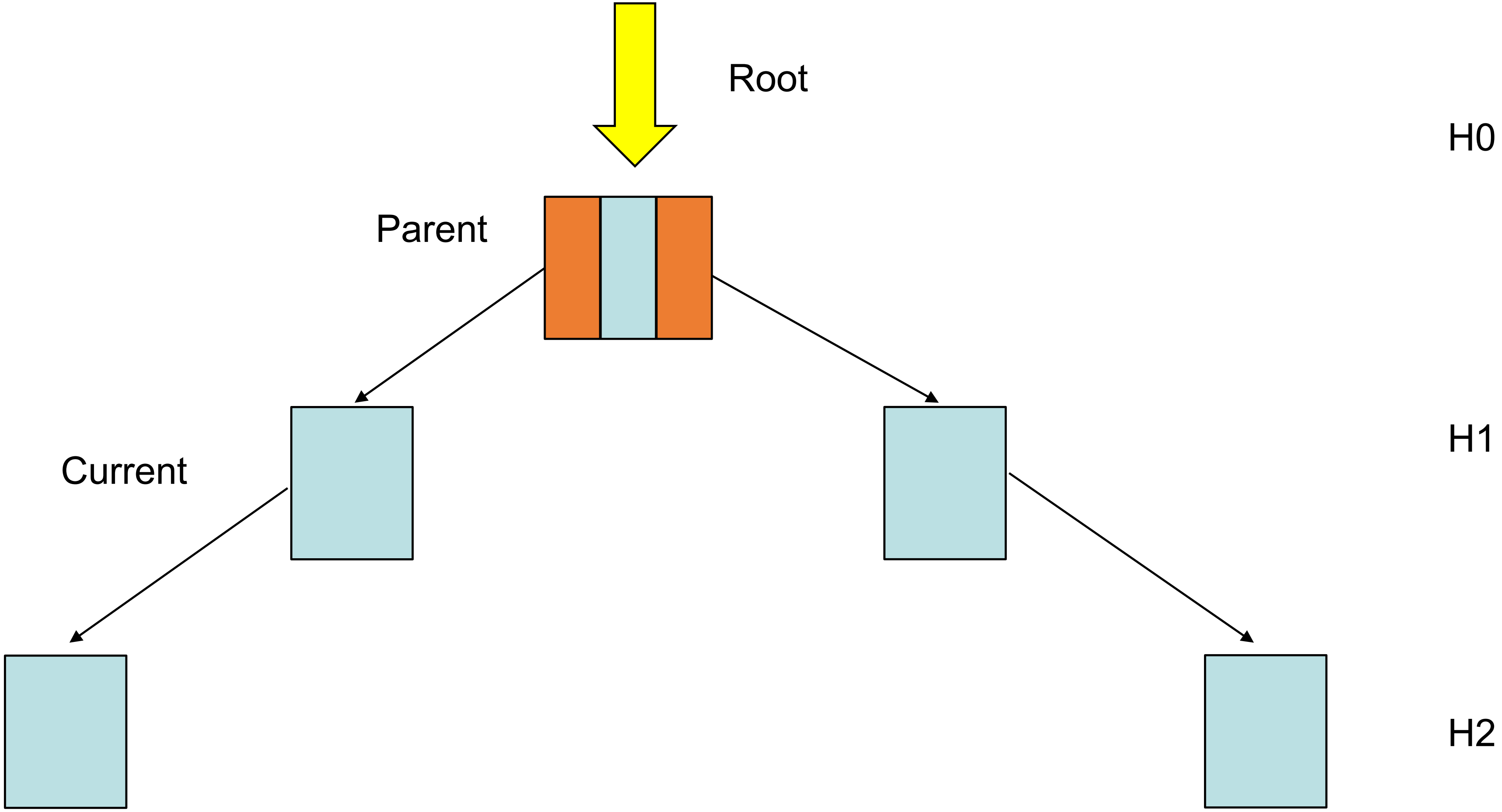
Trees



All smaller values to the left

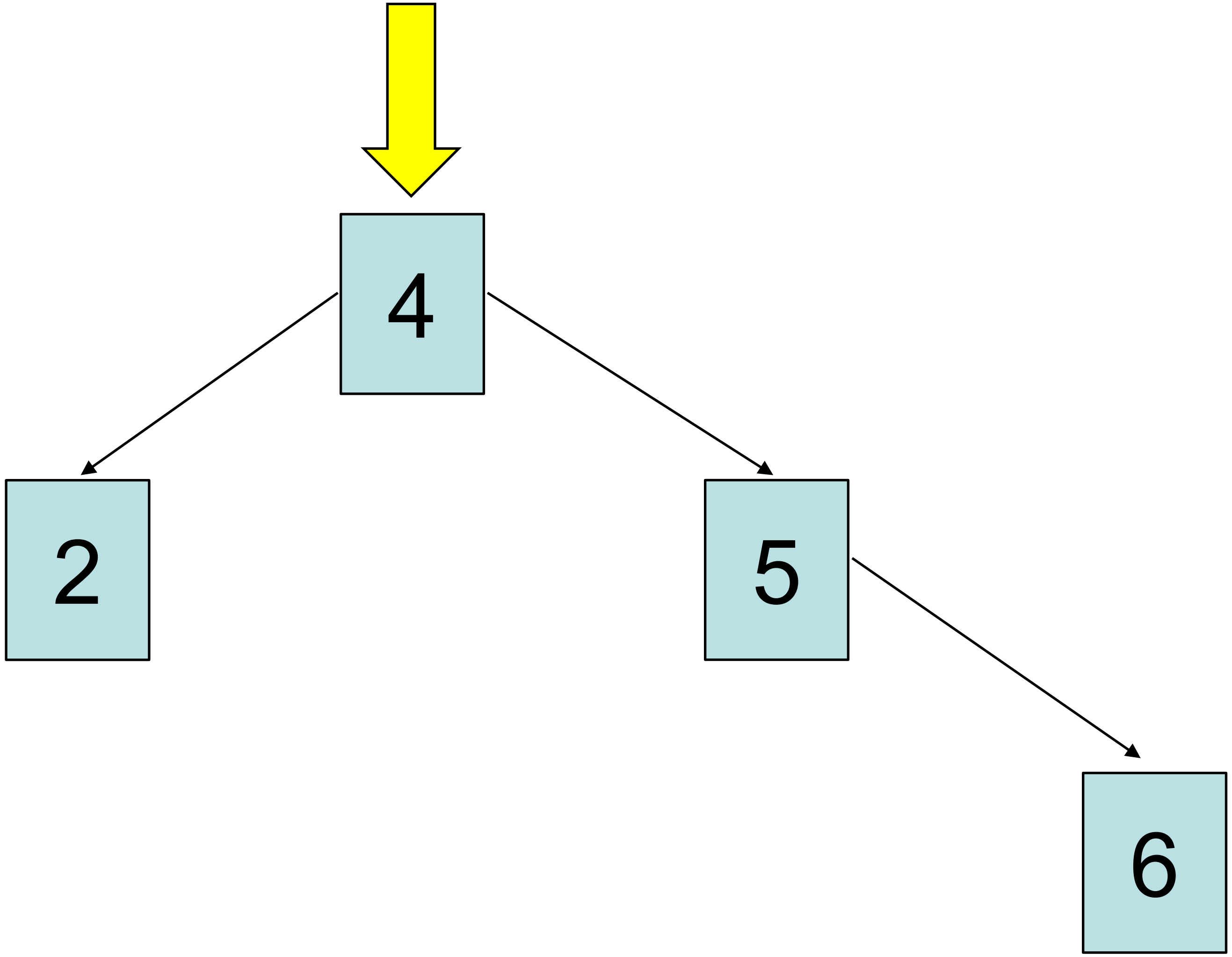
All higher values to the right

Tree



Build this tree

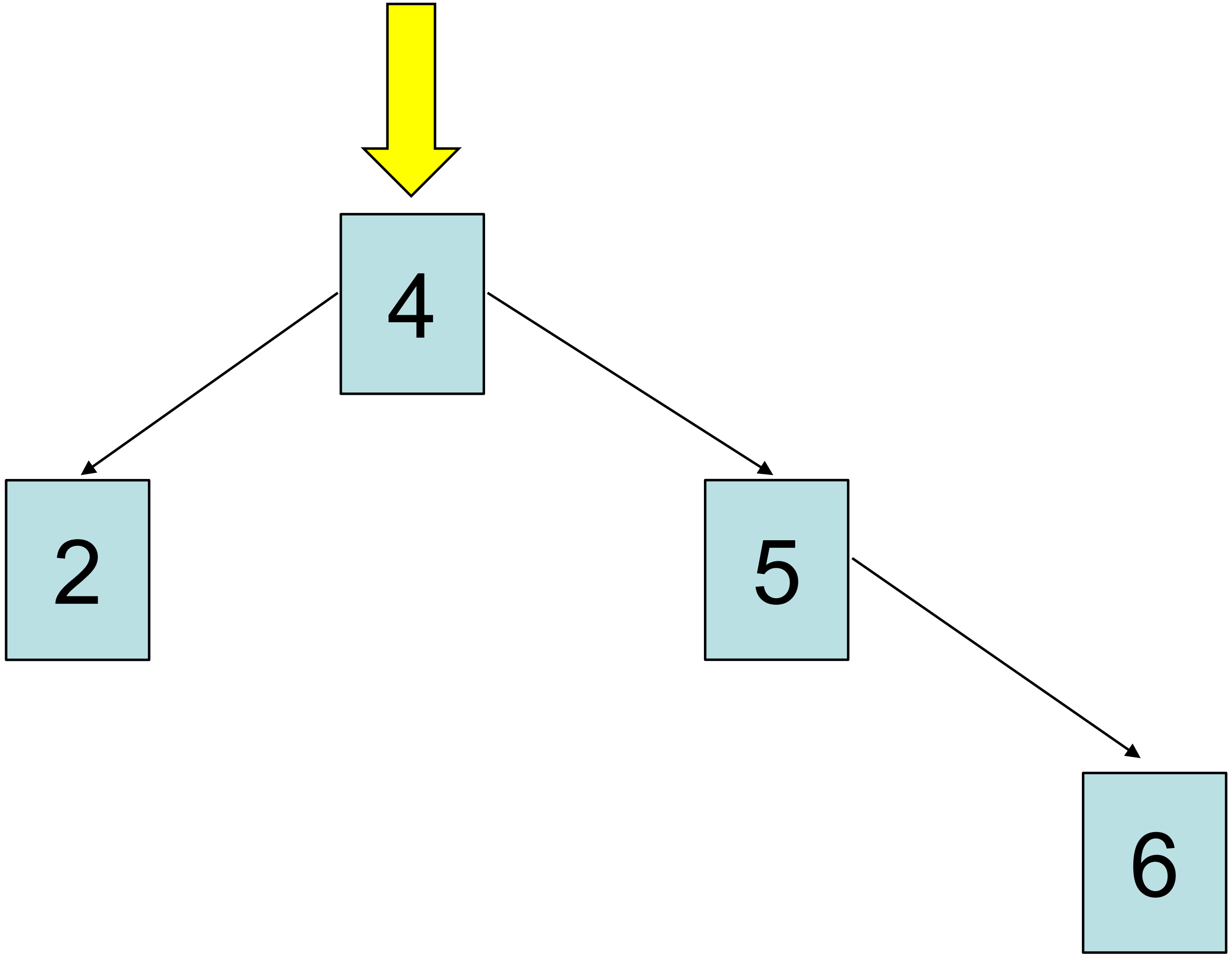
- Insert "1"



Build this tree

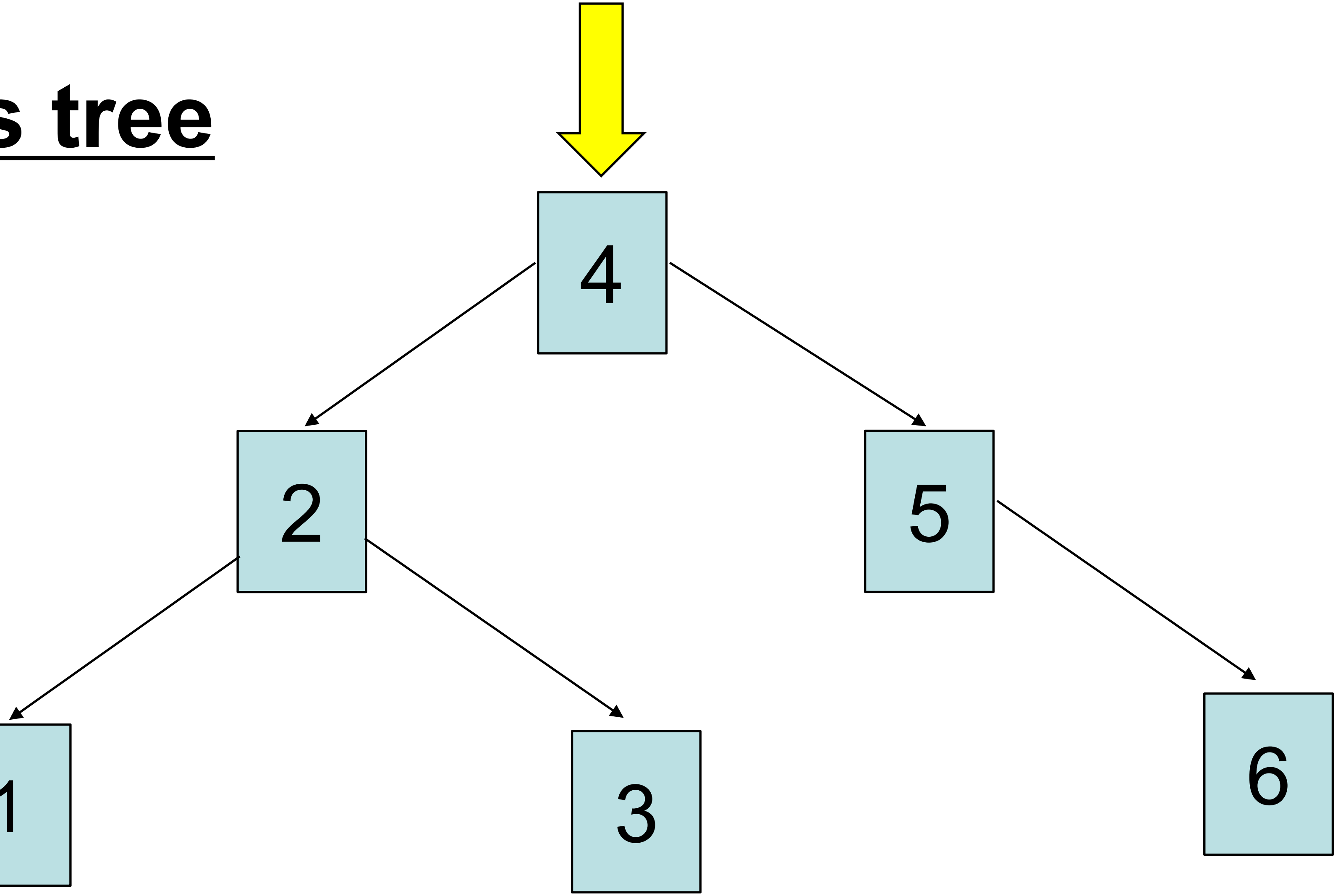
- Insert "1"

- Insert "3"



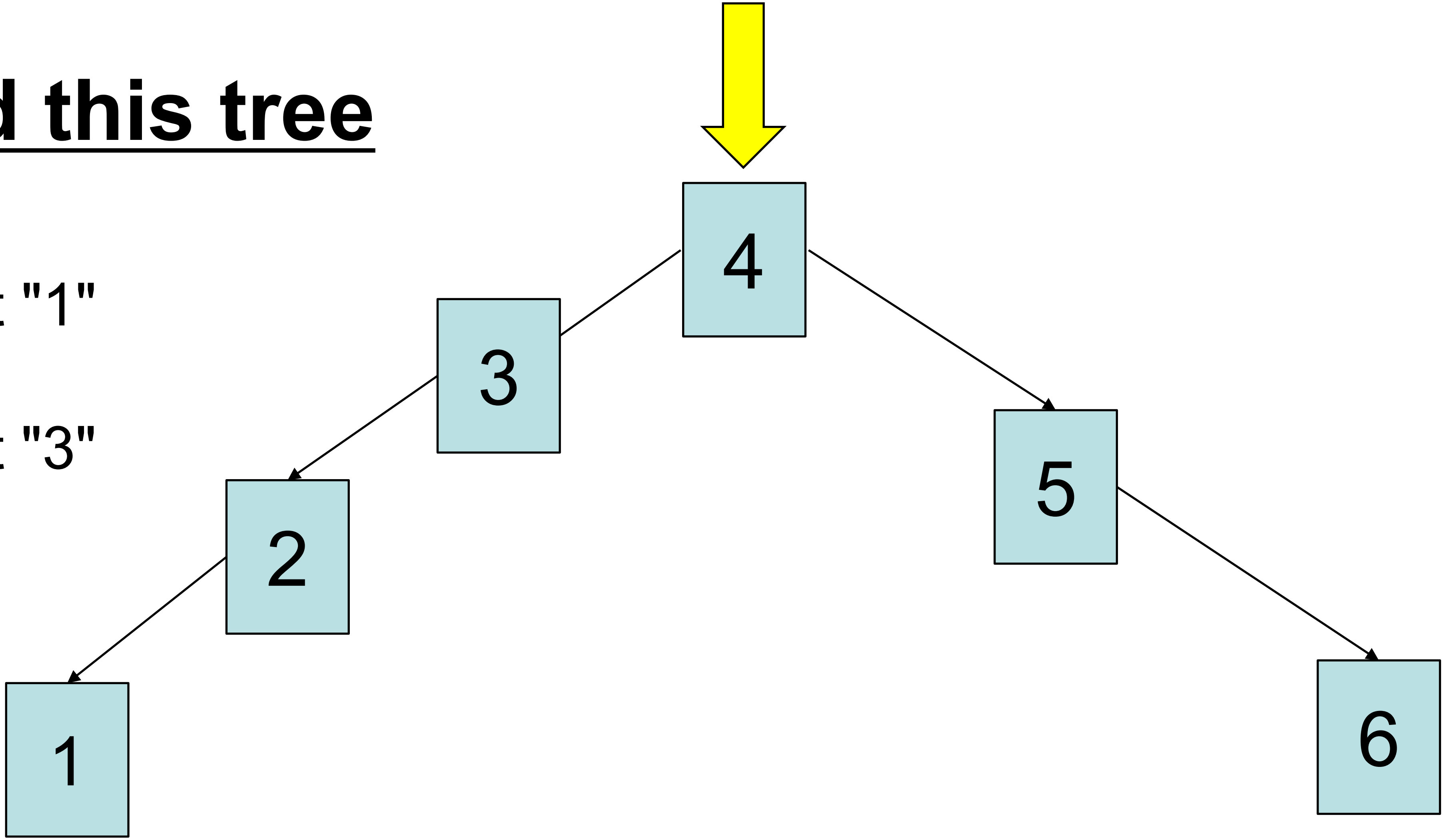
Build this tree

- Insert "1"
- Insert "3"

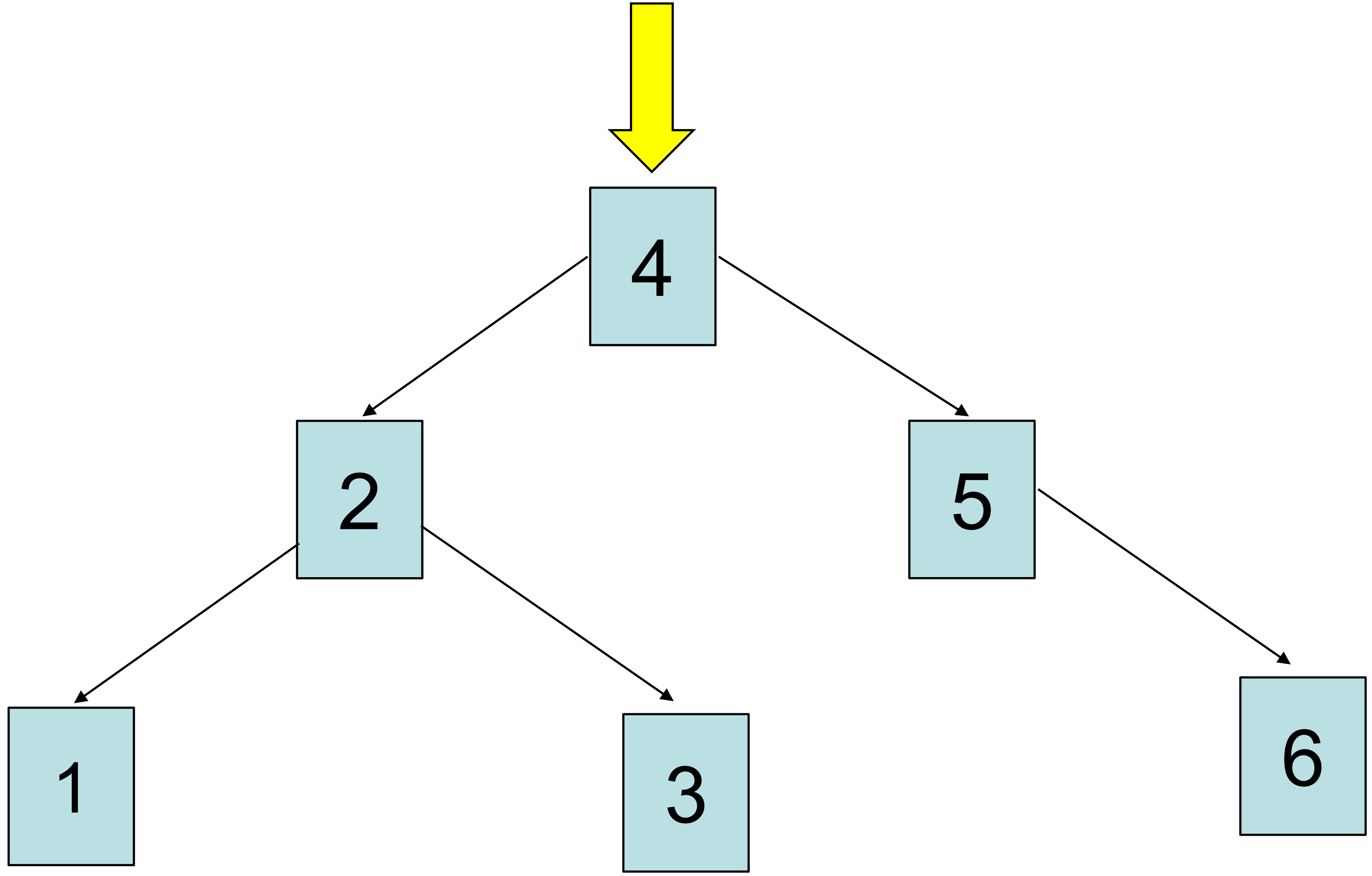


Build this tree

- Insert "1"
- Insert "3"

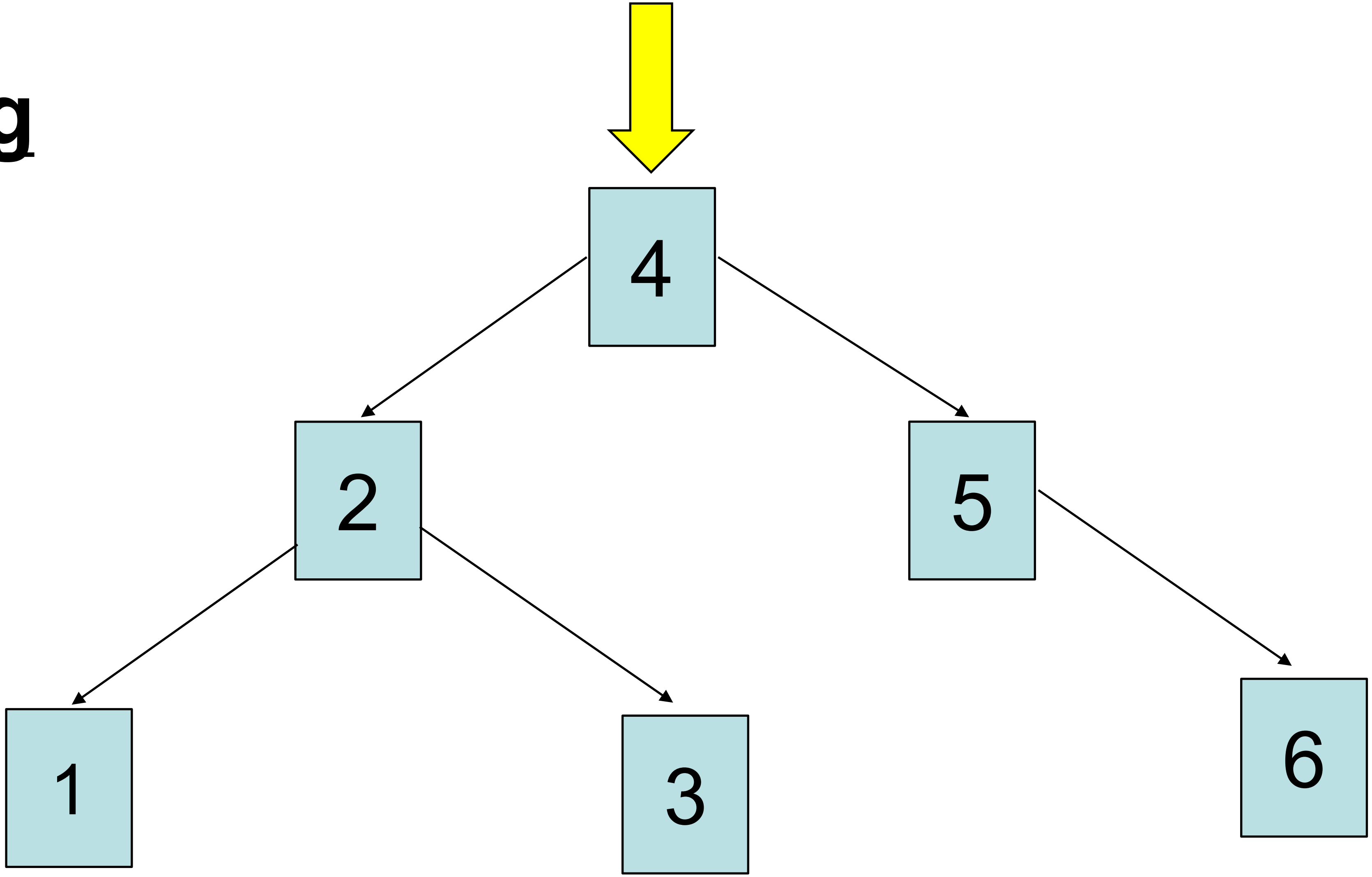


$\Theta(x)$?



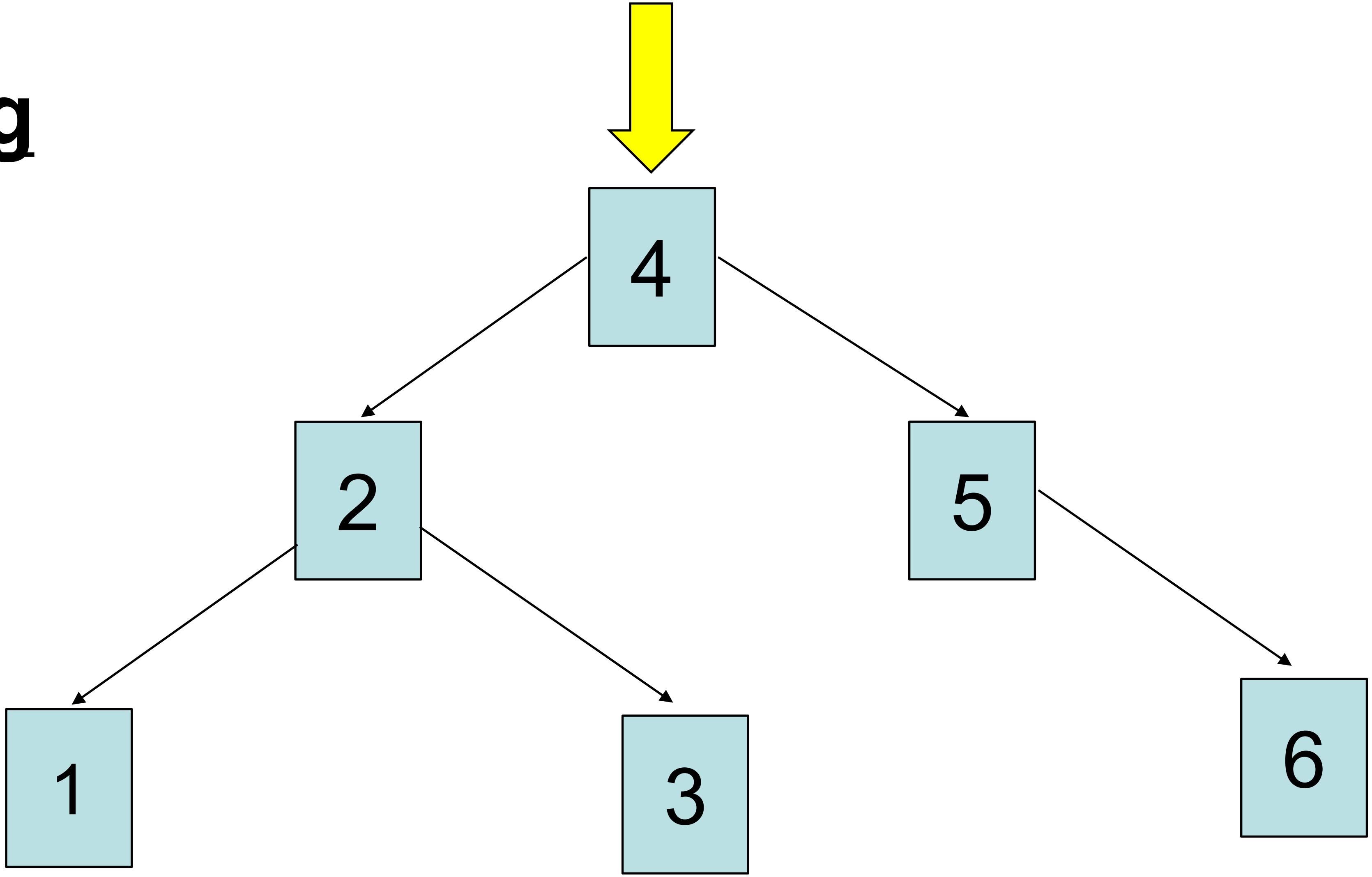
Deleting

- Delete "6"
- Delete "5"
- Delete "4"



Deleting

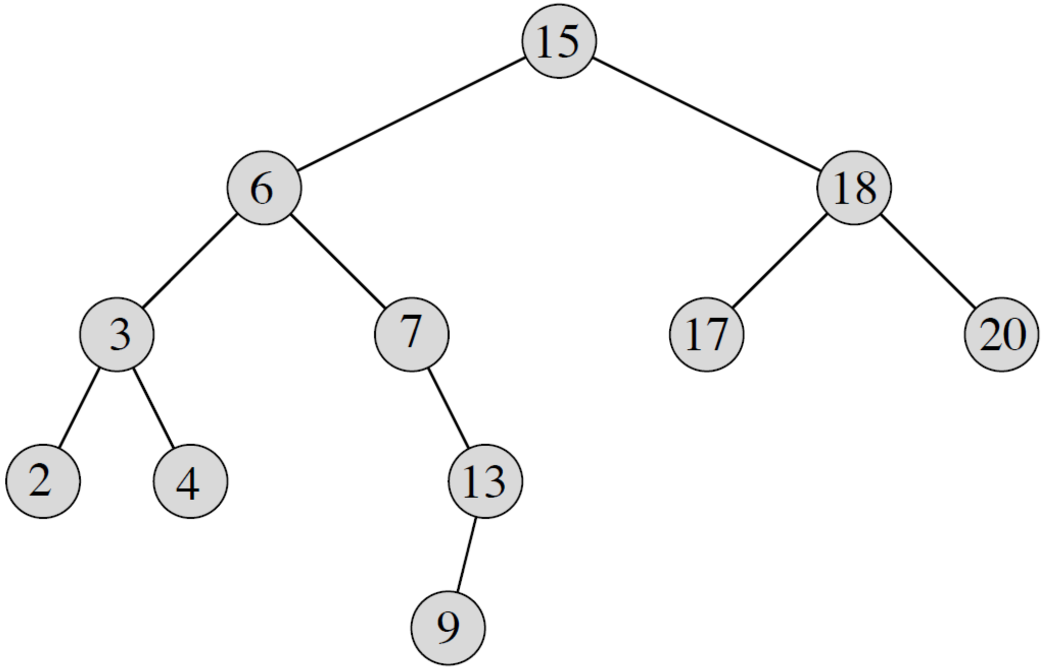
- Delete "6"
- Delete "5"
- Delete "4"



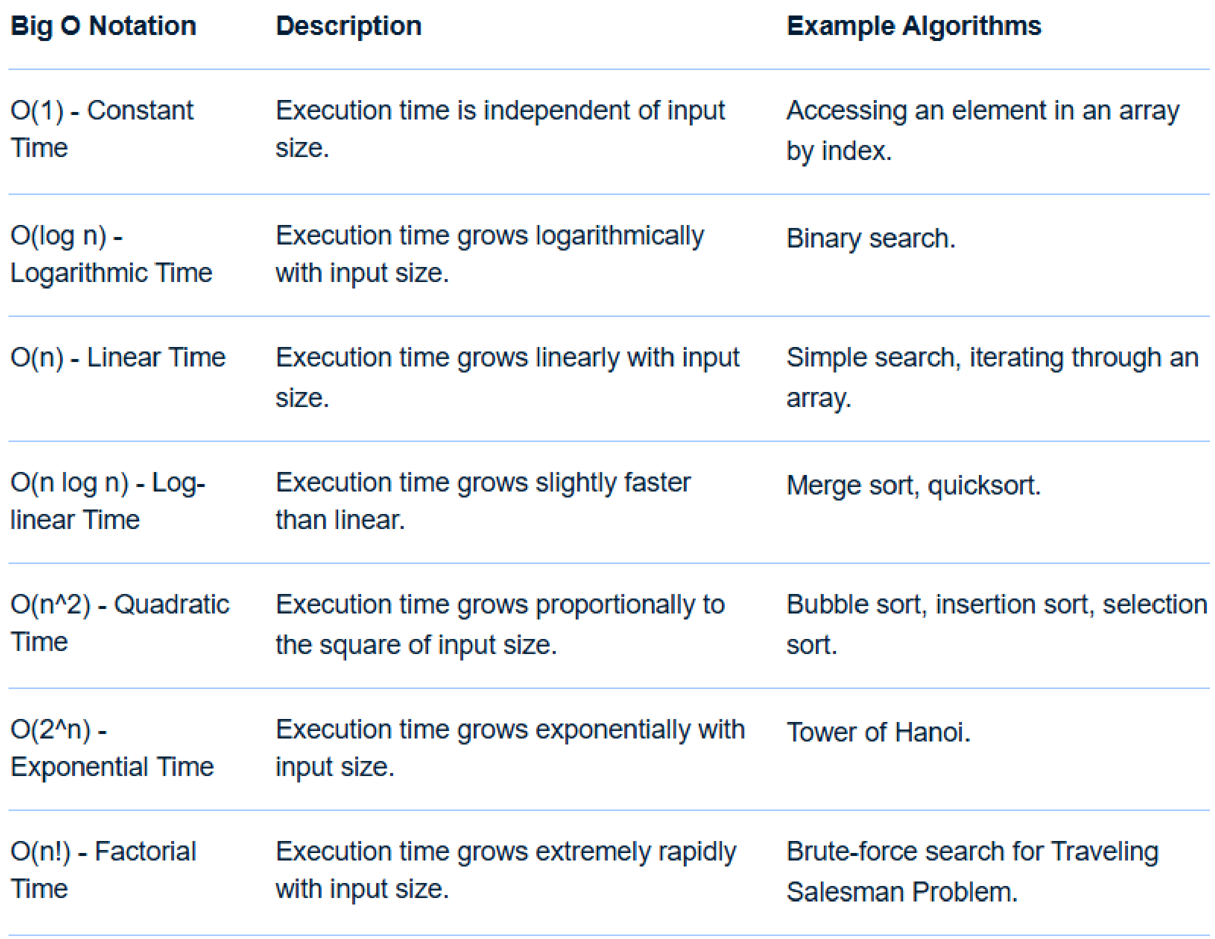
Trees - Implementation

```
TREE−SEARCH($x$, $k$)
if $x$ == NIL or $k$ == $x.key$
    return x
if k < x.key
    return TREE−SEARCH($x.left$, $k$)
else return TREE−SEARCH($x.right$, $k$)
```

The running time of TREE−SEARCH is $O(h)$, where h is the **height** of the tree.

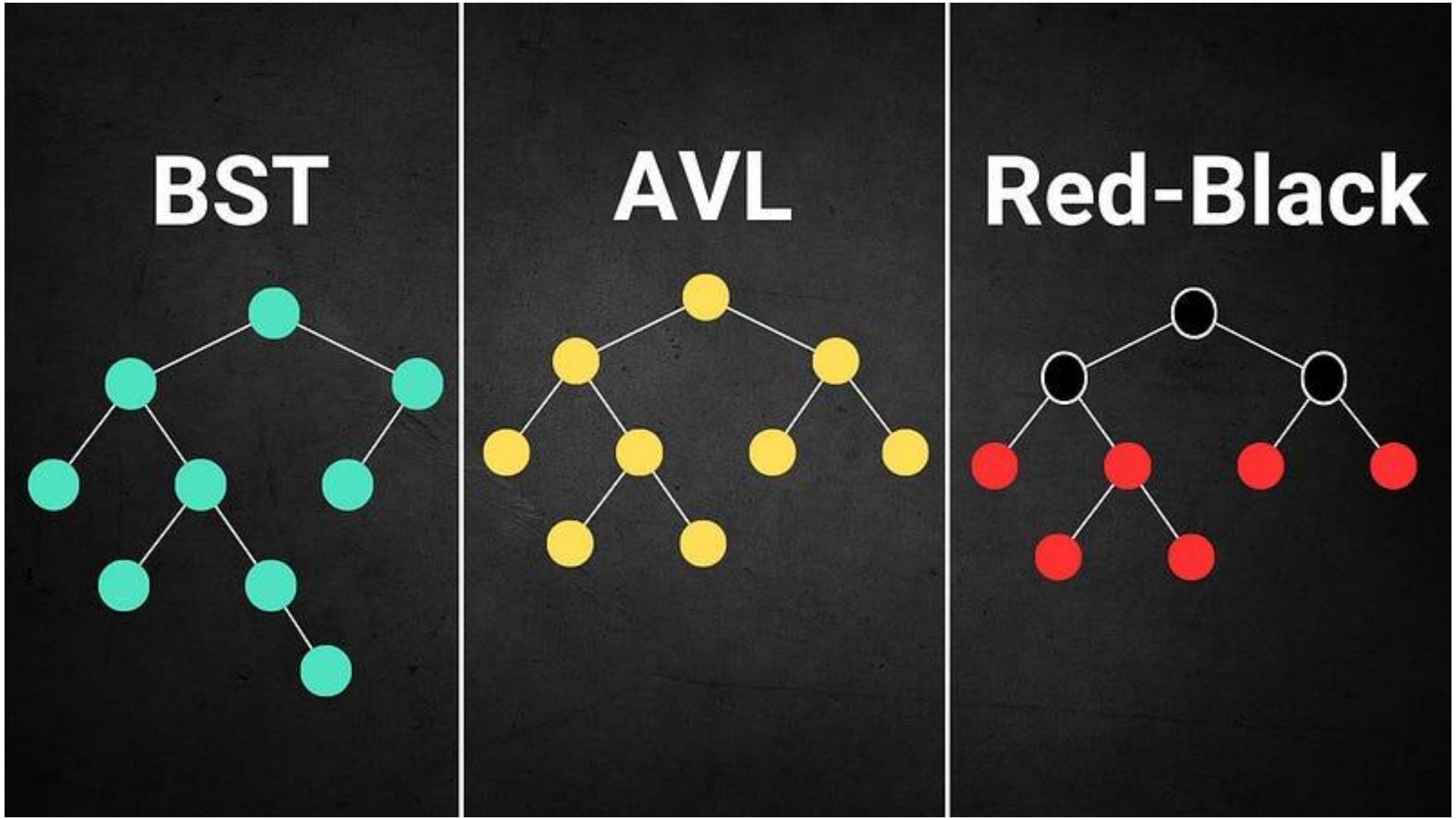


Why trees?



Big O Notation	Description	Example Algorithms
O(1) - Constant Time	Execution time is independent of input size.	Accessing an element in an array by index.
O(log n) - Logarithmic Time	Execution time grows logarithmically with input size.	Binary search.
O(n) - Linear Time	Execution time grows linearly with input size.	Simple search, iterating through an array.
O(n log n) - Log-linear Time	Execution time grows slightly faster than linear.	Merge sort, quicksort.
O(n^2) - Quadratic Time	Execution time grows proportionally to the square of input size.	Bubble sort, insertion sort, selection sort.
O(2^n) - Exponential Time	Execution time grows exponentially with input size.	Tower of Hanoi.
O(n!) - Factorial Time	Execution time grows extremely rapidly with input size.	Brute-force search for Traveling Salesman Problem.

Different types of trees



B-tree (self-balanced tree)

B-trees
5
43
71
97
29
37
53
59
61
67
73
83

Other type of structures

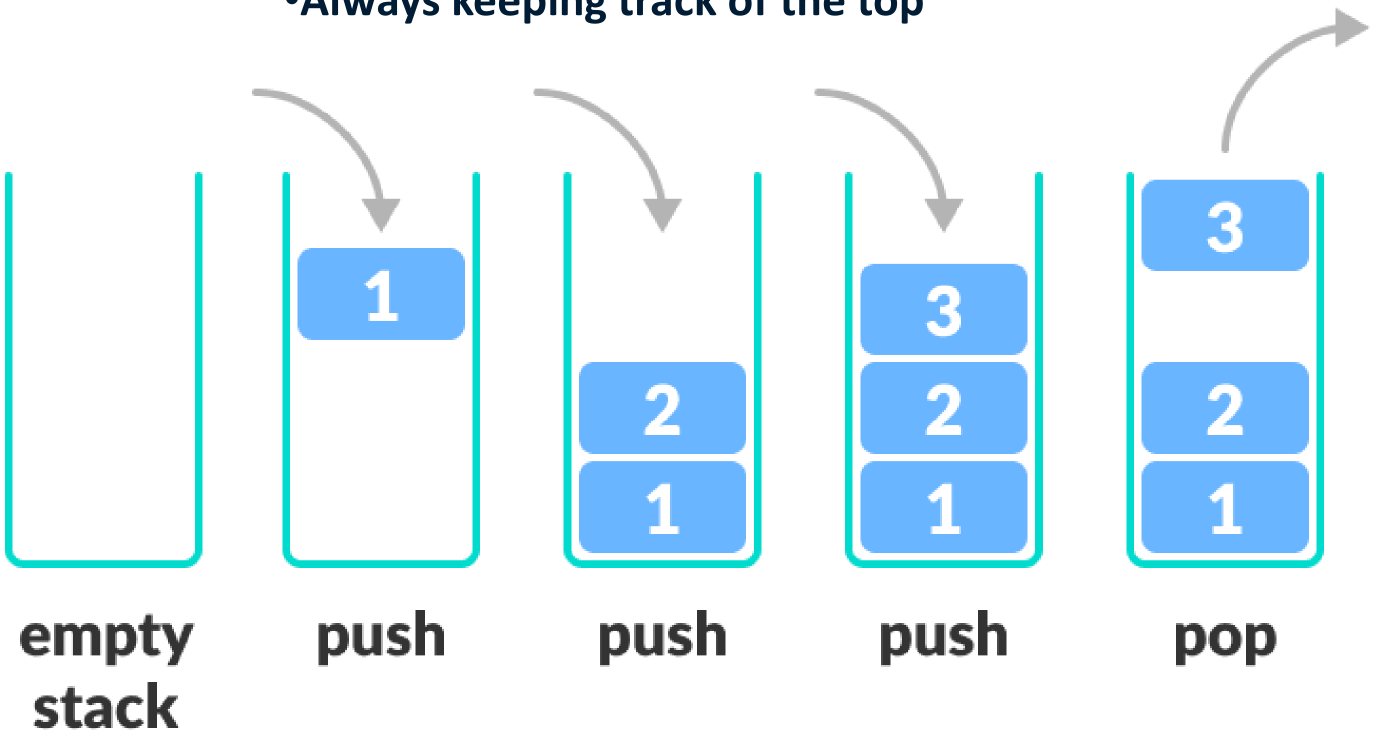
Stacks

A stack is a data structure that operates on a Last-In, First-Out (LIFO) principle, like a stack of plates where you can only add or remove plates from the top. Imagine a stack of books, you can only access the top book, not the ones underneath.

- **Operations:**
  - **Push:** Adds an element to the top of the stack.
  - **Pop:** Removes and returns the top element from the stack.
  - **Peek:** Returns the top element without removing it.

Stacks

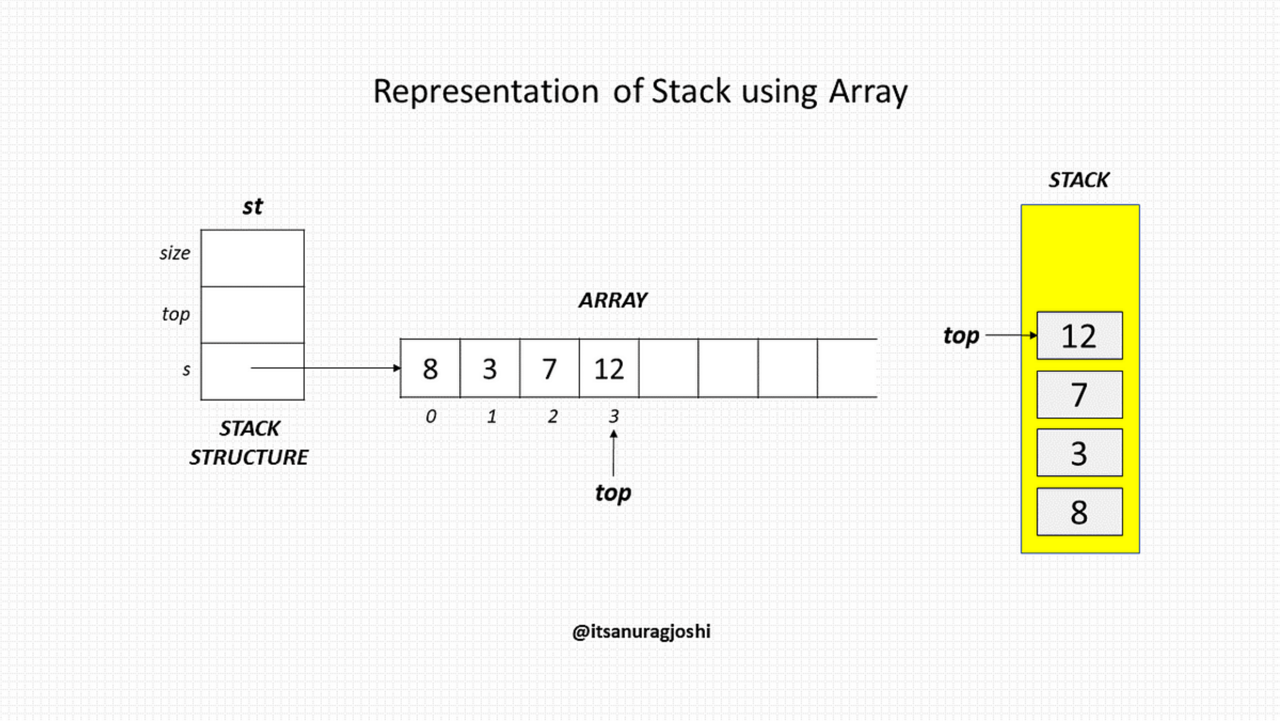
•Always keeping track of the top

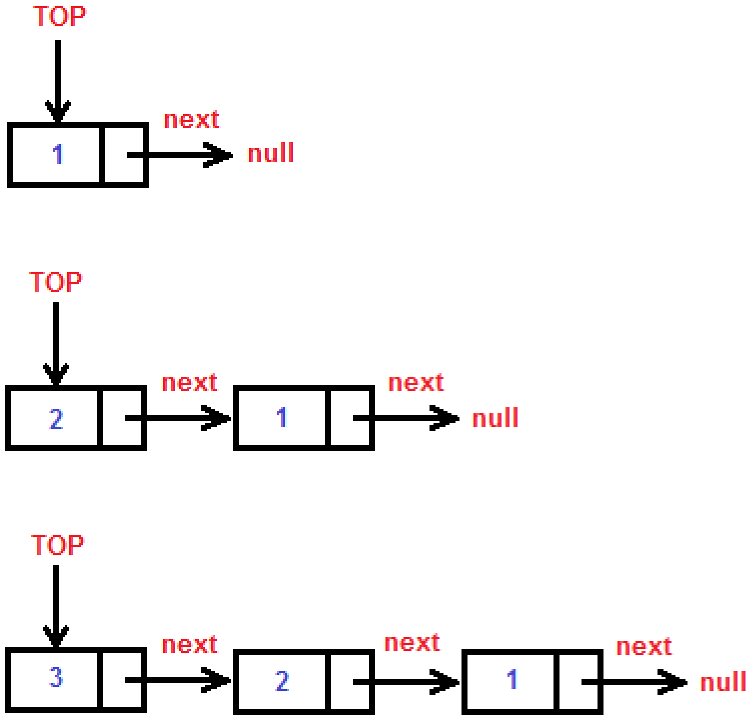


- Scenarios
  - Pushing to an empty stack
  - Pushing to an existing stack
  - Popping from an empty stack
  - Popping an existing stack

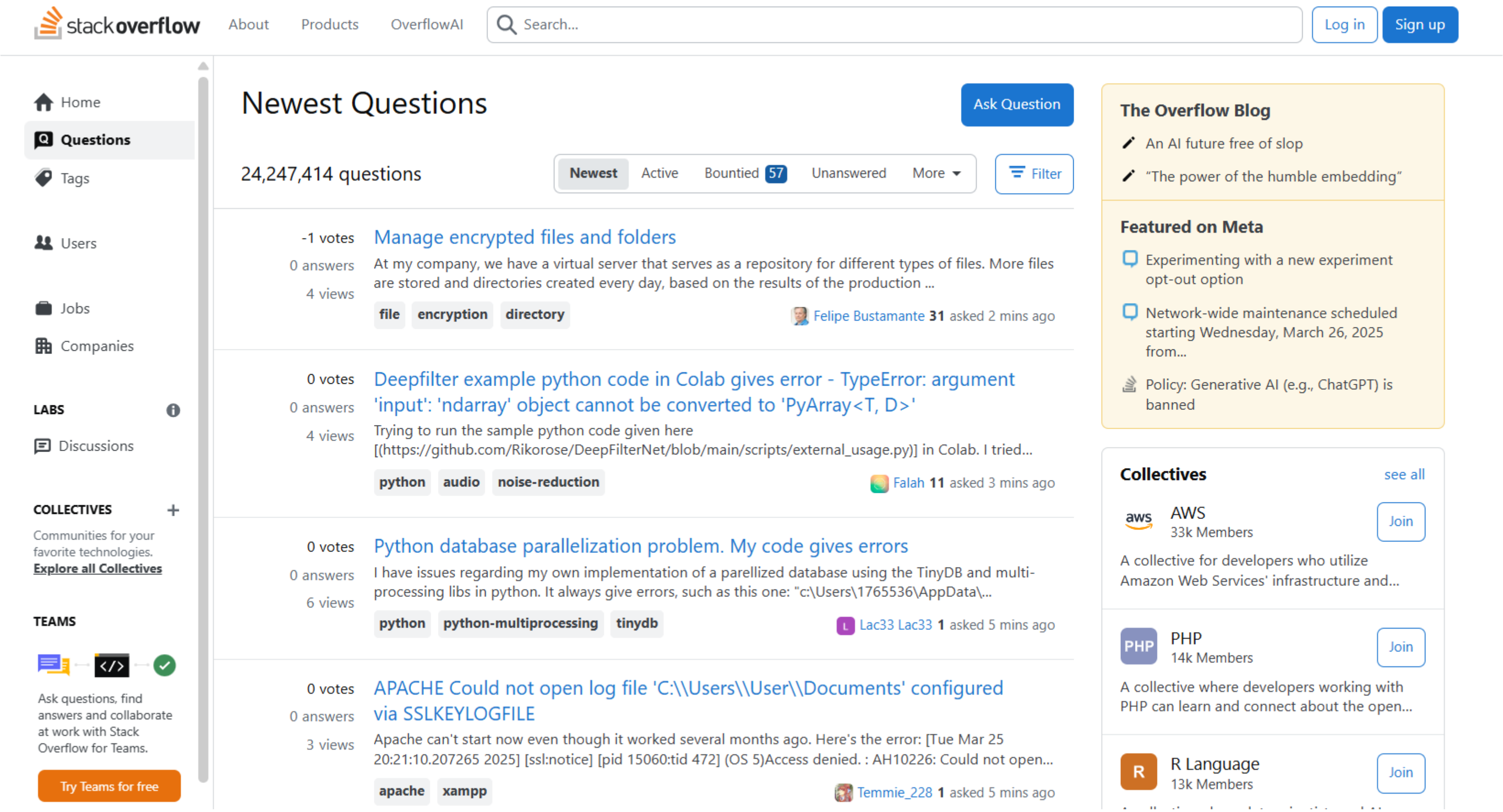
- Extra operations:
  - Consulting element on the top
  - Tracking count

Stack: Static vs. Dynamic





Stack: examples?



Newest Questions

24,247,414 questions

-1 votes
0 answers
4 views

Manage encrypted files and folders

At my company, we have a virtual server that serves as a repository for different types of files. More files are stored and directories created every day, based on the results of the production ...

file encryption directory

Felipe Bustamante 31 asked 2 mins ago

0 votes
0 answers
4 views

Deepfilter example python code in Colab gives error - TypeError: argument 'input': 'ndarray' object cannot be converted to 'PyArray<T, D>'

Trying to run the sample python code given here [(https://github.com/Rikorose/DeepFilterNet/blob/main/scripts/external_usage.py)] in Colab. I tried...

python audio noise-reduction

Falah 11 asked 3 mins ago

0 votes
0 answers
6 views

Python database parallelization problem. My code gives errors

I have issues regarding my own implementation of a parellized database using the TinyDB and multi-processing libs in python. It always give errors, such as this one: "c:\Users\1765536\AppData\...

python python-multiprocessing tinydb

Lac33 Lac33 1 asked 5 mins ago

0 votes
0 answers
3 views

APACHE Could not open log file 'C:\\Users\\User\\Documents' configured via SSLKEYLOGFILE

Apache can't start now even though it worked several months ago. Here's the error: [Tue Mar 25 20:21:10.207265 2025] [ssl:notice] [pid 15060:tid 472] (OS 5)Access denied. : AH10226: Could not open...

apache xampp

Temmie_228 1 asked 5 mins ago

The Overflow Blog

- An AI future free of slop
- “The power of the humble embedding”

Featured on Meta

- Experimenting with a new experiment opt-out option
- Network-wide maintenance scheduled starting Wednesday, March 26, 2025 from...
- Policy: Generative AI (e.g., ChatGPT) is banned

Collectives see all

AWS
33k Members
Join
A collective for developers who utilize Amazon Web Services' infrastructure and...

PHP
14k Members
Join
A collective where developers working with PHP can learn and connect about the open...

R Language
13k Members
Join

Queues

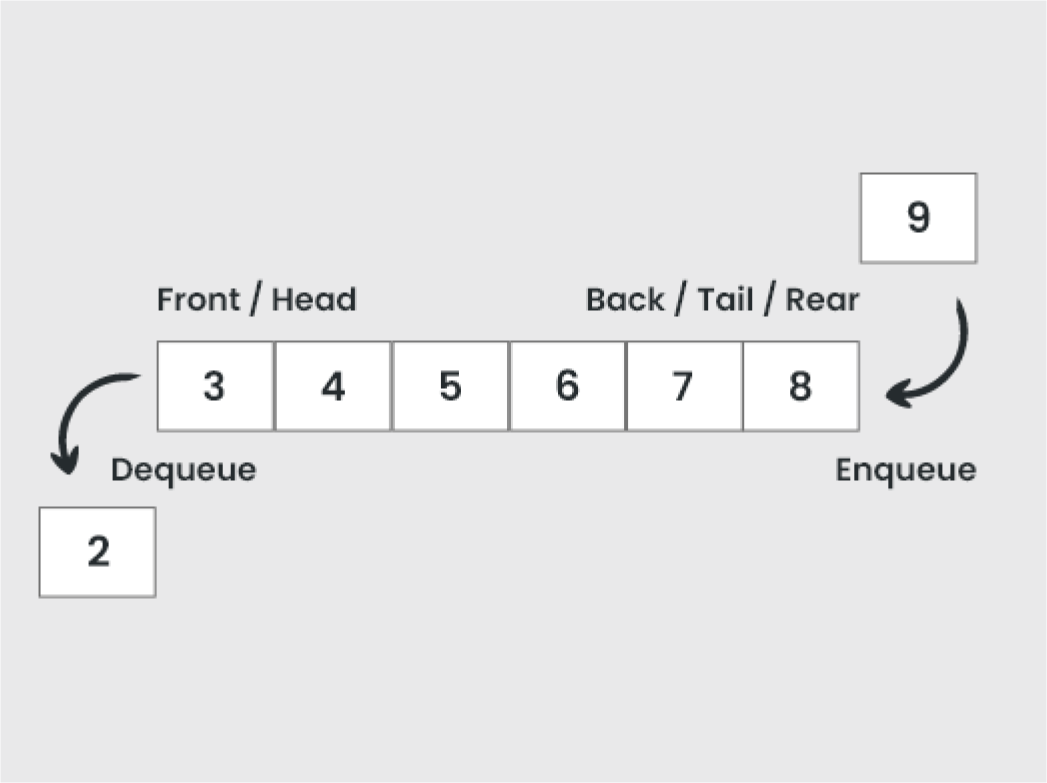
Queues

Is a linear data structure that follows the "first-in, first-out" (FIFO) principle, meaning the first element added is the first one to be removed, like a line at a checkout counter.

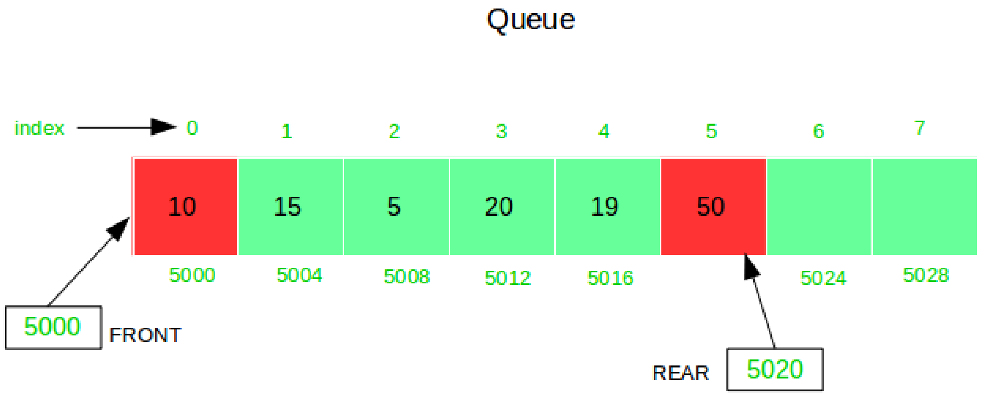
•The core principle of a queue is that elements are processed in the order they were added.

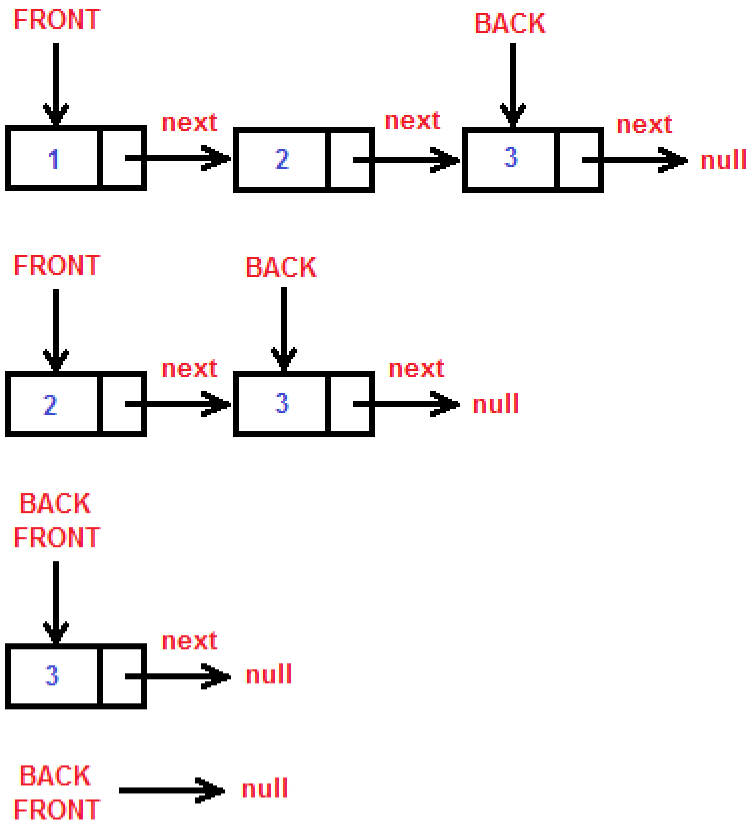
•**Linear Data Structure:**

•Queues are organized sequentially, with elements stored in a specific order.

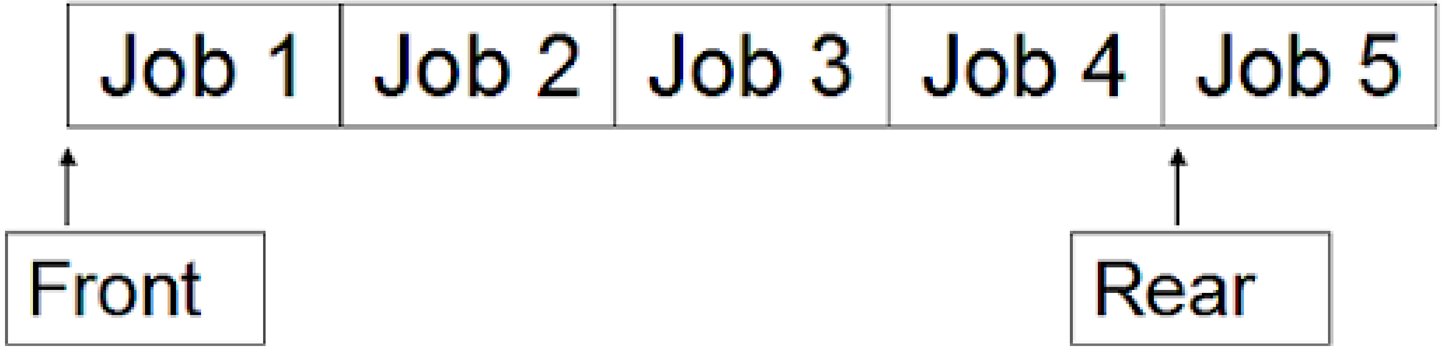


Queues: Static vs Dynamic





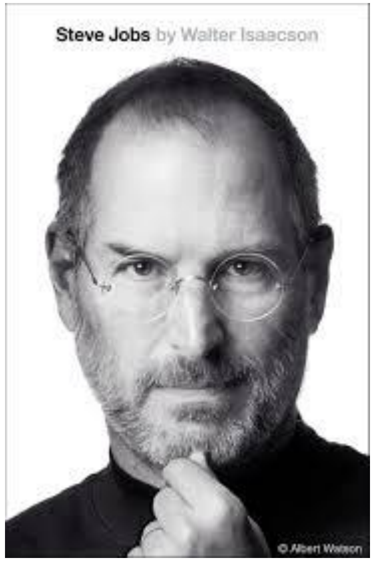
Queues: Examples?



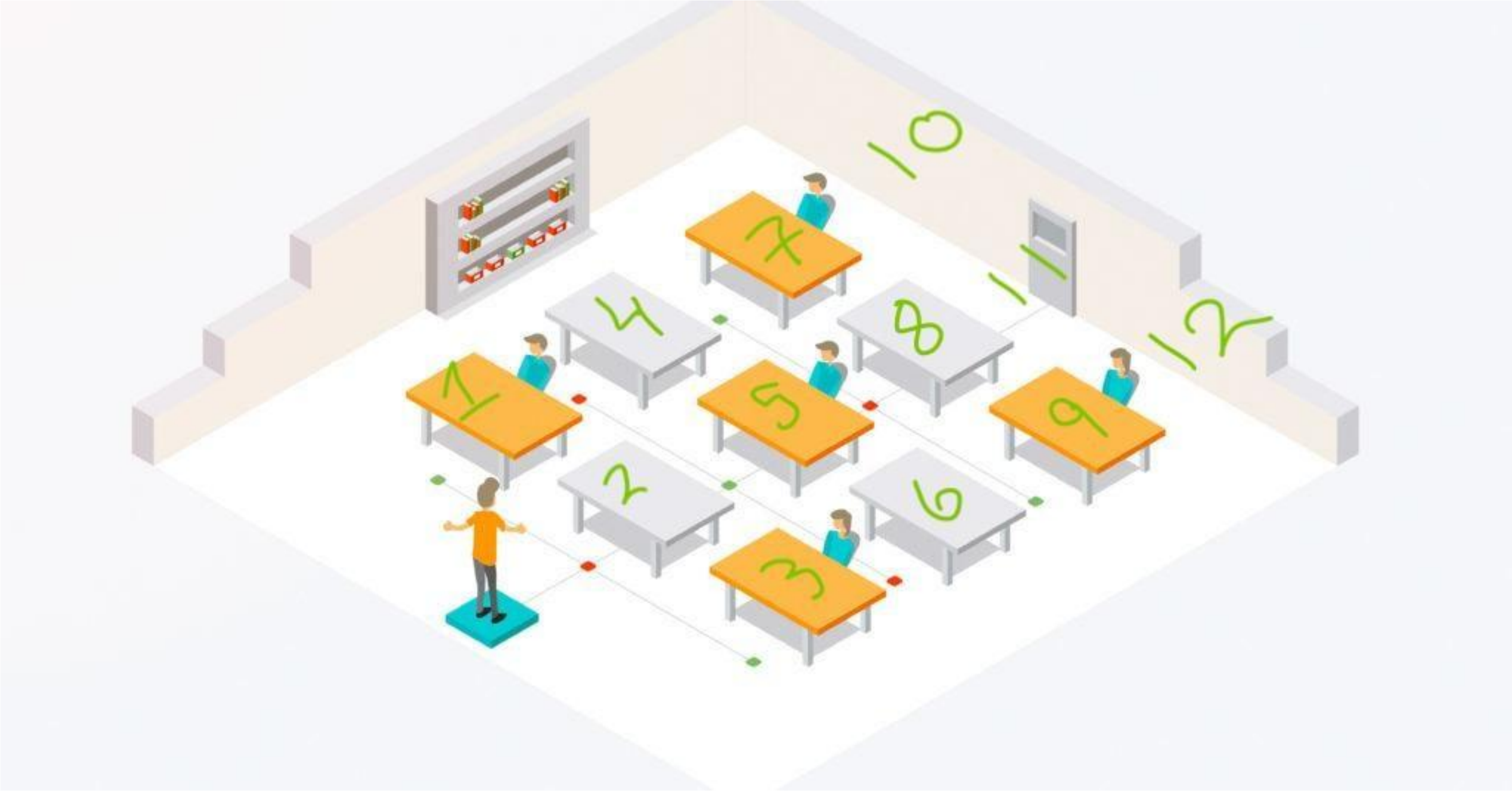
Job 1	Job 2	Job 3	Job 4	Job 5

Front

Rear



Hash map/table



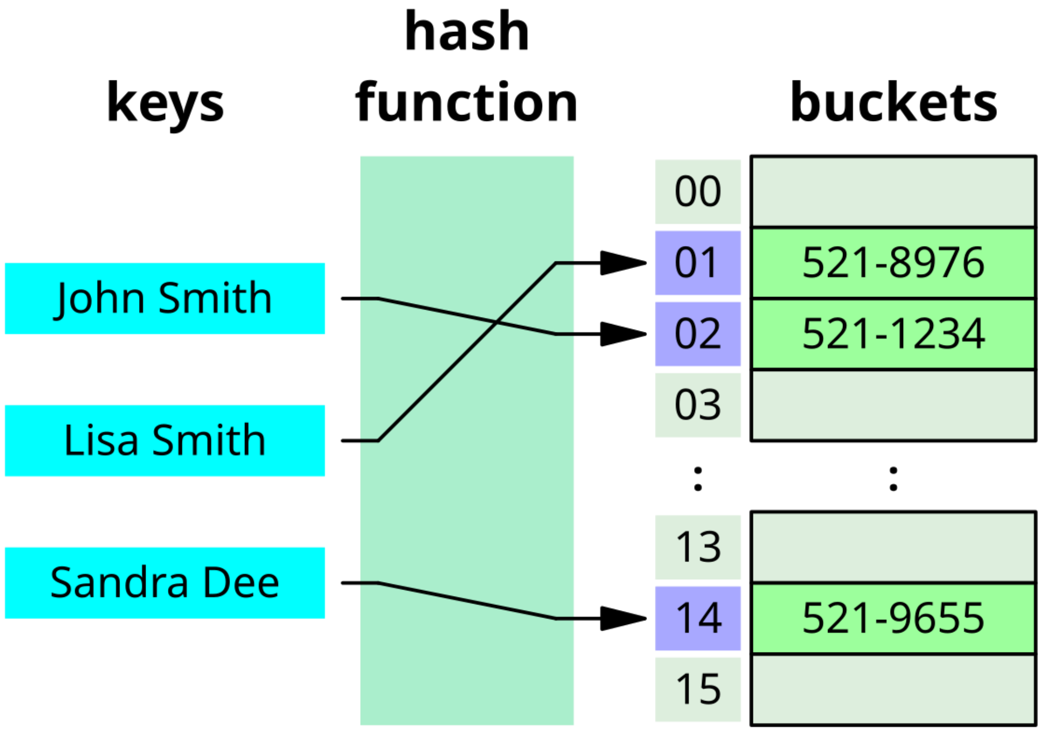
Hash map/table

A Hashmap is a data structure that stores data as key-value pairs, using a hash function to map keys to their corresponding values, enabling efficient retrieval and storage of data.
Here's a more detailed explanation:

- **Key-Value Pairs:**
  HashMaps store data in the form of key-value pairs, where each key is unique and maps to a specific value.

- **Hash Function:**
  A hash function is used to convert the key into an index (or hash code) that determines where the value is stored in the underlying array (or "buckets").

- **Efficient Retrieval:**
  Hashmaps allow for very fast retrieval of values by using the hash function to quickly locate the corresponding index in the array.

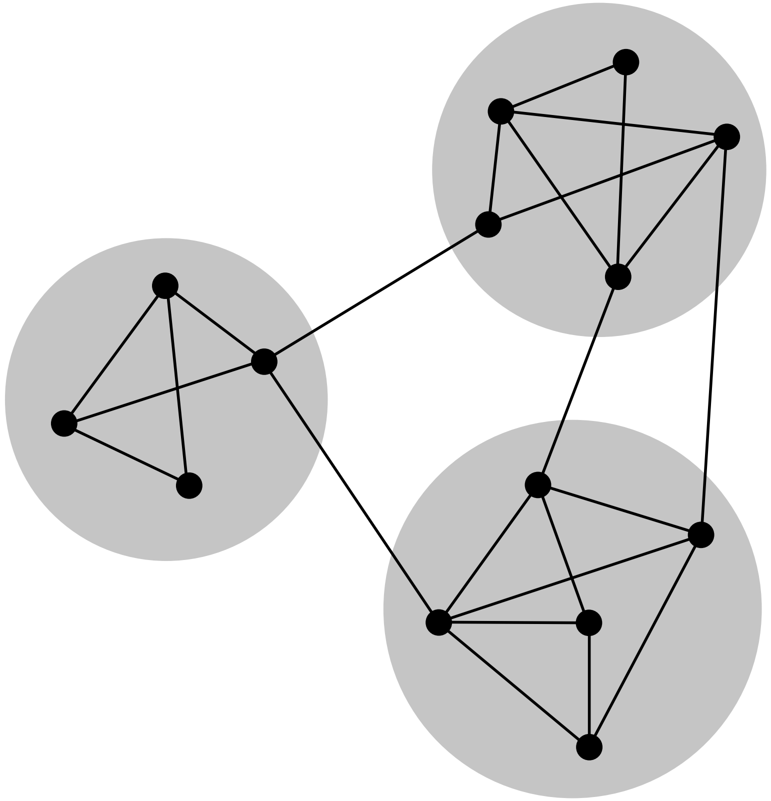


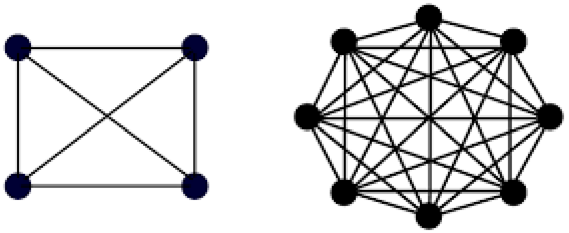
Hash map/table

Algorithm to count how many times the word "Hashmap" appears on the internet!

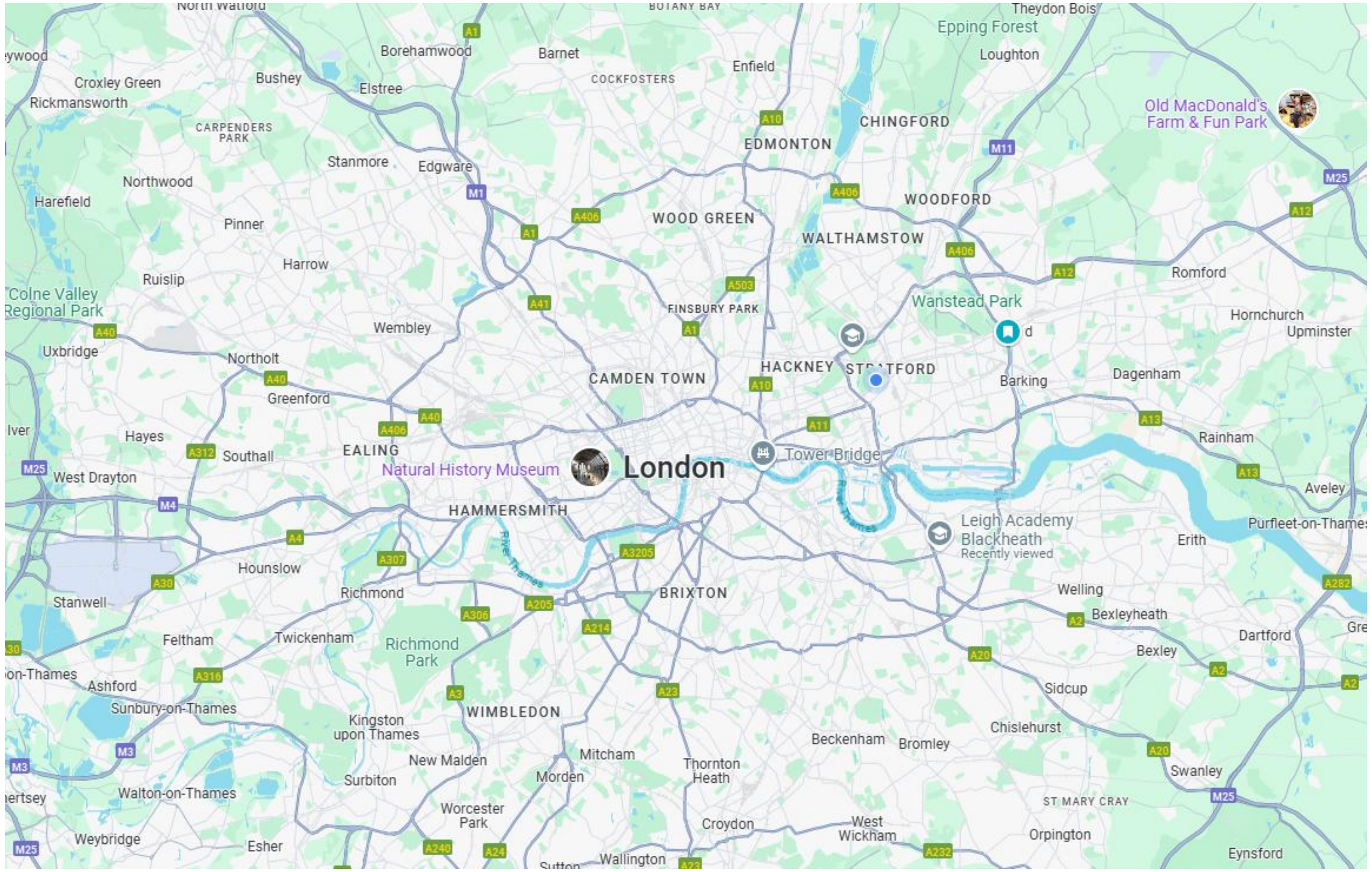
Why is it faster?

Graphs





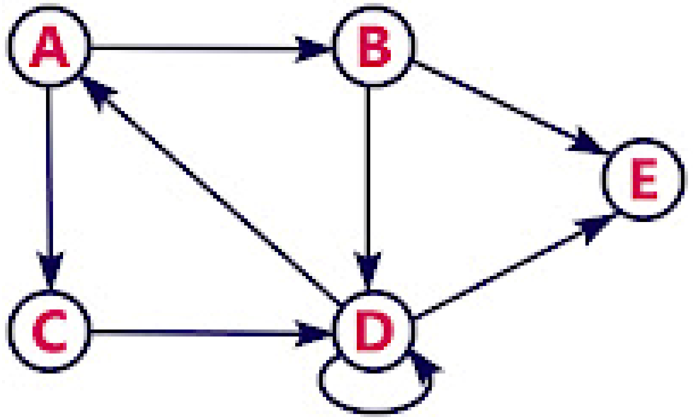
Graphs



https://maps.app.goo.gl/EHzJX8W8iqFosSMW7

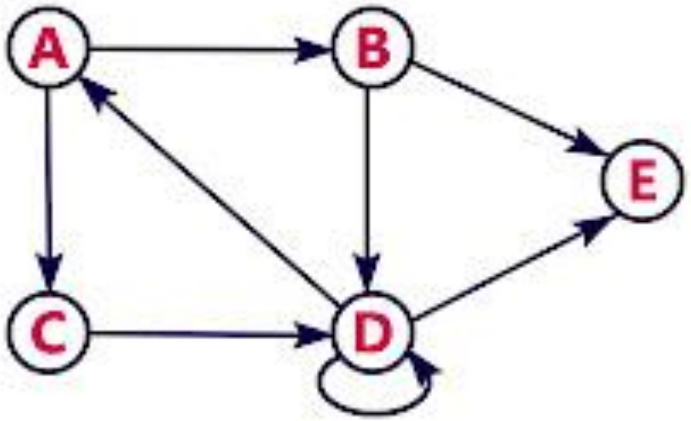
Graphs

- How to translate to code the connectivity?



Graphs

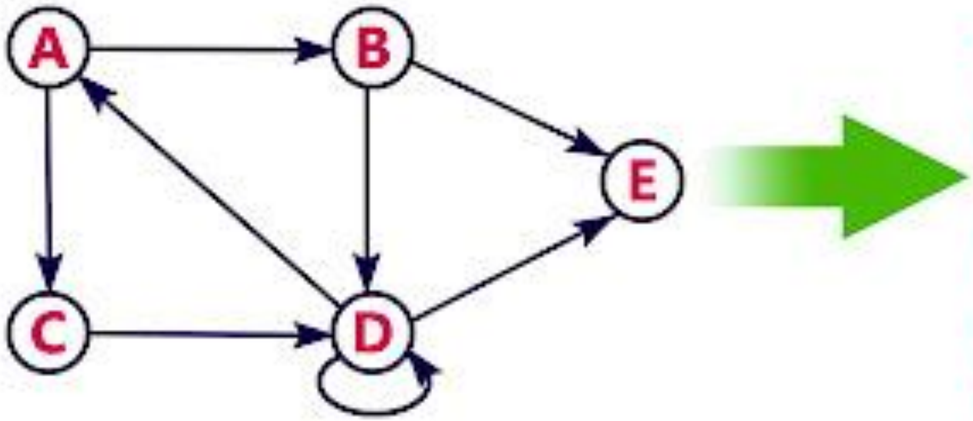
- How to translate to code the connectivity?



$$
\begin{array}{c|ccccc|}
 & A & B & C & D & E \\
\hline
A & 0 & 1 & 1 & 0 & 0 \\
B & 0 & 0 & 0 & 1 & 1 \\
C & 0 & 0 & 0 & 1 & 0 \\
D & 1 & 0 & 0 & 1 & 1 \\
E & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Exercise

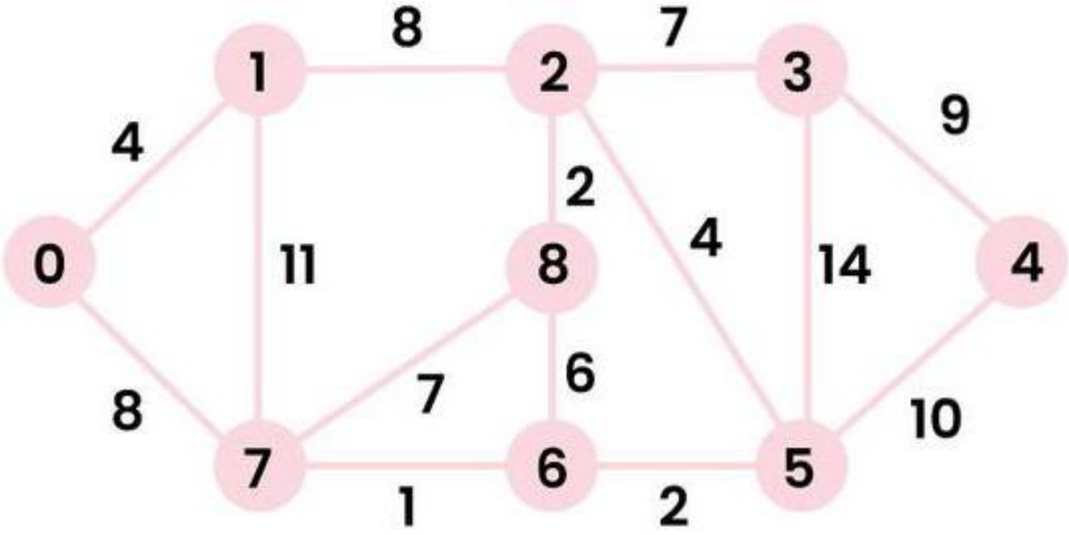
- Algorithm to find the best path (minimum cost) "

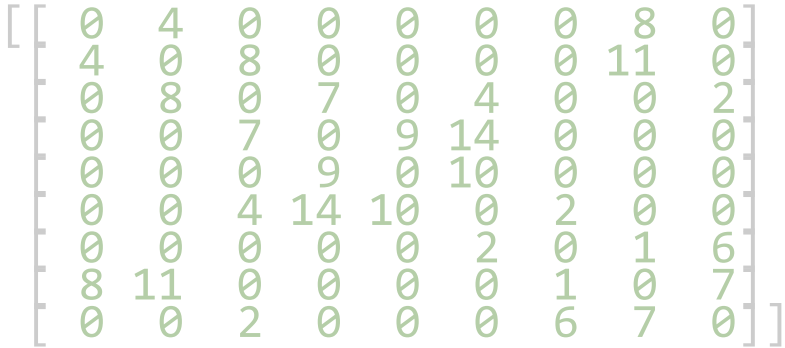


$$\begin{array}{c|ccccc}
 & A & B & C & D & E \\
\hline
A & 0 & 1 & 1 & 0 & 0 \\
B & 0 & 0 & 0 & 1 & 1 \\
C & 0 & 0 & 0 & 1 & 0 \\
D & 1 & 0 & 0 & 1 & 1 \\
E & 0 & 0 & 0 & 0 & 0
\end{array}$$

Breadth first

- I will look for all the neighbors first before moving to the next node.

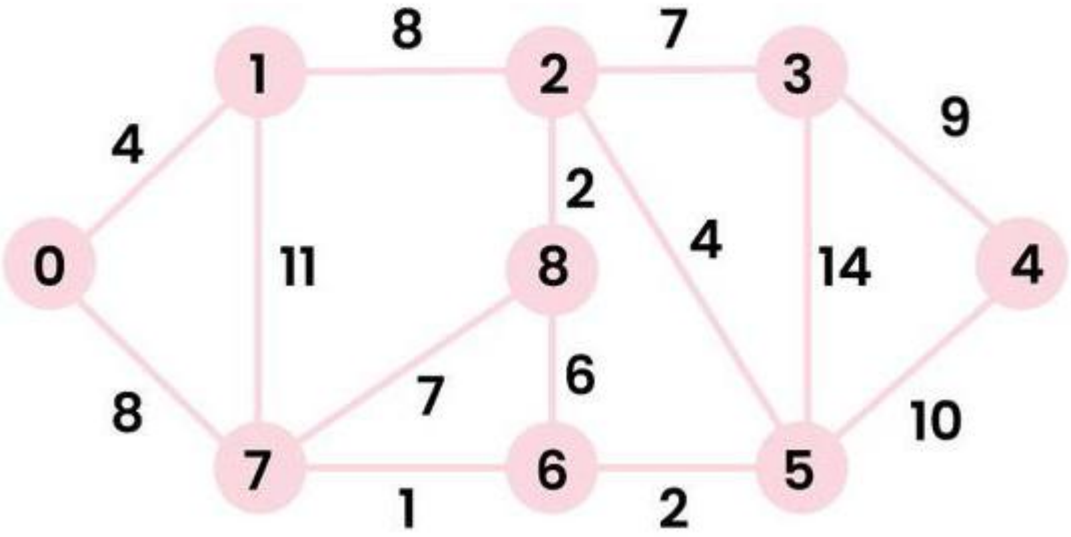


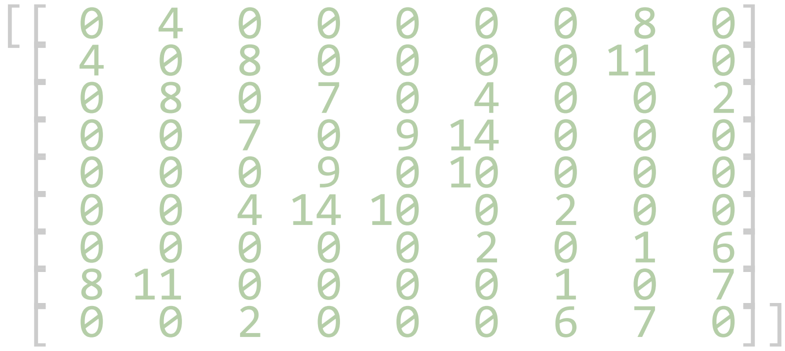


```
[[ 0  4  0  0  0  0  0  8  0]
 [ 4  0  8  0  0  0  0 11  0]
 [ 0  8  0  7  0  4  0  0  2]
 [ 0  0  7  0  9 14  0  0  0]
 [ 0  0  0  9  0 10  0  0  0]
 [ 0  0  4 14 10  0  2  0  0]
 [ 0  0  0  0  0  2  0  1  6]
 [ 8 11  0  0  0  0  1  0  7]
 [ 0  0  2  0  0  0  6  7  0]]
```

Depht first

- I will go as far away as I can first before checking my neighbors.

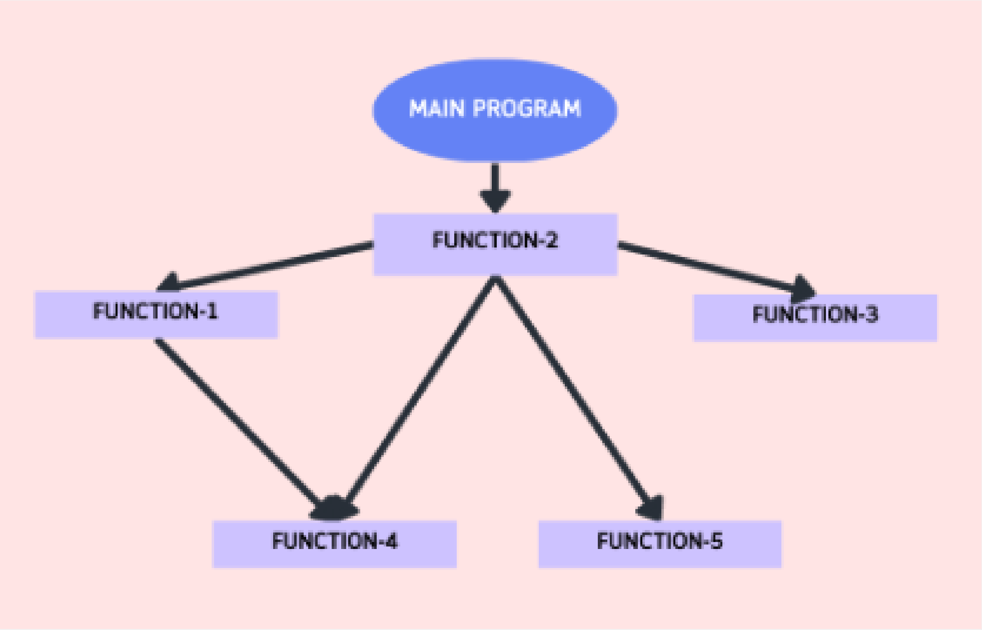




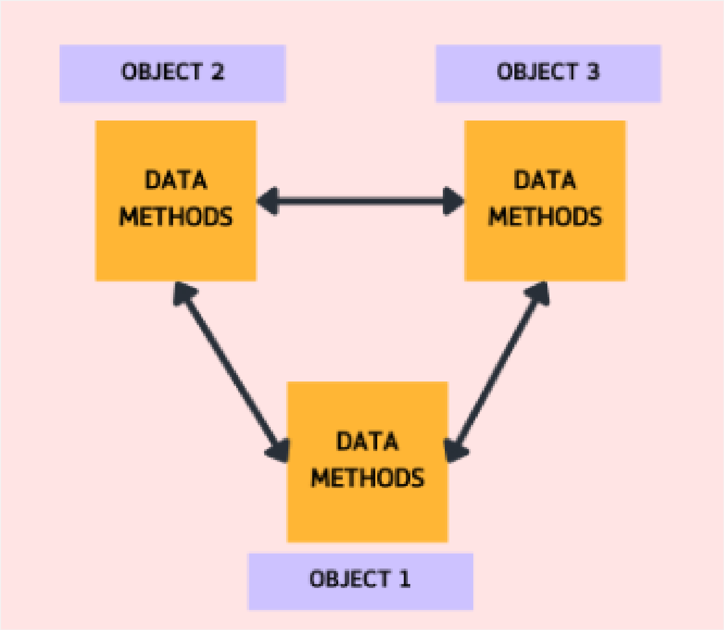
```
[[ 0  4  0  0  0  0  0  8  0]
 [ 4  0  8  0  0  0  0 11  0]
 [ 0  8  0  7  0  4  0  0  2]
 [ 0  0  7  0  9 14  0  0  0]
 [ 0  0  0  9  0 10  0  0  0]
 [ 0  0  4 14 10  0  2  0  0]
 [ 0  0  0  0  0  2  0  1  6]
 [ 8 11  0  0  0  0  1  0  7]
 [ 0  0  2  0  0  0  6  7  0]]
```

Exercises

Procedural programming (POP) vs OOP



POP



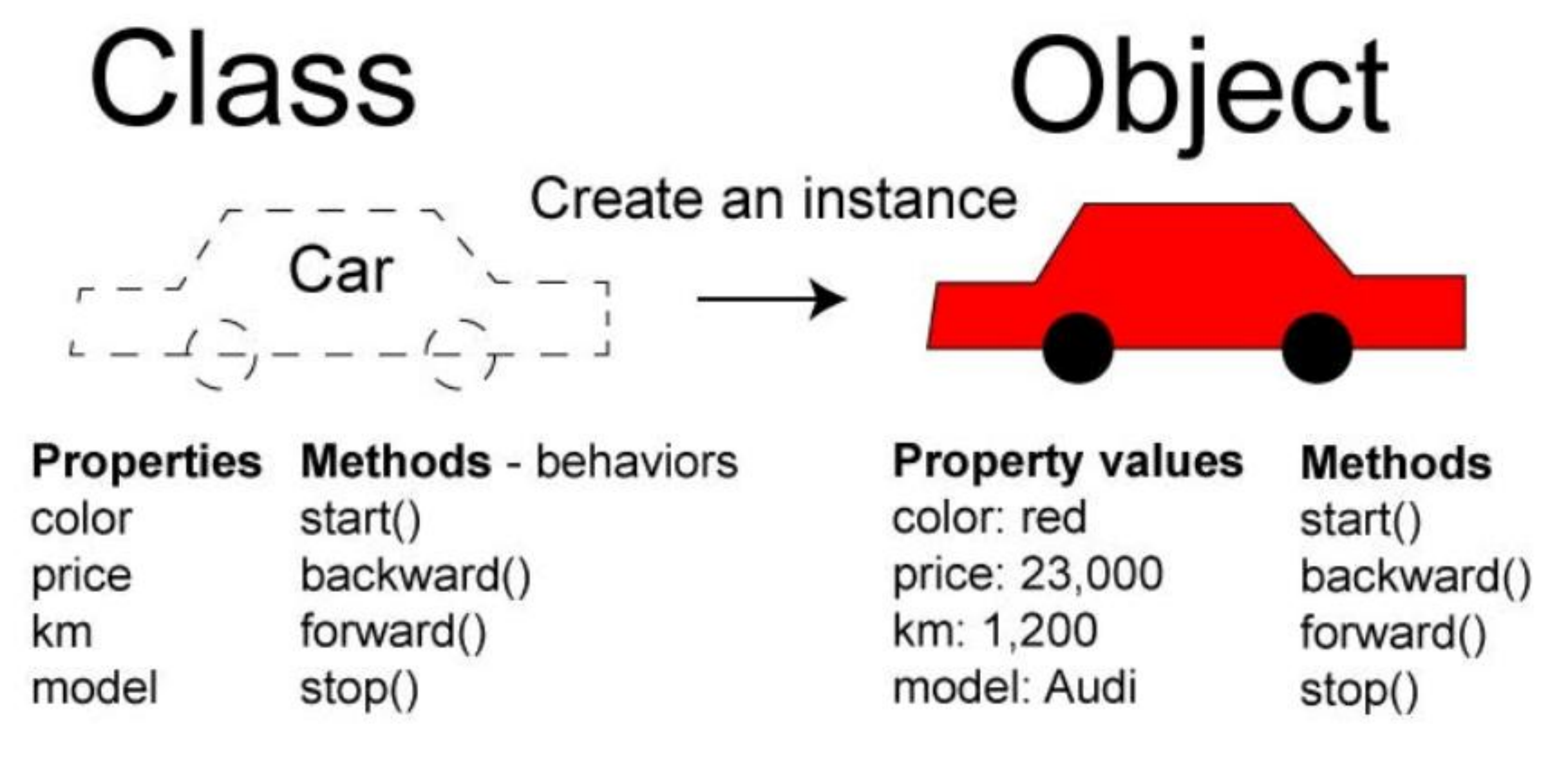
OOP

Class and Object

Class – define properties/attributes (variables), behaviors/functionalities (methods)

Object – instances with specific values of properties, their behaviours

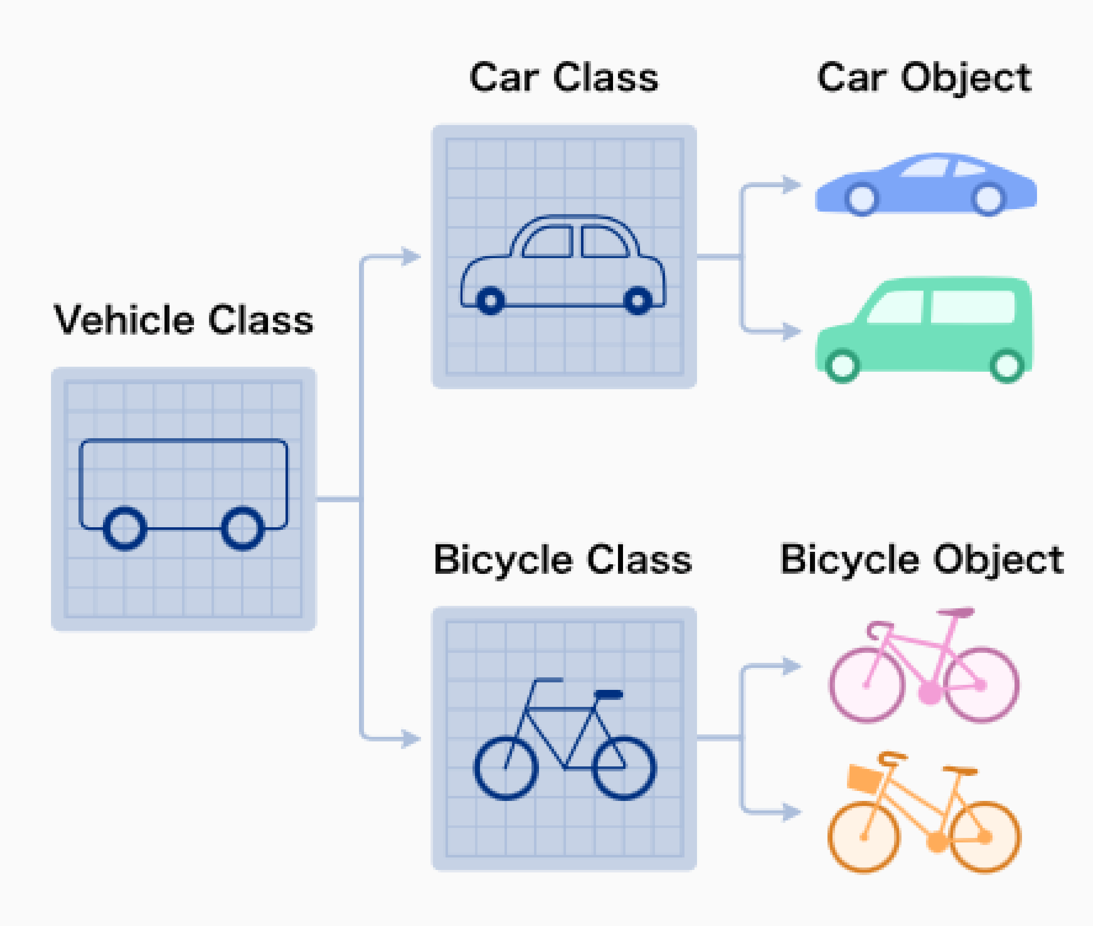
An Object is a data structure with both data (variables called attributes) And code (functions called methods)



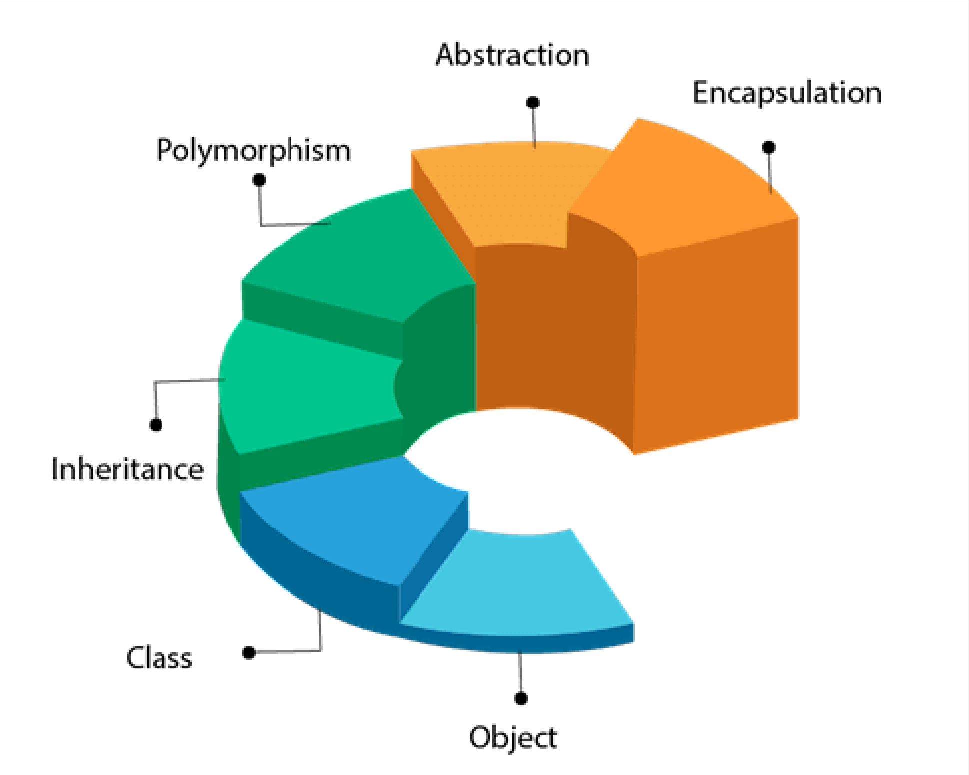
Properties	**Methods** - behaviors
color	start()
price	backward()
km	forward()
model	stop()

Property values	**Methods**
color: red	start()
price: 23,000	backward()
km: 1,200	forward()
model: Audi	stop()

Using classes and objects

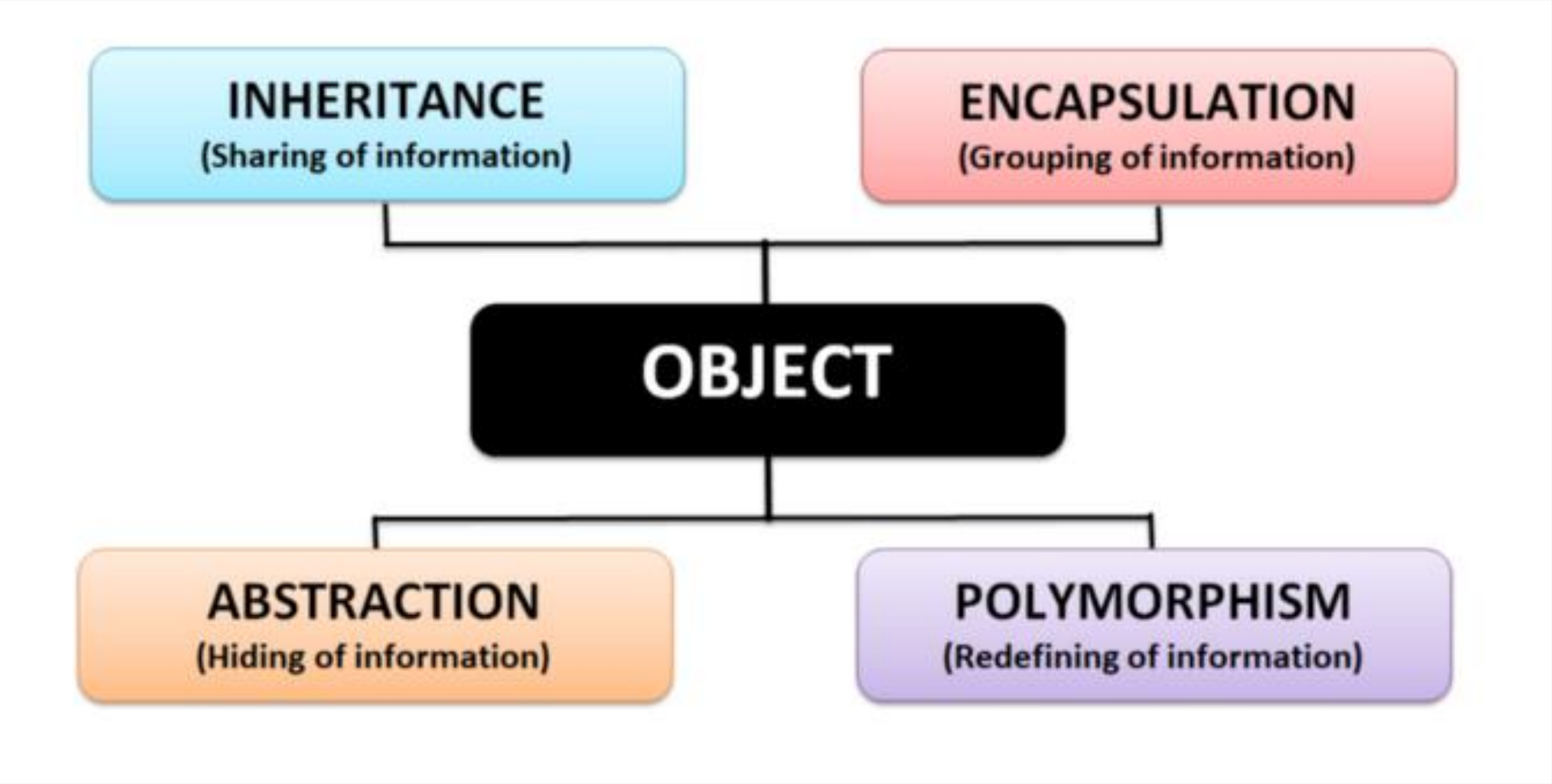


Important concepts in OOP



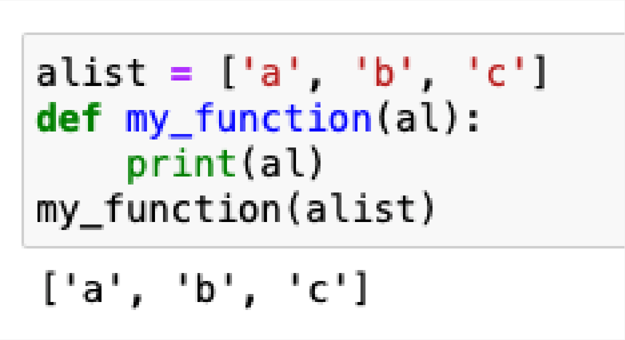
- **Classes** (types)
- **Objects and Instances**
- **Encapsulation** : Data manipulation
- **Inheritance** : Code re-use
- **Polymorphism** : Code efficiency

OOP concepts - closer look



Python

- Python is an interpreted, object-oriented programming language.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid code development
  - No need to declare a type of variable. Easy syntax, readable, modular.
  - Indentation to identify separate blocks of code.



```
alist = ['a', 'b', 'c']
def my_function(al):
    print(al)
my_function(alist)
```

Output:

```
['a', 'b', 'c']
```

Tasks in python

- Translate your code to python!
- Implement a linked list using classes
- Use the linked list to keep track of the visited nodes
- Use a tree to look for the visited nodes
- Implement your map as a graph
- Implement the searches for Breadth and Depth first methods

Thank you for your attention

Please leave your feedback.

Use the QR code or link:

SJTU Winter School- Tozadore Feedback – Fill in form