WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Principles of Software Programming: Basic Algorithms

Svitlana Vakulenko, MSc.

WS 2017

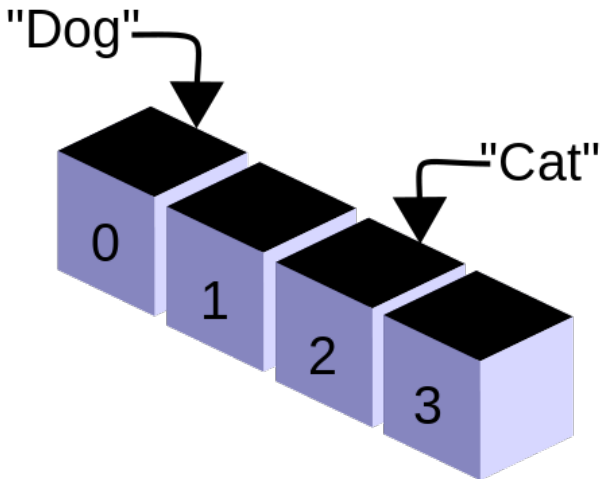
Acknowledgement: slides adopted from Dr. Jürgen Umbrich GZP SS 2014 lecture slides

This Episode

- 13:00-16:00
- Data structures:
  - Array
  - Vector
  - Linked list
  - Graph
  - Tree
- Basic Algorithms:
  - Search
  - Sort
  - Recursion

Array

- static (fixed size, does not grow)
- located next to each other in memory



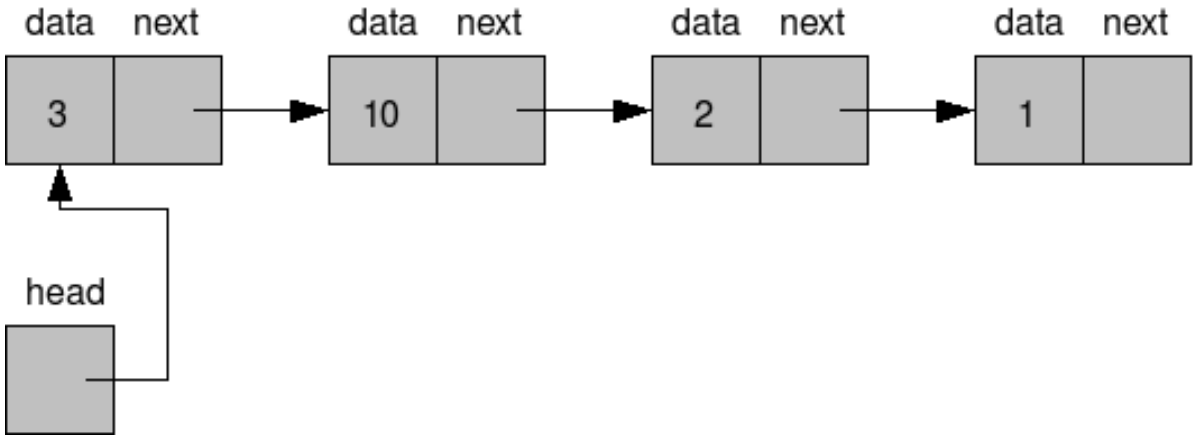
https://commons.wikimedia.org/wiki/File:CPT-programming-array.svg

- search: O(1)
- insert: O(N)
- (memory) space = n * element size
- **vector**: dynamic array (space x 2)

https://en.wikiversity.org/wiki/Data_Structures_and_Algorithms/Arrays,_Lists_and_Vectors

Linked list

- dynamic sequence of nodes (head/tail)
- **singly** linked: each node contains a link to another node
- **doubly** linked: 2 links (next and previous node)
- search: O(N)
- insert: O(N)
- space >

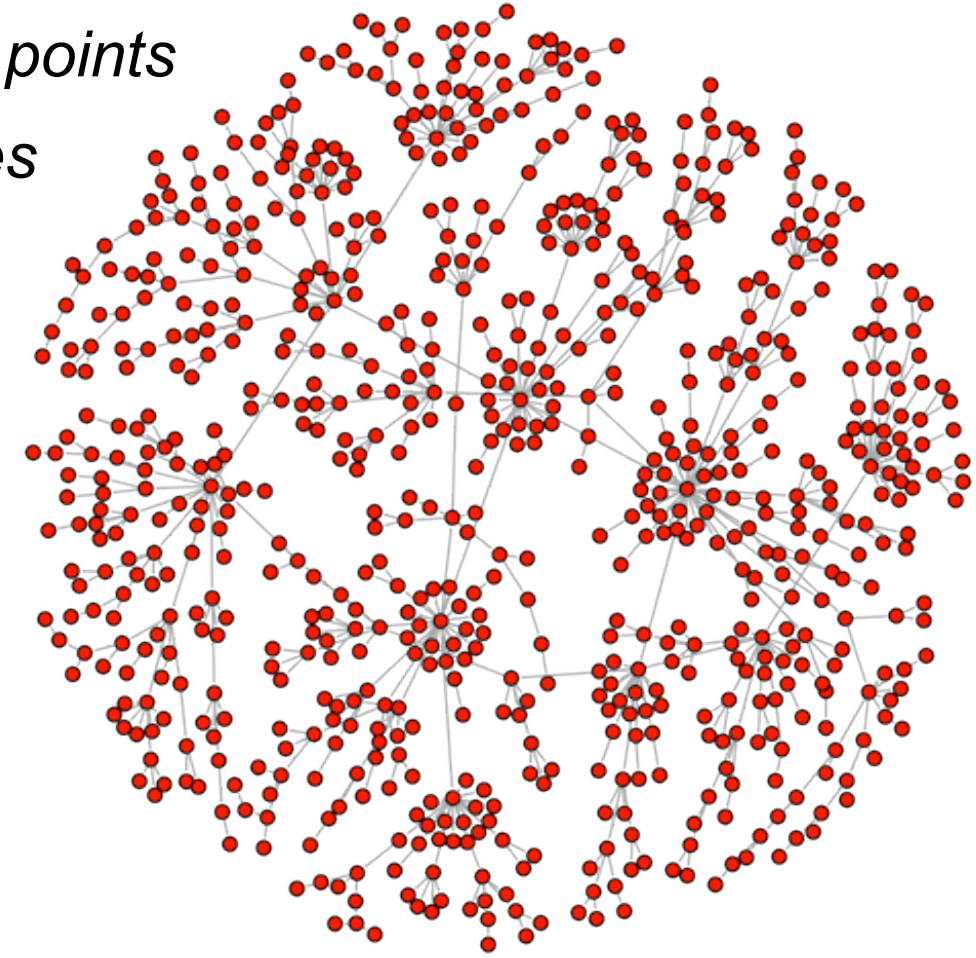


https://commons.wikimedia.org/wiki/File:C_language_linked_list.png

https://en.wikiversity.org/wiki/Data_Structures_and_Algorithms/Arrays,_Lists_and_Vectors

Graph

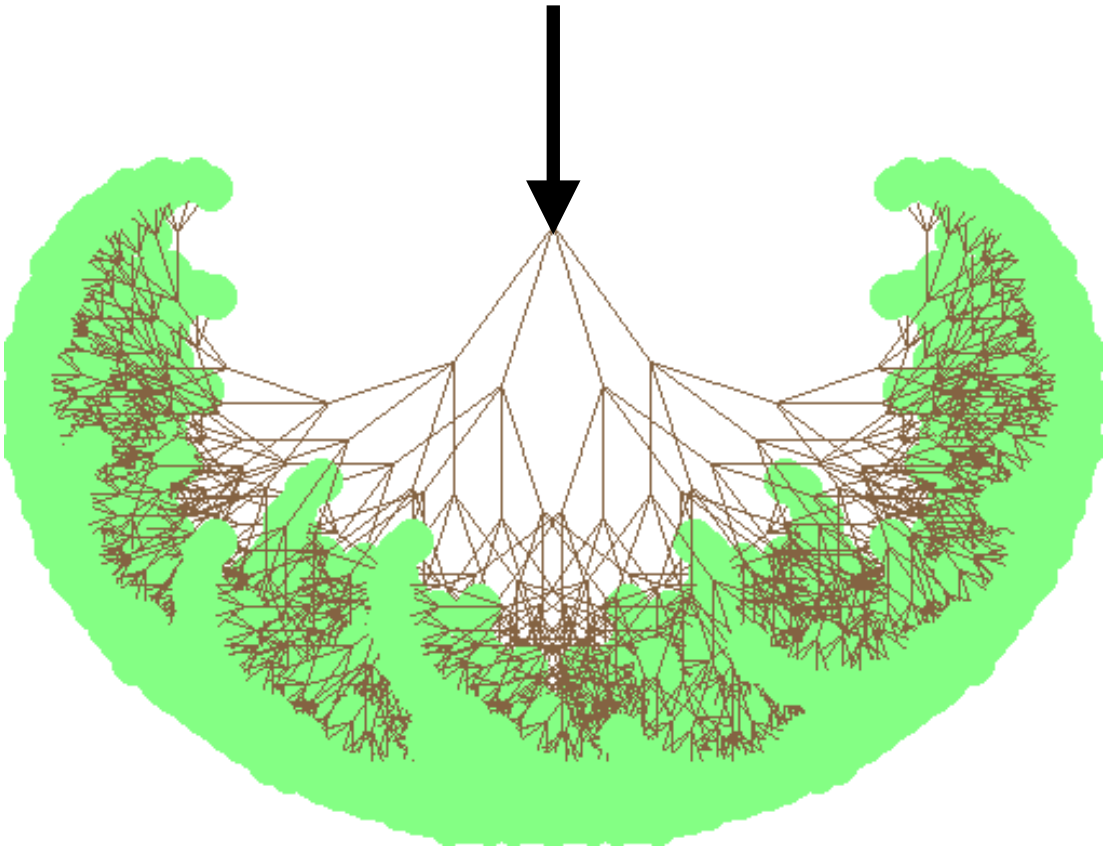
- $G = (V, E)$
- set V of *vertices*, *nodes* or *points*
- set E of *edges*, *arcs* or *lines*
- *e.g. social network, map,*
- *knowledge graph*



http://www.dianliwenmi.com/postimg_8035910_5.html

Tree

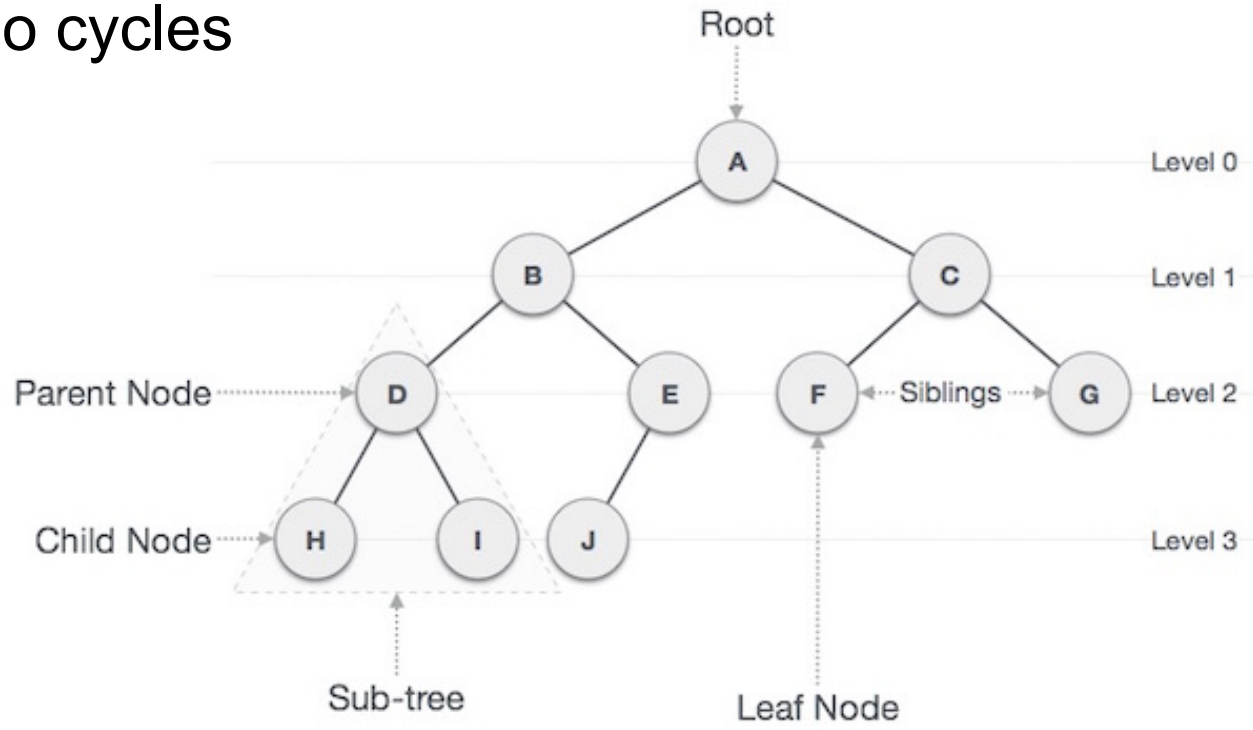
- is a **root**ed graph with **leaves**



http://lodev.org/cgtutor/recursiontrees.html

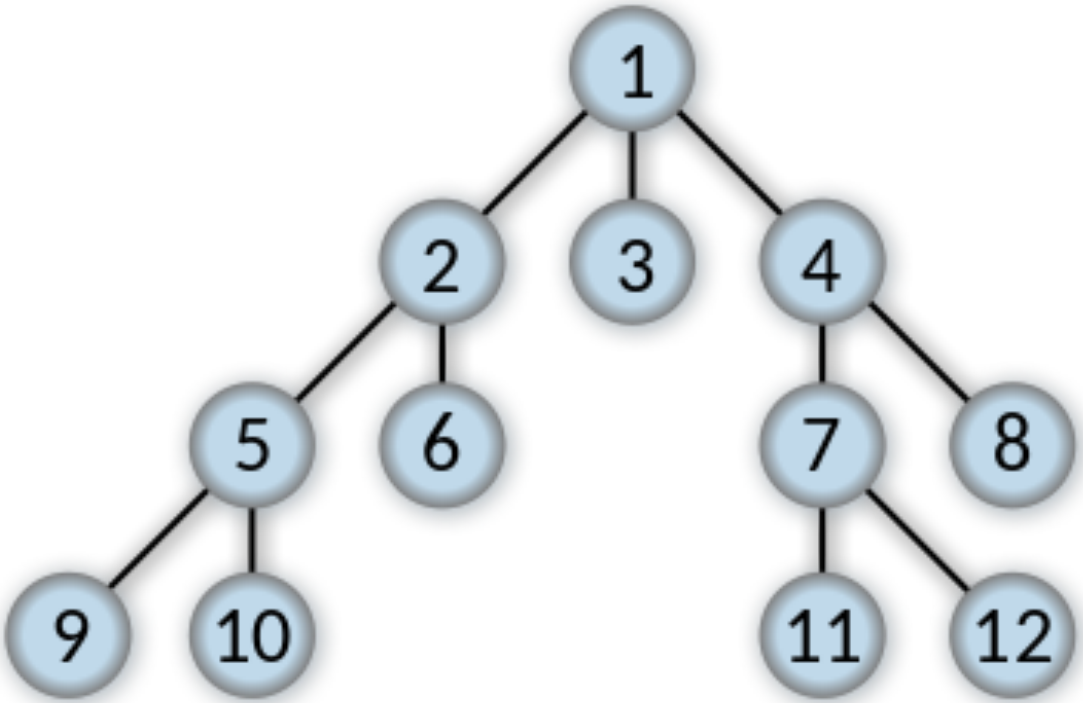
Tree

- every **child** node has only one **parent** node
- there is only a single unique **path** between every 2 nodes (path is a sequence of edges that connect 2 nodes)
- contains no cycles



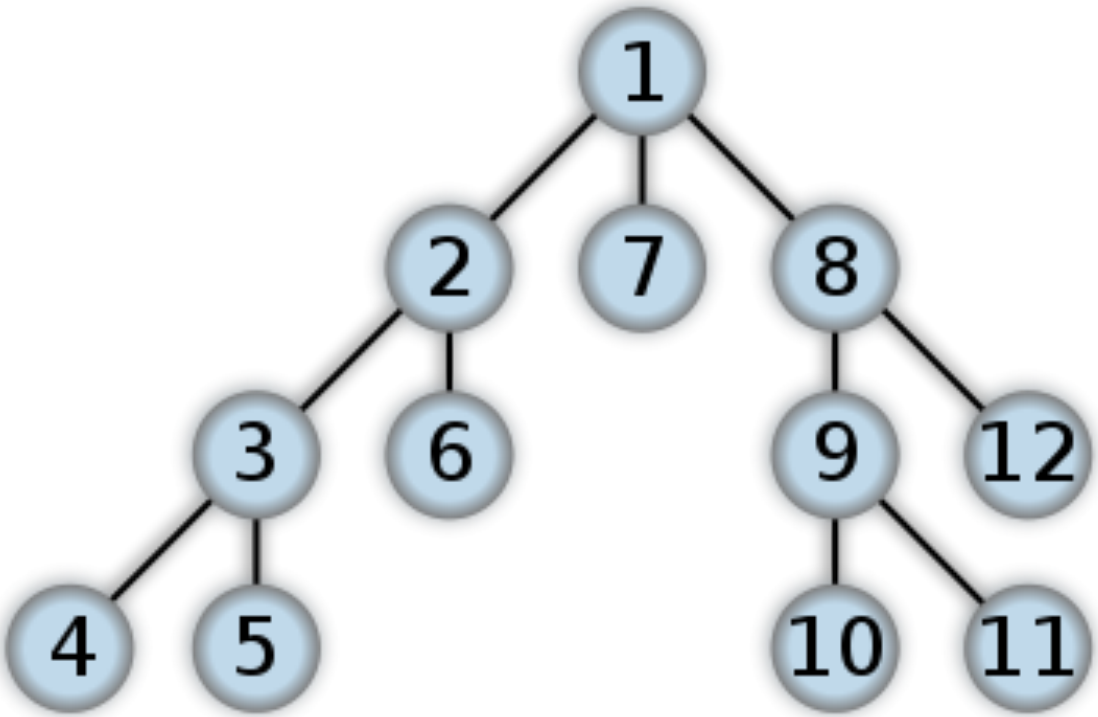
https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm

Breadths-first search (BFS)



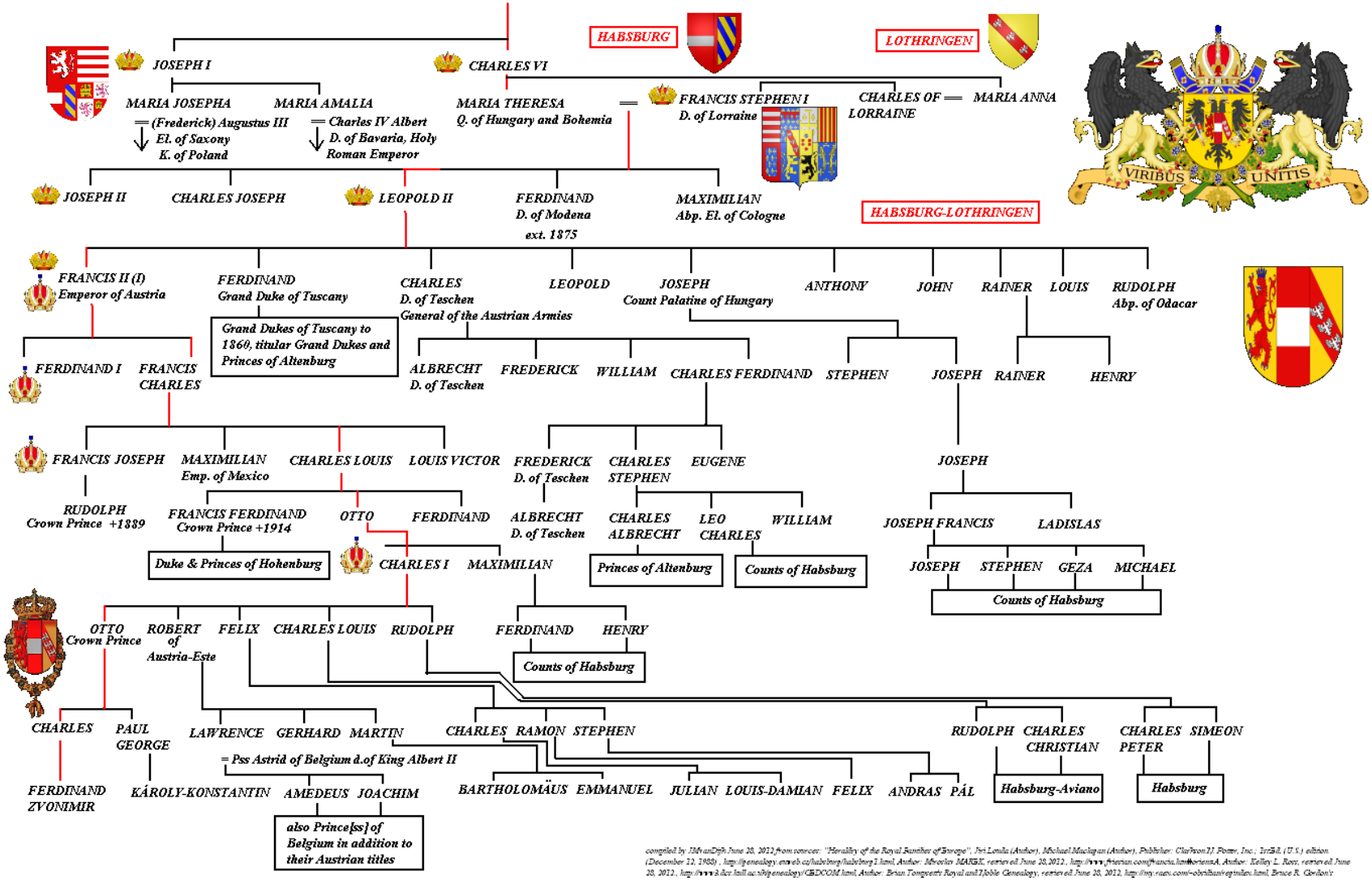
https://en.wikipedia.org/wiki/Breadth-first_search

Depth-first search (DFS)



https://en.wikipedia.org/wiki/Depth-first_search

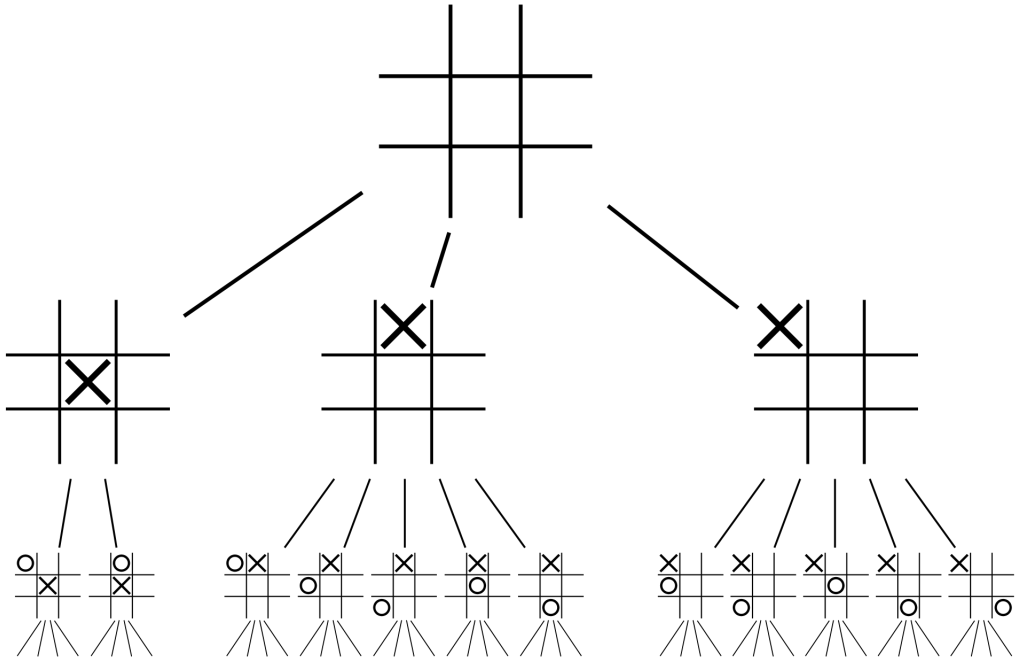
Ex.1: Habsburg Family Tree



https://en.wikipedia.org/wiki/File:Habsburg-Lorraine_Genealogy.PNG

Game tree

- The number of leaf nodes is the number of possible different ways the game can be played
- tic-tac-toe has 255,168 leafs



https://en.wikipedia.org/wiki/Game_tree

AI Games

Task	Year	Human	Machine	Method	Score
Chess	1997	Kasparov	IBM Deep Blue		2:1:3
Poker	2008	Grudzien	University of Alberta: Polaris 2.0	Learning	
Jeopardy	2011	Brad Rutter	IBM Watson	DeepQA	
Image recognition (ImageNet)	2015	A n d r e j Karpathy	Baidu Minwa	Neural network	95.42% : 95%
Go	2016	Lee Sedol	DeepMind: AlphaGo	Deep Learning + Monte Carlo tree search	4:1
Dota 2 1v1 International	2017	Dendi	OpenAI		

https://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

https://openai.com/the-international/

Mechanical Turk

- Wolfgang von Kempelen

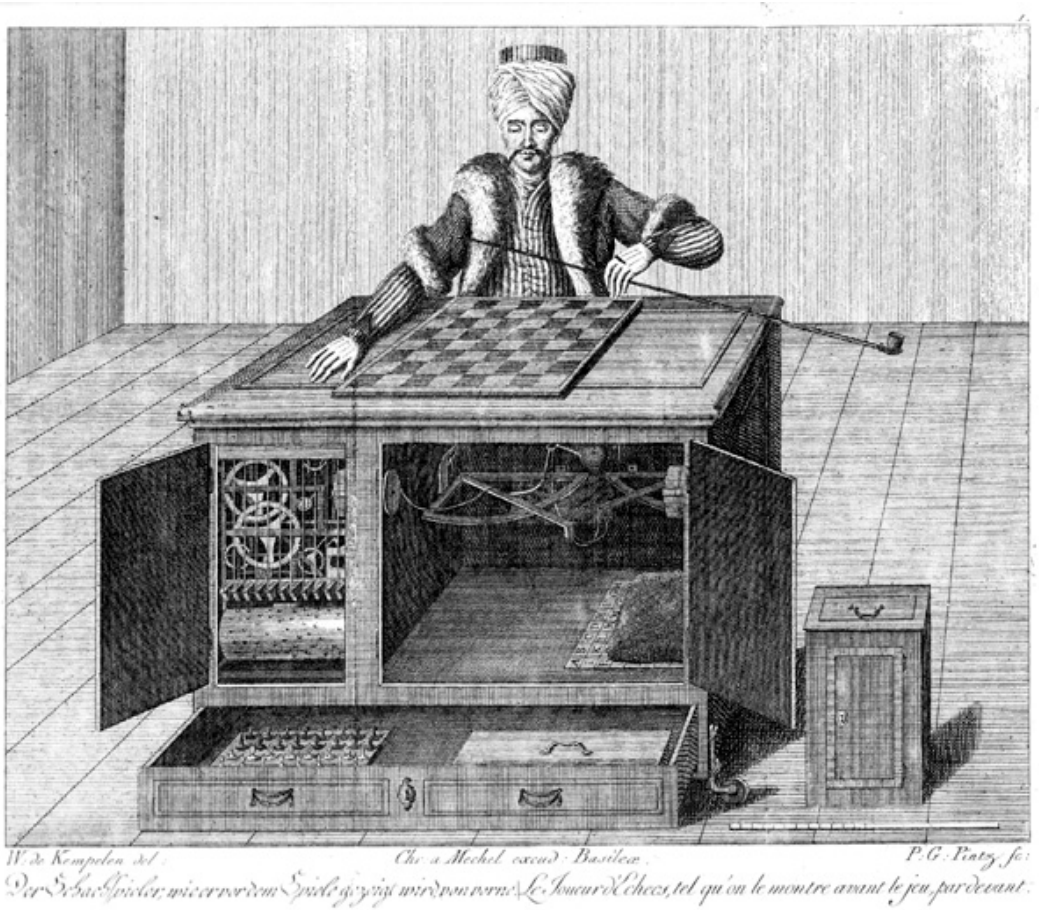
built in **Vienna** in **1770**

for Maria Theresa

- played chess

vs Napoleon Bonaparte

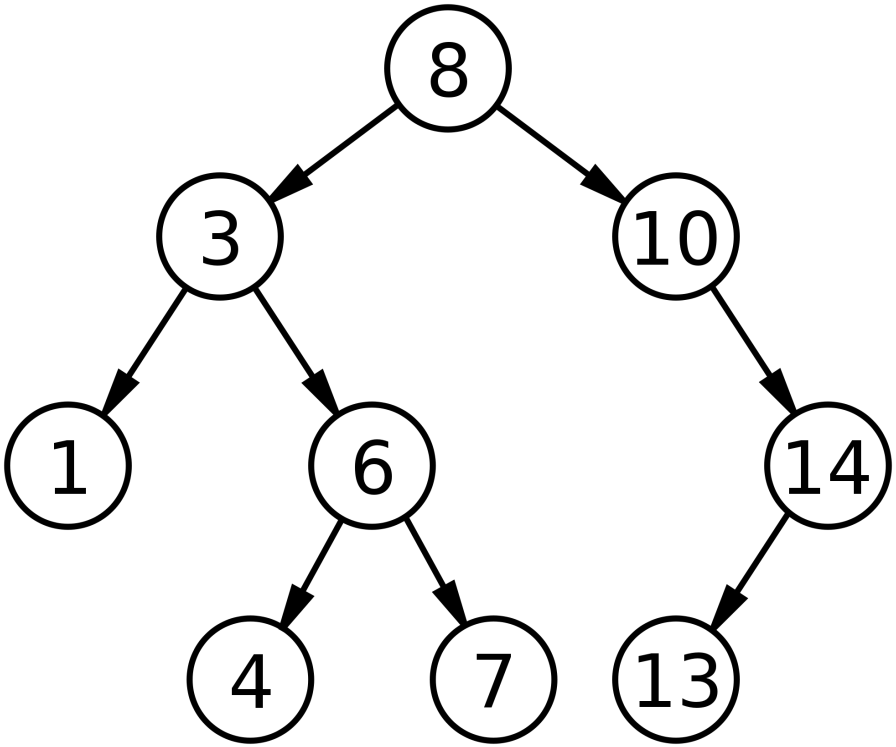
and Benjamin Franklin



https://www.uh.edu/engines/epi2765.htm

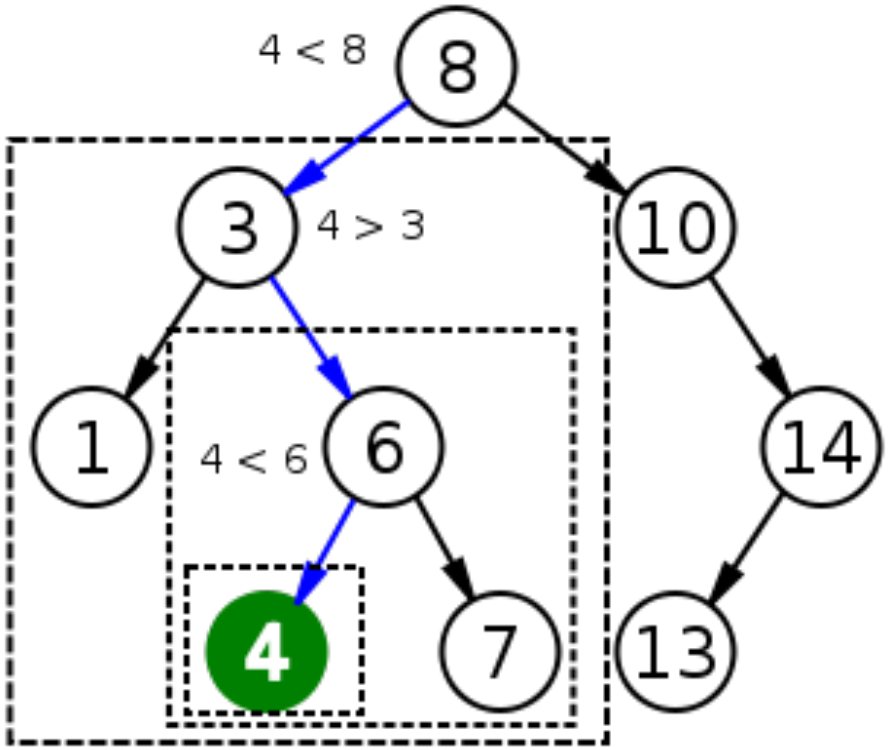
Binary tree

- maximum 2 children (left & right)



https://en.wikipedia.org/wiki/Binary_search_tree

Binary search tree (BST)



https://commons.wikimedia.org/wiki/File:Binary_search_tree_search_4.svg

$h = O(\log n)$

https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html

https://medium.com/the-renaissance-developer/learning-tree-data-structure-27c6bb363051

Binary search algorithm

```
def binary_search(item_list,item):
    first = 0
    last = len(item_list)-1
    found = False
    while( first<=last and not found):
        mid = (first + last)//2
        if item_list[mid] == item :
            found = True
        else:
            if item < item_list[mid]:
                last = mid - 1
            else:
                first = mid + 1
    return found

print(binary_search([1,2,3,5,8], 6))
print(binary_search([1,2,3,5,8], 5))
```

http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBinarySearch.html

https://www.w3resource.com/python-exercises/data-structures-and-algorithms/python-search-and-sorting-exercise-1.php

https://en.wikipedia.org/wiki/Binary_search_algorithm

Recursive search

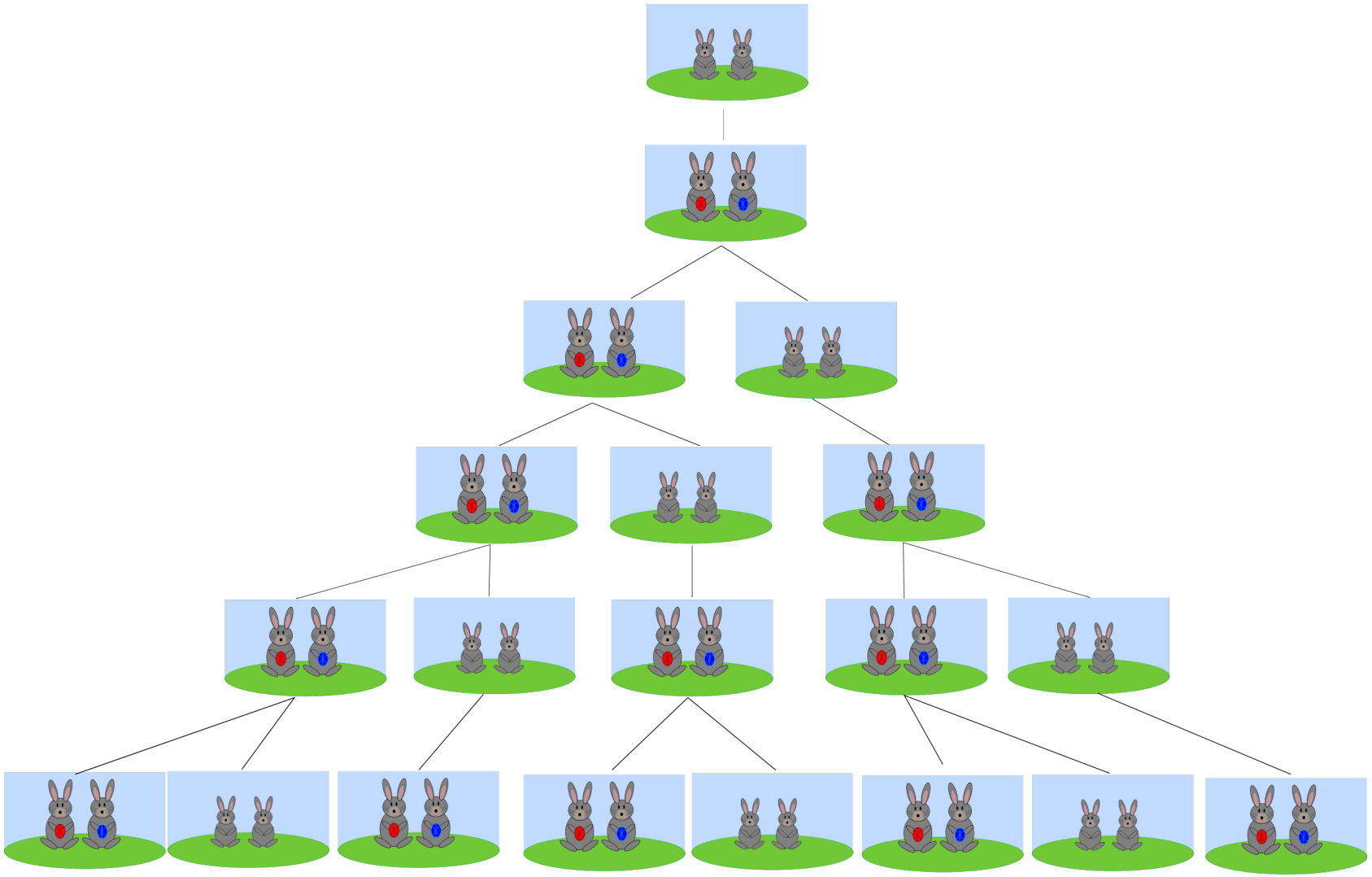
```
def binary_search_recursive(arr, elem, start=0, end=None):
    if end is None:
        end = len(arr) - 1
    if start > end:
        return False

    mid = (start + end) // 2
    if elem == arr[mid]:
        return mid
    if elem < arr[mid]:
        return binary_search_recursive(arr, elem, start, mid-1)
    # elem > arr[mid]
    return binary_search_recursive(arr, elem, mid+1, end)
```

https://stackoverflow.com/questions/19989910/recursion-binary-search-in-python

Recursion

https://xkcd.com/1557/

Ex.2: Fibonacci numbers



https://commons.wikimedia.org/wiki/File:FibonacciRabbit.svg

Sort

- put elements of a list in a certain order
- output is a permutation (reordering with all of the original elements) of the input

https://commons.wikimedia.org/wiki/File:AZ_Sort.png

https://de.wikipedia.org/wiki/Garderobe_(Raum)

https://en.wikipedia.org/wiki/Sorting_algorithm

Insert sort

https://www.pexels.com/photo/game-deck-cards-shuffle-102107/

Insert sort

```
def insertionsort(A):
    for j in range(1,len(A)):
        key = A[j]
        i = j-1
        while (i > -1) and key < A[i]:
            A[i+1]=A[i]
            i=i-1
        A[i+1] = key
    return A
```

https://gist.github.com/basarat/3216903

http://interactivepython.org/courselib/static/pythonds/SortSearch/TheInsertionSort.html

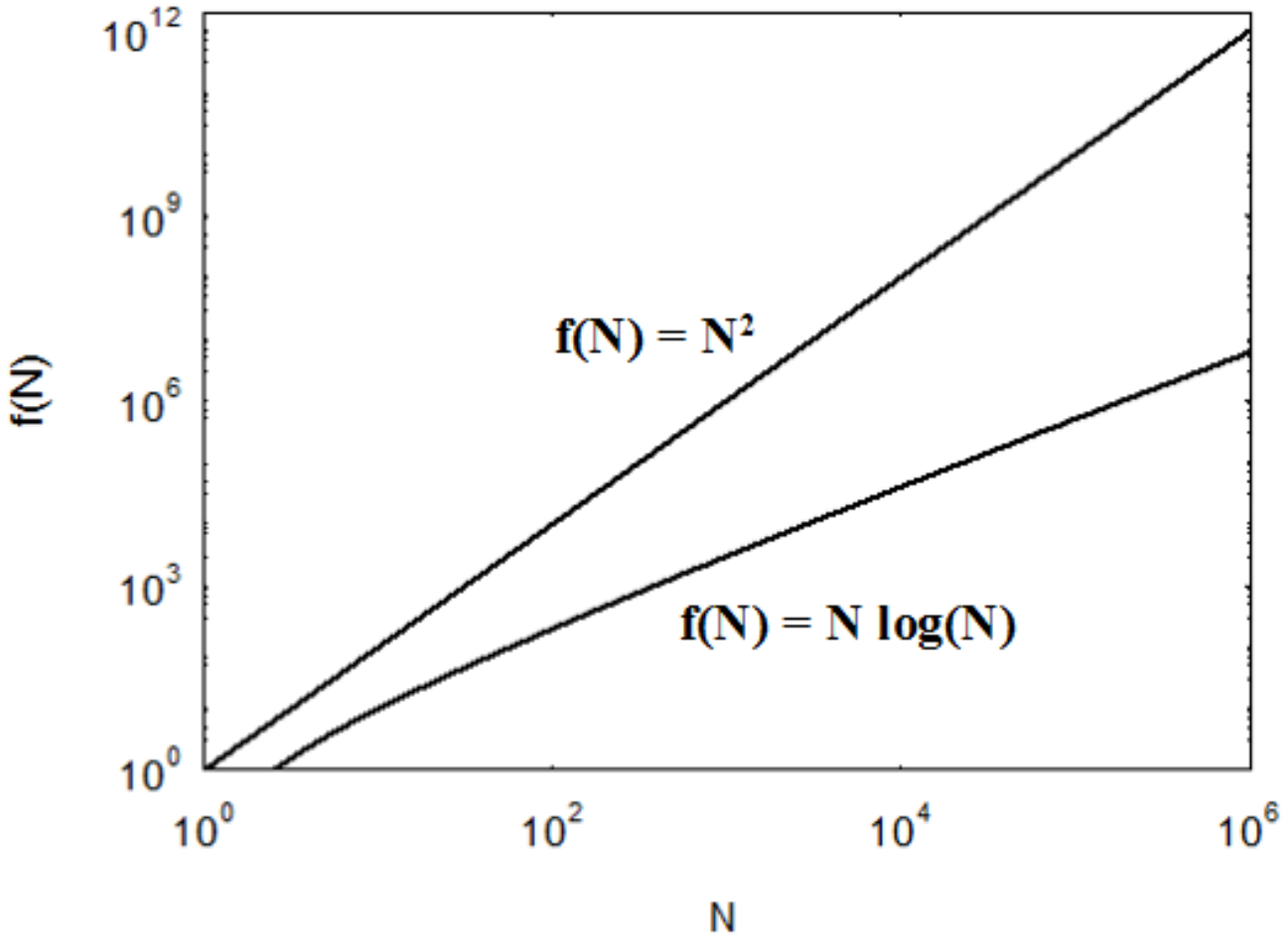
Insert sort

- best case O(*n*): already sorted
- worst&average case O(*n***n*): sorted in reverse order

```
def insertionsort(A):
    for j in range(1,len(A)):
        key = A[j]
        i = j-1
        while (i > -1) and key < A[i]:
            A[i+1]=A[i]
            i=i-1
        A[i+1] = key
    return A
```

Insert sort with Romanian folk dance: https://www.youtube.com/watch?v=ROalU379l3U

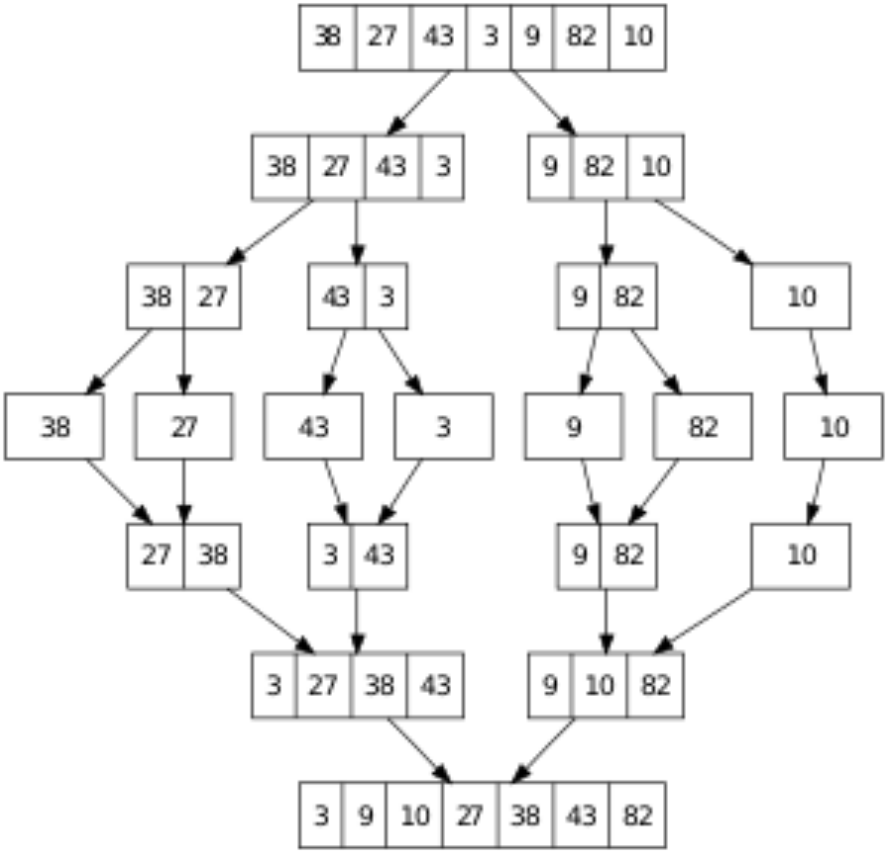
Big O Notation



https://commons.wikimedia.org/wiki/File:N_Körper_System_hierarchische_Algorithmen_Effizienz.png

Merge sort

- divide and conquer technique



https://en.wikipedia.org/wiki/Merge_sort

Merge sort with saxon folk dance: https://www.youtube.com/watch?v=XaqR3G_NVoo

Merge sort

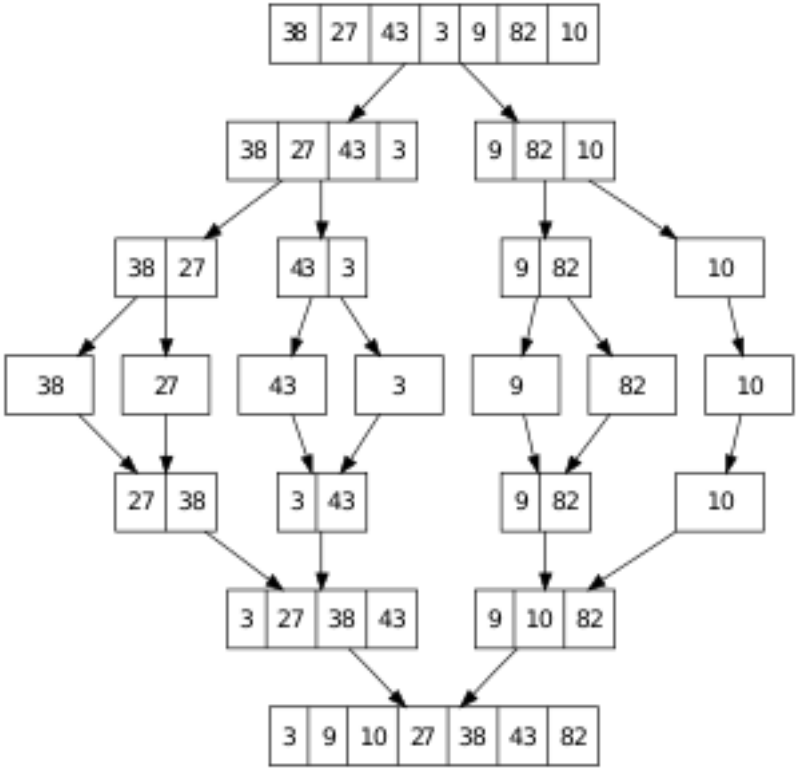
- **recursively** dividing list into smaller sublists which are then sorted

```
def mergesort(list):
    if len(list) < 2:
        return list

    middle = len(list)/2
    left = mergesort(list[:middle])
    right = mergesort(list[middle:])

    return merge(left, right)
```

https://gist.github.com/jvashishtha/2720700

Merge sort

- worst-case running time is O(*n* log *n*)

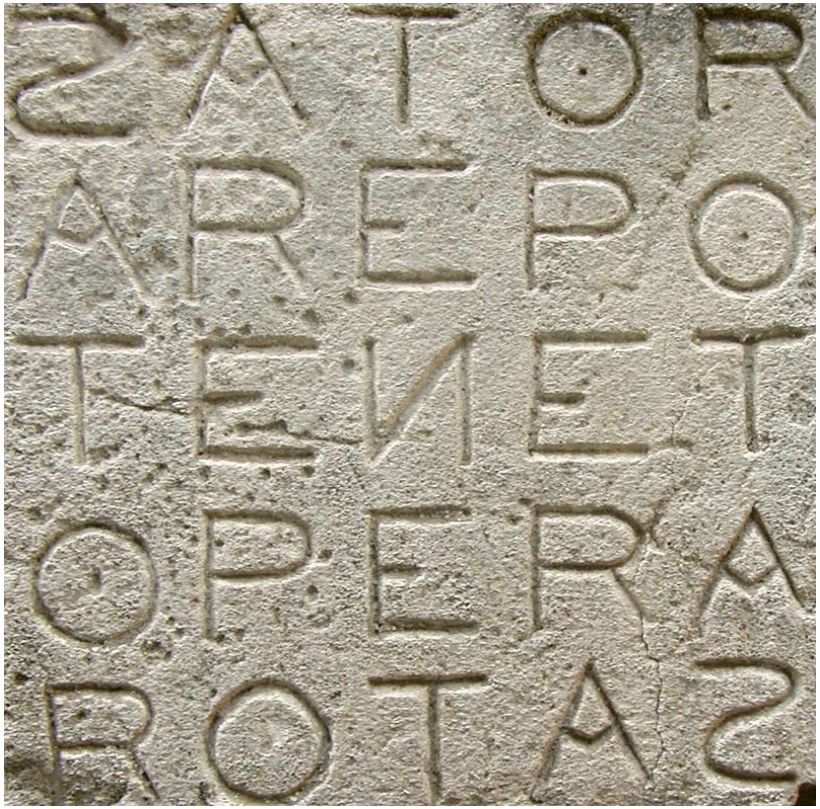


https://en.wikipedia.org/wiki/Merge_sort

http://interactivepython.org/courselib/static/pythonds/SortSearch/TheMergeSort.html

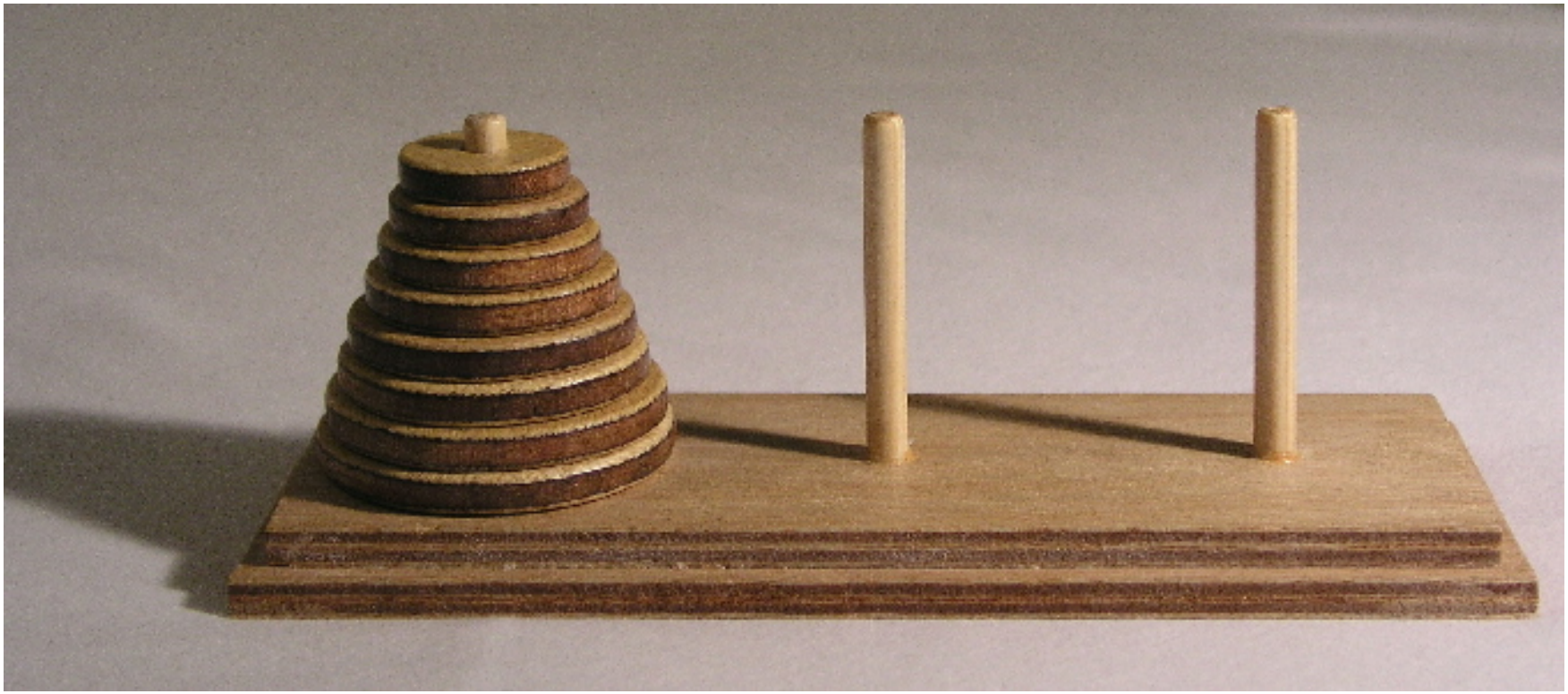
Ex.3: Palindrome

- Sator Square: "Sator Arepo Tenet Opera Rotas"
- "The sower Arepo holds with effort the wheels"



https://en.wikipedia.org/wiki/Palindrome

Ex.4: Tower of Hanoi

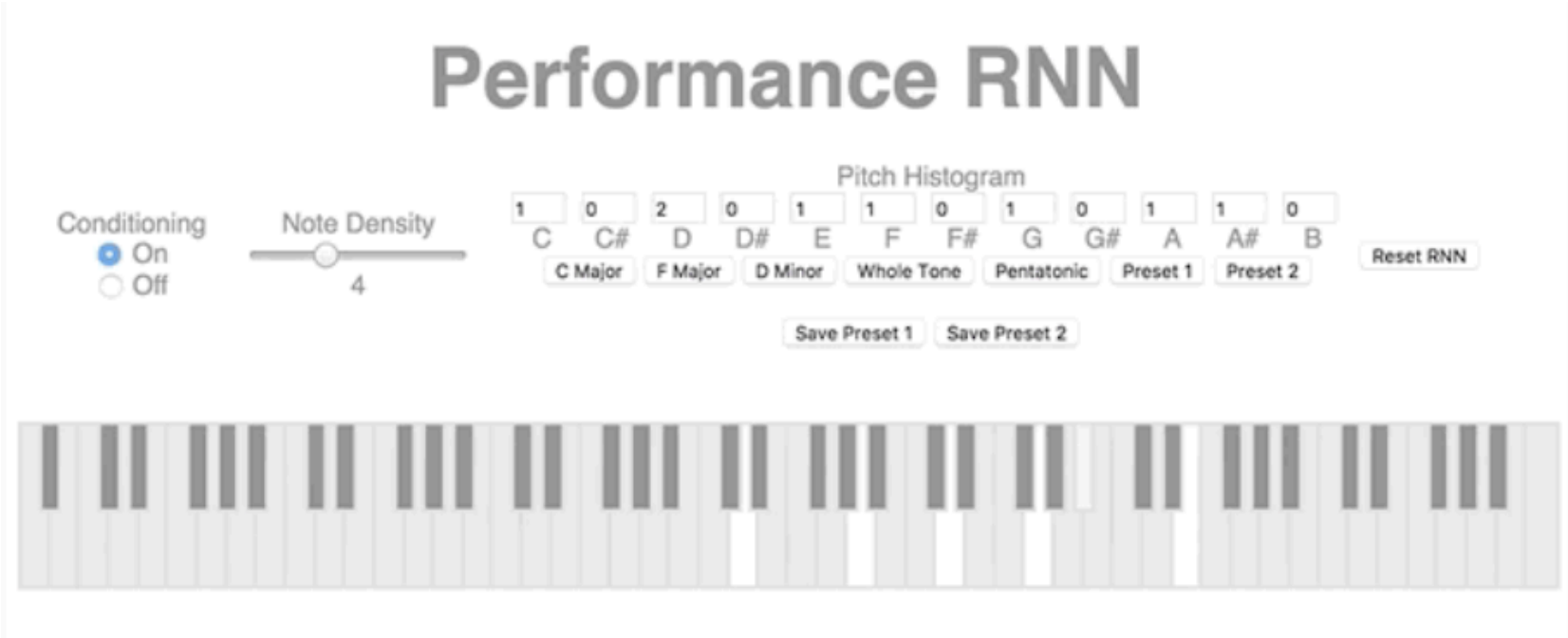


https://commons.wikimedia.org/wiki/File:Tower_of_Hanoi.jpeg

Machine Learning

- model trained on ~1400 performances by skilled pianists

https://deeplearnjs.org/demos/performance_rnn



https://magenta.tensorflow.org/performance-rnn-browser

https://github.com/AlexiaJM/Deep-learning-with-cats

https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Thank you!

svitlana.vakulenko@wu.ac.at