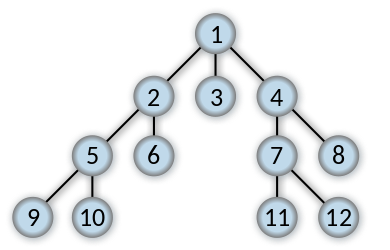
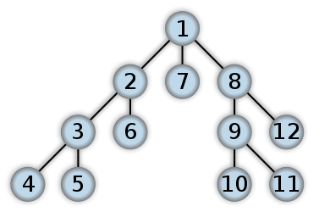
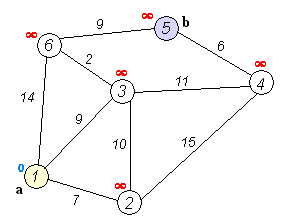
Introduction presentation:

* Final grade: progress 33, final presentation 33, code 33
* Progress earn it: make linear progres in the subject
* 3 x algorithms
* Constructive algorithms iterative algorithms, population based algorithms
* Git college turingzaal CWI
* Excl. Final presentation 3x presentations

Algorirthms:

* Breadth-first
  + **Breadth-first search** (**BFS**) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for traversing or searching [tree](https://en.wikipedia.org/wiki/Tree_data_structure) or [graph](https://en.wikipedia.org/wiki/Graph_(data_structure)) data structures. It starts at the [tree root](https://en.wikipedia.org/wiki/Tree_(data_structure)#Terminology) (or some arbitrary node of a graph, sometimes referred to as a 'search key'[[1]](https://en.wikipedia.org/wiki/Breadth-first_search#cite_note-1)) and explores the neighbor nodes first, before moving to the next level neighbours.
  + 
* Depth-first
  + **Depth-first search** (**DFS**) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for traversing or searching [tree](https://en.wikipedia.org/wiki/Tree_data_structure) or [graph](https://en.wikipedia.org/wiki/Graph_(data_structure)) data structures. One starts at the [root](https://en.wikipedia.org/wiki/Tree_(data_structure)#Terminology) (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before [backtracking](https://en.wikipedia.org/wiki/Backtracking).
  + 
* Uniform cost -> dijkstra(?)
  + **Dijkstra's algorithm** is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) between [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)), which may represent, for example, road networks. It was conceived by [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist) [Edsger W. Dijkstra](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) in 1956 and published three years later.
  + 
* A\* branch-n-bound
  + **Branch and bound** (**BB**, **B&B**, or **BnB**) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) design paradigm for [discrete](https://en.wikipedia.org/wiki/Discrete_optimization) and [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) problems, as well as [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization). A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of [state space search](https://en.wikipedia.org/wiki/State_space_search): the set of candidate solutions is thought of as forming a [rooted tree](https://en.wikipedia.org/wiki/Tree_(graph_theory)) with the full set at the root. The algorithm explores *branches* of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated *bounds* on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.
* Greedy
  + We know this one already
* Combi
  + -wtf-? <http://www.gmdh.net/GMDH_com.htm>
  + This is the basic GMDH algorithm. It use an input data sample as a matrix containing *N* levels (points) of observations over a set of *M* variables. A data sample is divided into two parts. If regularity [criterion](http://www.gmdh.net/GMDH_var.htm#var) *AR(s)* is used, then approximately two-thirds of observations make up the training subsample *NA*, and the remaining part of observations (e.g. every third point with same variance) form the test subsample *NB* . The training subsample is used to derive estimates for the coefficients of the polynomial, and the test subsample is used to choose the structure of the optimal model, that is one for which *the regularity criterion AR(s)* takes on a minimal value:
* hill climber
  + In numerical analysis, **hill climbing** is a [mathematical optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)) technique which belongs to the family of [local search](https://en.wikipedia.org/wiki/Local_search_(optimization)). It is an [iterative algorithm](https://en.wikipedia.org/wiki/Iterative_algorithm) that starts with an arbitrary solution to a problem, then attempts to find a better solution by [incrementally](https://en.wikipedia.org/wiki/Incremental_heuristic_search) changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.
* simulated anneualing
  + **Simulated annealing** (**SA**) is a [probabilistic](https://en.wikipedia.org/wiki/Probabilistic_algorithm) technique for approximating the [global optimum](https://en.wikipedia.org/wiki/Global_optimum) of a given [function](https://en.wikipedia.org/wiki/Function_(mathematics)). Specifically, it is a [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic) to approximate [global optimization](https://en.wikipedia.org/wiki/Global_optimization) in a large [search space](https://en.wikipedia.org/wiki/Mathematical_optimization#Optimization_problems). It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent).
* particle swarms
  + In [computer science](https://en.wikipedia.org/wiki/Computer_science), **particle swarm optimization** (**PSO**) is a computational method that [optimizes](https://en.wikipedia.org/wiki/Mathematical_optimization) a problem by [iteratively](https://en.wikipedia.org/wiki/Iterative_method) trying to improve a [candidate solution](https://en.wikipedia.org/wiki/Candidate_solution) with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed [particles](https://en.wikipedia.org/wiki/Point_particle), and moving these particles around in the [search-space](https://en.wikipedia.org/wiki/Optimization_(mathematics)#Concepts_and_notation) according to simple [mathematical formulae](https://en.wikipedia.org/wiki/Formula) over the particle's [position](https://en.wikipedia.org/wiki/Position_(vector)) and [velocity](https://en.wikipedia.org/wiki/Velocity). Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solution
* genetic algorithms
  + In [computer science](https://en.wikipedia.org/wiki/Computer_science) and [operations research](https://en.wikipedia.org/wiki/Operations_research), a **genetic algorithm** (**GA**) is a [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic) inspired by the process of [natural selection](https://en.wikipedia.org/wiki/Natural_selection) that belongs to the larger class of [evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm) (EA). Genetic algorithms are commonly used to generate high-quality solutions to [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)) and [search problems](https://en.wikipedia.org/wiki/Search_algorithm) by relying on bio-inspired operators such as [mutation](https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)), [crossover](https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)) and [selection](https://en.wikipedia.org/wiki/Selection_(genetic_algorithm)).[[1]](https://en.wikipedia.org/wiki/Genetic_algorithm#cite_note-FOOTNOTEMitchell19962-1)
* plant propagation
  + w-t-f?
* fireworks
  + In general the Fireworks Algorithm (FWA) is an approach to exploring a very large solution space by choosing a set of random points confined by some distance metric in the hopes that one or more of them will yield promising results, allowing for a more concentrated search nearby.
* computationele complexiteit
* csp
* fop
* cop
* constraint relaxation

Hoorcollege2:

3 types problemen

* constraint satisfaction problem (CSP)
  + oplossingen moeten voldoen aan constraints
    - sudoku: 1 tm 9 in rij, kolom, hok
  + alle oplossingen even goed
    - doorgaans 1 opl.
* Free optimization problem (FOP)
  + Oplossing: zo goed mogelijk
  + Objective function (scorefunctie, fitnessfunctie)
  + Geen constraints aan variabelen
  + Fitten formule: y=ax+b a, b -> moet kleinste afst. van alle punten zijn (optimization criteria)
* Constrained optimization problem (COP)
  + Oplossing zo goed mogelijk
  + Objective function (scorefunctie, fitnessfunctie)
  + OOK constraint aan variabelen (als treintijd <2u moet zijn en het is >2u dan niet eens kijken naar opl)
  + Traveling salesman problem -> lijntjes trekken langs puntjes 0, 1, 2 etc. Constraint: elk puntje 1x, closed loop, objective function: route zo kort mogelijk

Nrplaat

[16 85 JS]

10 \* 10 \* 10 \* 10 \* 26 \* 26 = heul veul -> Toestantsruimte (statespace) ANALYSEREN ALS MOGELIJK

Schuifpuzzel [4 \* 3] : niet alle mogelijkheden mogelijk (12!) -> de helft.

2 schakelaars

2 \* 2, worst case 4. Altijd vanuit gaan.

5 schakelaars

2^5-1

N schakelaars, 2^n: bijna altijd vreselijk! Indicates how fast the state space is growing

Schaakbord 8\*8:

Hoeveel koningen niet slaan: niet meer dan 8. Pidgeonhole

64 boven 8

n!/ (n!(k-n)!)

in log log, exponentieel grafiek. Erger dan erger

64! / (8!(64-8)!)

Je kan wel veel snoeien, bijv 2 koningen op zelfde lijn. PRUNING

Sneller maken: doe exponentieel

**Few good man kijken**

Toestantsruimte sudoku (25 lege vakjes): niet groter dan: 9^25 (UPPER BOUND). Te complex om te berekenen. Upper bound en lower bound belangrijk.

Traveling salesman problem:

Statespace: 10!

Objective function: afst. zo kort mogelijk

Lower bound: op zn minst kortste afst. \* 10. k1, k2, k3 min en max score

EINDPRESENTATIE: BOUNDS BEREKENEN

OPLOSSINGSDICHTHEID: alle 8 puzzelstukjes hetzelfde: 1

: niet hetzelfde: 1/8!

**Puzzel: 2mil: eternity II**

Gebrek aan keuze maakt problemen makkelijekr

Hillclimbing

Convexe functie 1 plek alle afgeleiden nul

Non-convexe: Veel lokale minima

## Process points!

Process points: aan het begin 0, door het vak heen meer.

Voorstel indienen woordelijk bij tech assistent. Waarom ik dat vind bij idea punten. (Infrastructuur, datastructuur (iedereen begrijpen), experimenteren, algoritmes). alle drie aanwezig

Punten bouwen op.

### College 3 constructieve algoritmes

Traveling salesman problem: stap voor stap/ Randomizer kan ook

Stack: last in first out (lifo) dienbladen

Geen random in een depth first

Depth first vs breadth first is eerst breedte/diepte in de boom

Pruning wil je hebben

Backtracking van achter naar voren door een boomstructuur heen (breadth first of depth frist)

Archief: bijv bij schuifpuzzel depthfirst dan alle gedane toestanden er in opslaan zodat je die weer kan prunen

Als laatste opl. In de statespace is, dan is statespace in t archief

Benchmark net niet normaal verdeeld