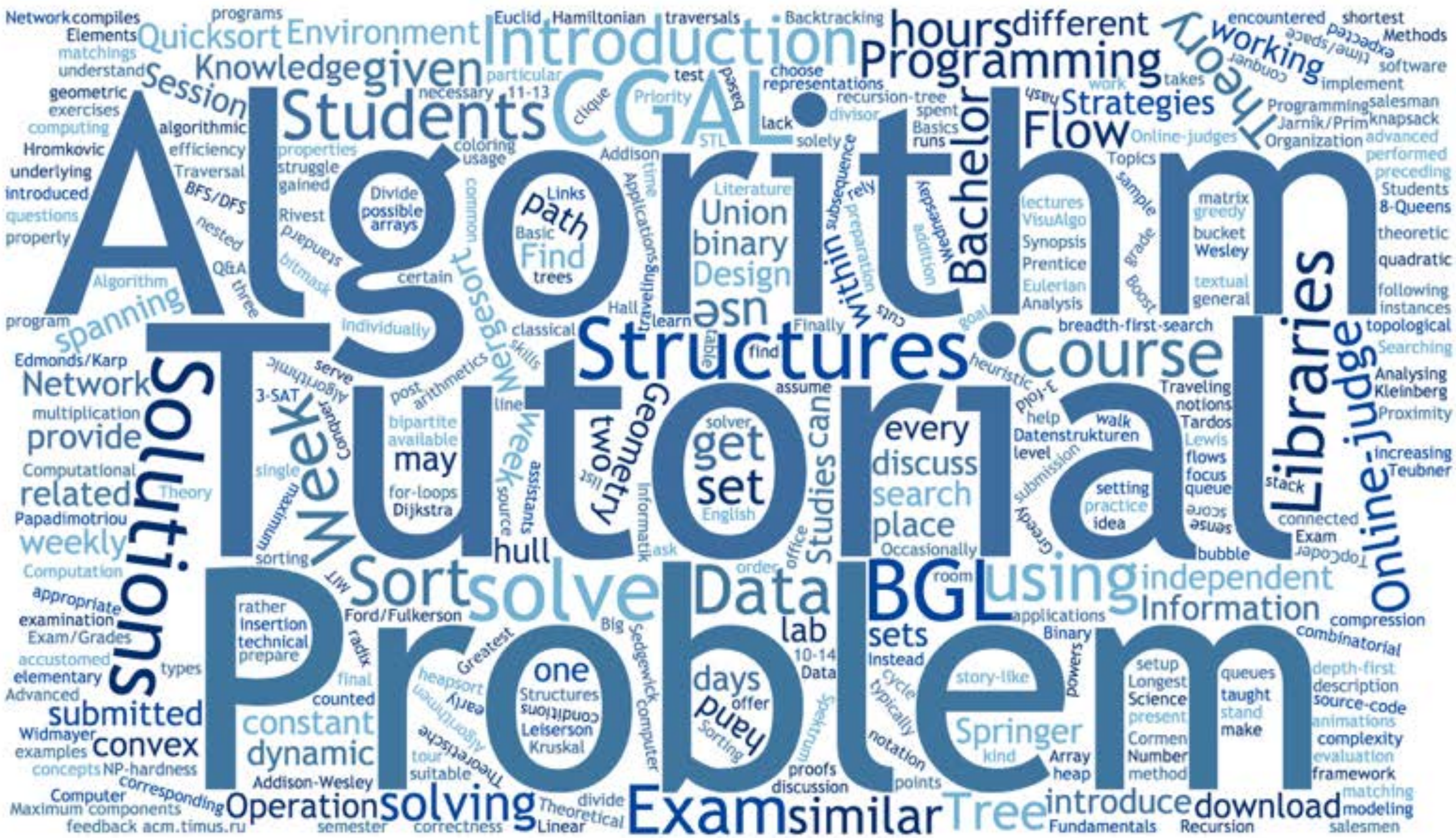
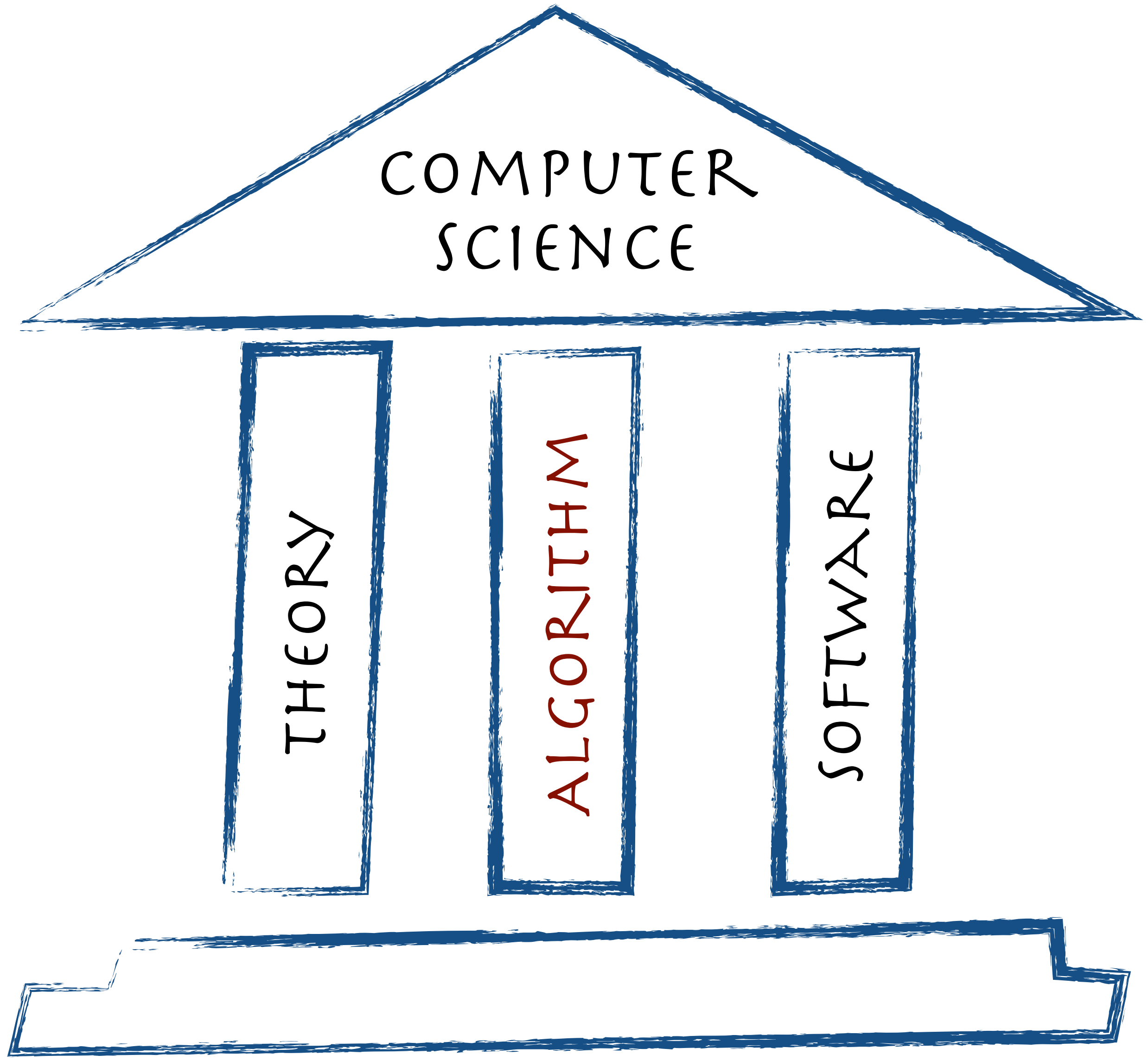
Algorithms Lab



Angelika Steger, Emo Welzl, Peter Widmayer
Luis Barba, Hafsteinn Einarsson, Daniel Graf, Michael Hoffmann,
Petar Ivanov, Miloš Trujić, Felix Weissenberger, Manuel Wettstein



A computer scientist is able to combine concepts and techniques from different areas and apply them to solve problems in various application domains.

Applying Knowledge is difficult 🙁

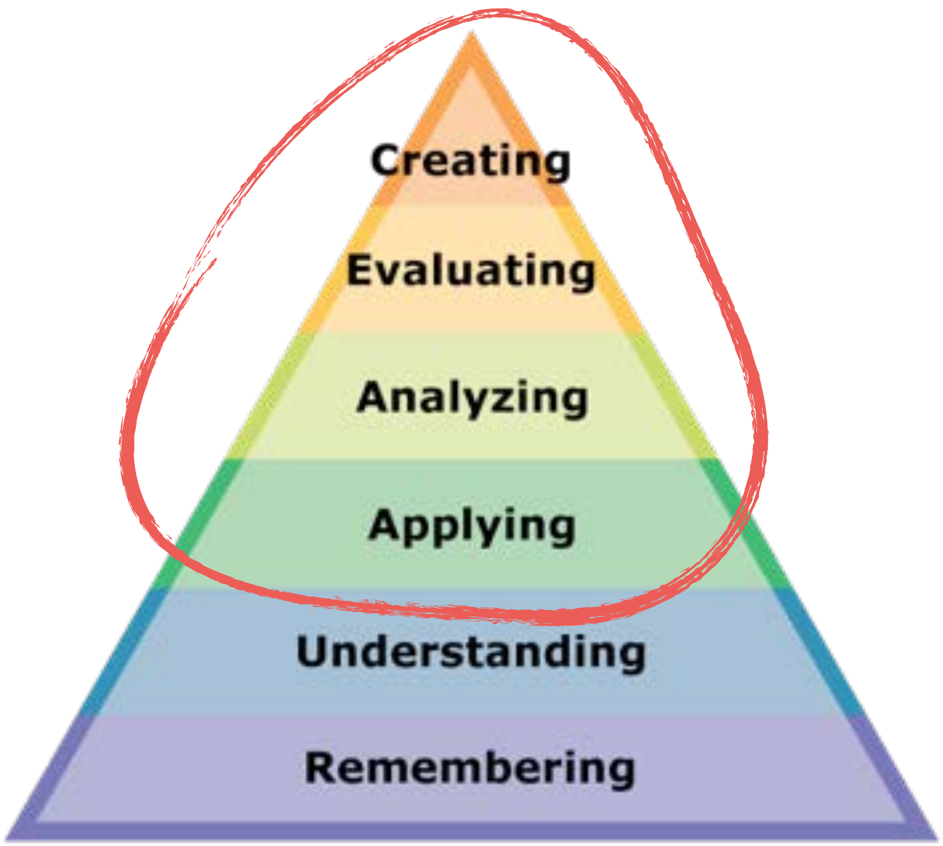
Typical coursework: Can you solve X using Y? (knowing it works)

Want: Do you know any way to solve X? (not knowing if possible)

Here: Can you solve X using a subset of $Y_1,\ldots,Y_k$?

Problem: There is no general algorithm to design algorithms.

→ Practice over and over on concrete examples.



Taxonomy of learning objectives acc. to Bloom/Anderson

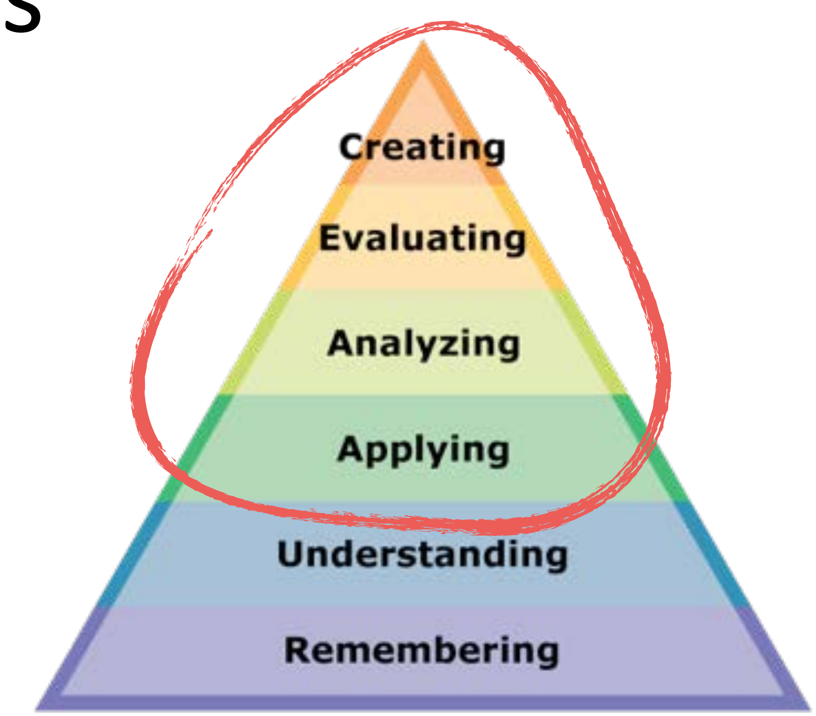
[Picture: http://www.learnnc.org/lp/pages/4719]

Applying Knowledge is difficult 😅

→ Practice over and over on concrete examples.

Goals:

- develop intuition/experience in algorithm design: which techniques work for which problem?
- learn about some standard tools/ libraries for algorithm implementation
- develop secondary skills: creativity, problem solving, time management, self-assessment, implementation, testing, …



Taxonomy of learning objectives acc. to Bloom/Anderson
[Picture: http://www.learnnc.org/lp/pages/4719]

Objective: Master of Algorithms

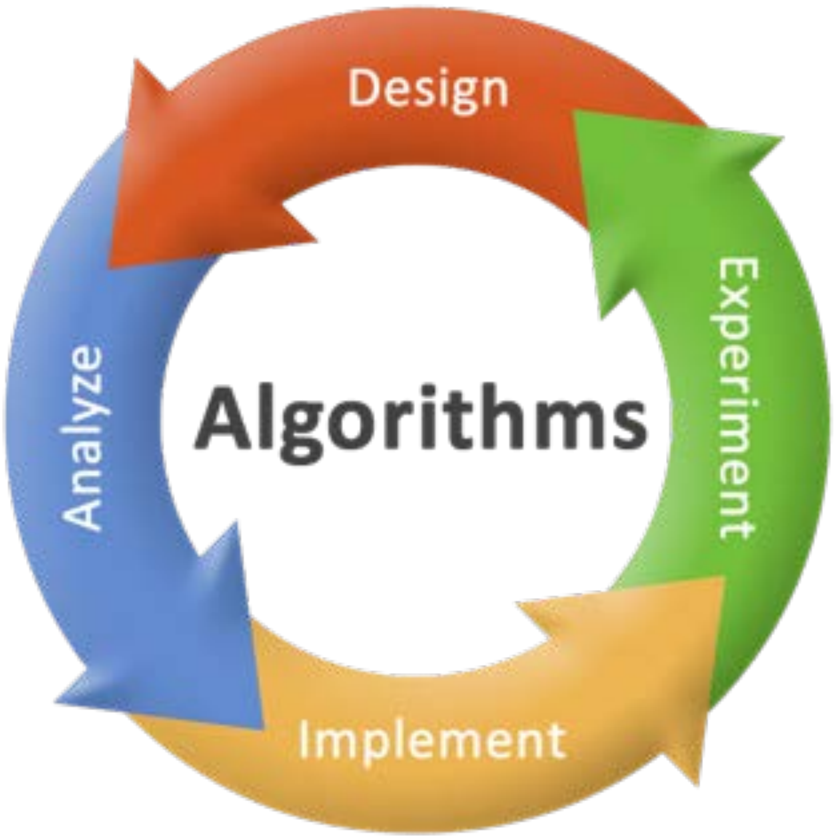
Design efficient algorithms for real-world problems.

In a toy world...

Problems are posed in form of a story.

Task:

- find an appropriate model
- design a suitable algorithm to solve it efficiently
- implement and test the algorithm on given data



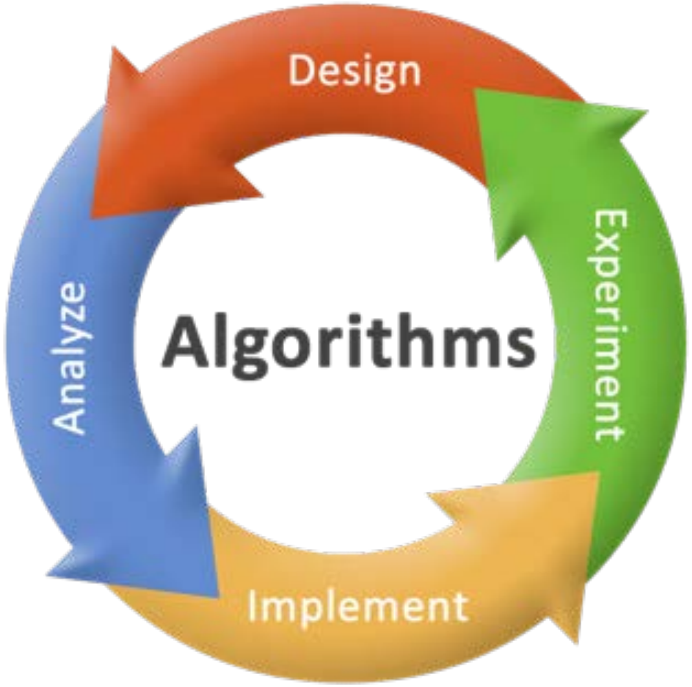
Time: 2h / problem

[Picture from http://openclassroom.stanford.edu/MainFolder/courses/IntroToAlgorithms/cs161logo.png]

Objective: Master of Algorithms

Usually the main challenge is here:

- find an appropriate model
- design a suitable algorithm to solve it efficiently



No need to reinvent the wheel => use libraries for standard data structures & algorithms.

All problems can be solved with ≤ 100 lines of (well formatted) C++ code.

[Picture from http://openclassroom.stanford.edu/MainFolder/courses/IntroToAlgorithms/cs161logo.png]

Example Problem

Story

Exercise – *Clues*

Holmes and Watson are out on the streets to keep an eye on various people and places, hoping to obtain clues regarding criminal activities. Each of them carries a radio set and they want to coordinate through it as soon as something interesting happens. As the area of interest is quite large, Holmes and Watson cannot maintain a direct connection throughout the investigation. Instead they setup a network of radio stations to route the communication. In the following we use the term *clients* to refer to radio sets and radio stations collectively. All clients have the same operation range r so that they can communicate with every client in distance at most r.

For the actual communication there are four different frequencies available. Any client can receive on all frequencies. But in order to avoid interferences, any two clients that are in range of each other must send on different frequencies. Each of the two radio sets has one exclusive sending frequency assigned to it. This leaves two frequencies for the stations to work with. To keep the protocol simple, Holmes wants to assign one fixed frequency to each station so that the station sends on this frequency only. Is it possible to achieve such an assignment without generating any interferences? If so, which collections of clues can be routed within this network between Holmes and Watson?

The radio sets are "intelligent" in the sense that they automatically select the client to connect to. If the other radio set is in range, they connect to the other radio set. Otherwise—if any station is in range—they select the station with the strongest signal (closest client) to connect to. You may assume that this station is unique in all test instances.

Input The first line of the input contains the number $t \leq 30$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains three integers `n m r`, separated by a space and such that $1 \leq n, m \leq 9 \cdot 10^4$ and $0 < r < 2^{24}$. Here n denotes the number of stations, m denotes the number of clues, and r denotes the operation range of the clients.
- The following n lines define the positions $s_0, \ldots, s_{n-1}$ of the stations. You may assume that these positions are pairwise distinct.
- The final m lines define the m clues. Each clue is defined by two positions a_i and b_i, for $i \in \{0, \ldots, m-1\}$, where a_i describes the position of Holmes and b_i describes the position of Watson at the moment when this clue is obtained.

Each position is described by two integer coordinates `x y`, separated by a space and such that $|x|, |y| < 2^{24}$.

Output For each test case output a line with one character "y" or "n" per clue, that is, a string $c_0 c_1 \cdots c_{m-1}$ of m characters. For each $i \in \{0, \ldots, m-1\}$, the character c_i is "y", if and only if clue i can be routed within this network, as defined in the next paragraph.

Denote the set of stations by $S = \{s_0, \ldots, s_{n-1}\}$. A *network without interferences* on S corresponds to a map $f : S \to \{0, 1\}$ such that $f(u) \neq f(v)$, for all u, v with $||u - v|| \leq r$. A clue can be *routed* from a_i to b_i, if there exists a network without interferences on S and there exist $k \in \mathbb{N}$ and a sequence $t_0, \ldots, t_k$, such that

(1) $t_0 = a_i$, $t_k = b_i$, and $t_j \in S$, for $j \in \{1, \ldots, k-1\}$;

(2) $||t_j - t_{j-1}|| \leq r$, for all $j \in \{1, \ldots, k\}$;

(3) If $t_1 \neq b_i$ (and, thus, $t_{k-1} \neq a_i$), then t_1 is the (unique) client from S that is closest to a_i and t_{k-1} is the (unique) client from S that is closest to b_i.

Points There are four groups of test sets, worth 100 points in total.

1. For the first group of test sets, worth 20 points, you may assume that $n \leq 5'000$, $m = 1$, and $a_0 = b_0$. (Effectively, this reduces to the question of whether or not there exists a network without interferences.)
2. For the second group of test sets, worth 30 points, you may assume that there exists a network without interferences on S and $m \leq 20$.
3. For the third group of test sets, worth 30 points, you may assume that there exists a network without interferences on S.
4. For the fourth group of test sets, worth 20 points, there are no additional assumptions.

Corresponding sample test sets are contained in `testi.in/out`, for $i \in \{1, 2, 3, 4\}$.

Sample Input

```
2
2 3 2
0 0
2 0
-2 0 3 0
-2 1 2 -1
0 1 -1 2
3 1 2
0 0
2 0
1 1
3 0 1 1
```

Sample Output

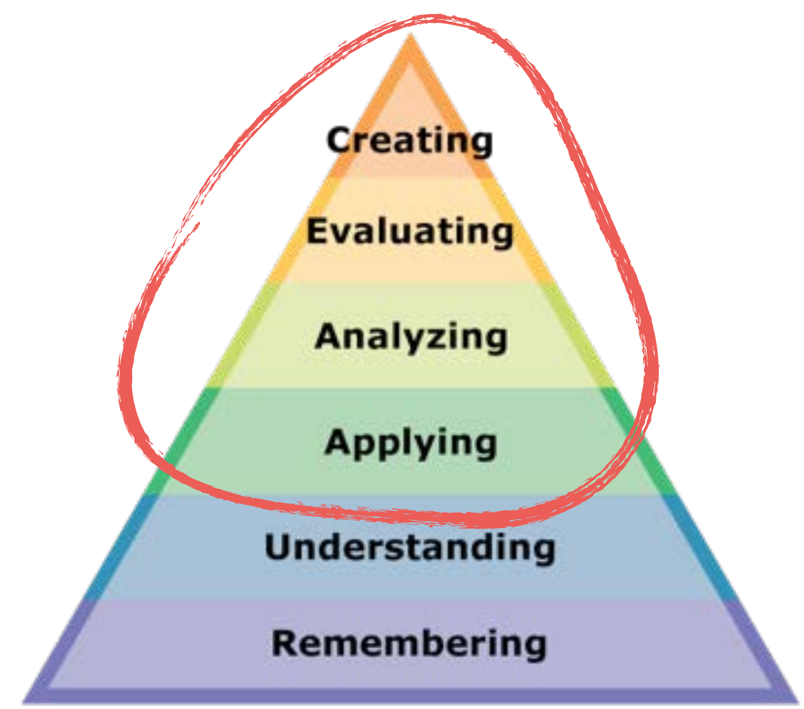
```
yny
n
```

Point Distribution

Precision Definition of Input and Output Format

Applying Knowledge is difficult 😅

- Practice over and over on concrete examples.
- Students autonomously solve complex problems.
- Also the exam consists of complex problems.
- New assessment methods needed.



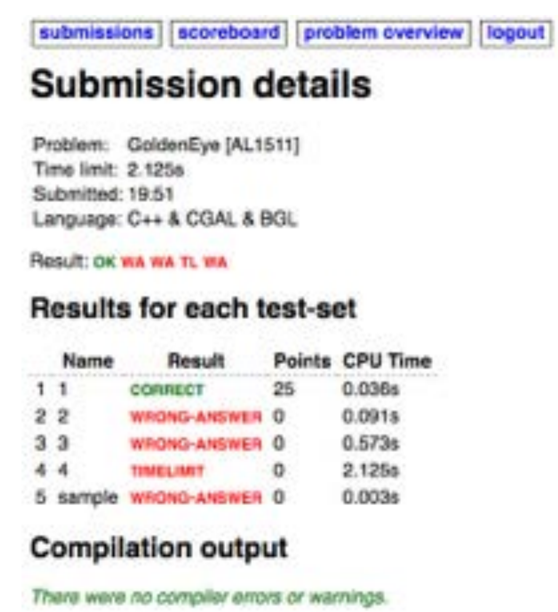
Taxonomy of learning objectives acc. to Bloom/Anderson

[Picture: http://www.learnnc.org/lp/pages/4719]

Black Box Evaluation

CORRECT
WRONG ANSWER
TIMELIMIT
SEGMENTATION FAULT
BUS ERROR
ASSERTION FAILURE

- automatic evaluation by online judge
- 4-5 groups of 20-30 test instances each
- given timeframe (X sec. to complete)
- only completely solved groups score
- arbitrary many submissions, the best is graded



submissions | scoreboard | problem overview | logout

Submission details

Problem: GoldenEye [AL1511]
Time limit: 2.125s
Submitted: 19:51
Language: C++ & CGAL & BGL

Result: OK WA WA TL WA

Results for each test-set

	Name	Result	Points	CPU Time
1	1	CORRECT	25	0.036s
2	2	WRONG-ANSWER	0	0.091s
3	3	WRONG-ANSWER	0	0.573s
4	4	TIMELIMIT	0	2.125s
5	sample	WRONG-ANSWER	0	0.003s

Compilation output

There were no compiler errors or warnings.

Course Format

- Tutorial: Wednesday 17–19
  - background and technical issues related to programming environment and software libraries
  - recap of known algorithmic concepts with examples (also teach a few new ones, but focus on applications rather than theory)
  - 3-fold problems: individual work
- Consulting hours: Wednesday 19– (after the tutorial)
- Problem of the week: Monday 17–19
  Exam like conditions: one problem, 2h to solve
- Individual Work: ~12h / week

- Every Wednesday: new set of problems (3–5)
- Students submit their solutions within one week.
- Automated grading/feedback by an online judge
- We provide solutions for a few selected problems.

Important: Take the time and make the effort to work out the solutions to these problems by yourself.

Picture: [https://pixabay.com/en/board-font-problem-solution-chalk-1521347/]

Course Workload

Tutorial:	$2\,h \times 13 =$	$26\,h$
Problem of the week:	$2\,h \times 13 =$	$26\,h$
Individual Work:	$12\,h \times 13 =$	$156\,h$
Exam preparation:		$20\,h$
Exam:		$12\,h$
Total:		**240h**

= 8 ECTS credits (of 30h each)

Prerequisites

http://www.cadmo.ethz.ch/education/lectures/HS17/algolab/prereqs

- Strategies: Brute force, greedy, divide & conquer, dynamic programming, backtracking, binary search
- Data structures: Array, stack, set, queue, tree, heap, hash-table
- Graph algorithms: DFS, BFS, MST, Dijkstra
- Graph concepts: Directed graph, coloring, matching, topological sorting, (strongly) connected components, matchings

Occasionally we do a recap, but *not* full explanations.

Course Outline

we are here →

1	FUNDAMENTAL ALGORITHMS
2	
3	
4	
5	ADVANCED ALGORITHMS
6	
7	
8	
9	
10	EXAM PREPARATION
11	
12	
13	
14	no tutorial on Dec 20

Course Outline

we are here →

1	Introduction: course, problems, judge, forums
2	BFS/DFS, greedy, divide & conquer
3	Geometric computing in CGAL
4	Graph representations & algorithms in BGL
5	Dynamic programming, brute force, split & list
6	Network flow algorithms in BGL
7	Linear and quadratic programming in CGAL
8	Proximity structures in CGAL
9	Applications of network flows: matchings and cuts
10	How to solve problems in the exam?
11	
12	3-fold problem sets
13	
14	no tutorial on Dec 20

Central Course Web Site: Moodle

`https://moodle-app2.let.ethz.ch/login`

use NETHZ account / NETHZ password

Enrolment Key (for the judge) freelunch

There you can

- download slides of the tutorials,
- download problem sheets,
- submit solutions, and
- discuss problems with your colleagues in the forums.

Getting Help

If you cannot solve a problem, you have two options.

- Your best bet for quick help are our forums, where other students or an assistant can help you out;
- or use the consulting hours after the tutorial.
- Important: Try to solve the problems on your own. In the exam you will not have access to the forums.

For administrative or technical problems (e.g., with the judge or moodle) use

```
algolab@lists.inf.ethz.ch
```

All other questions should be discussed on the forums.

Problem of the Week (PotW)

- Exam-like problem posted every Monday at 17:00
- Solve it within the next 2 hours; a scoreboard shows what you achieved during these 2 hours.
- Use this opportunity to assess your skills. To get a realistic assessment, only use resources also available during the exam (→ `judge/doc`).
- (PotW and exam only) Some testsets are hidden, you do not see the results during the 2/6 hours period.
- (PotW and exam only) Questions about the problem statement must be submitted as a clarification request on the judge.

Computer Rooms

- Solutions to problems (incl. the problem of the week) can be handed in from anywhere.
- We have reserved computer rooms on Mondays which you can use for solving the PotW:
    - In CAB: CAB H 56, CAB H 57
    - In HG: HG E 26.1
- Or use your own computer (+Internet connection).

Exam

The grade is based solely on the exam.

- 6 problems —> 12h = 2 x 6h
- HG computer rooms, no custom hardware
- Submission/judging exactly as during the semester
- Very similar to PotW
- Documentation on `judge/doc`
- No additional material
- Repetition: exam once per year

Afterwards:

- Sample problem: Even Pairs
  - Four solutions and
  - all you need to know to solve problems & submit solutions to the judge.
- Course website and forum etiquette