Technische
Universität
Braunschweig

Algorithm Engineering 2026: Sheet 06 - Exam Prep

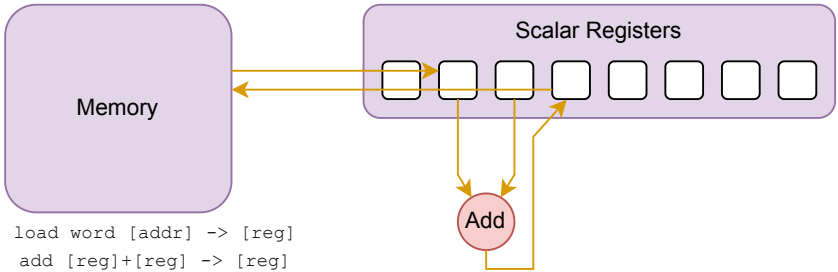
Rouven Kniep, July 16, 2026

1a: Array of Structures vs Structure of Arrays

Describe the terms ‘array of structures’ and ‘structure of arrays’ in your own words and the benefits and drawbacks of each.

1a: Array of Structures vs Structure of Arrays

Scalar Pipeline (example addition):

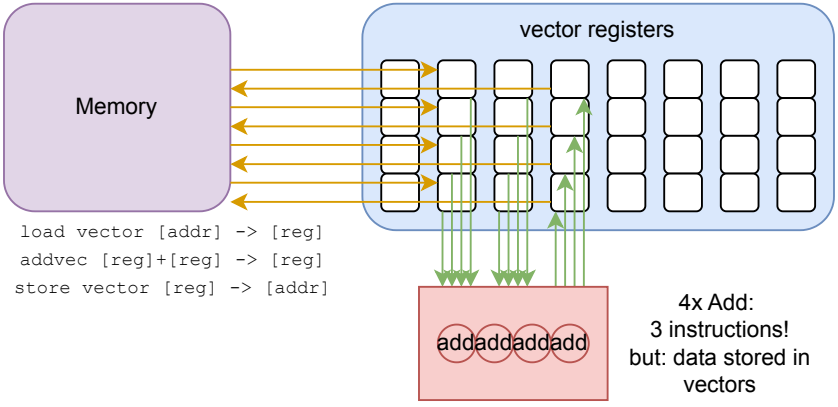



```
load word [addr] -> [reg]
 add [reg]+[reg] -> [reg]
store word [reg] -> [addr]
```

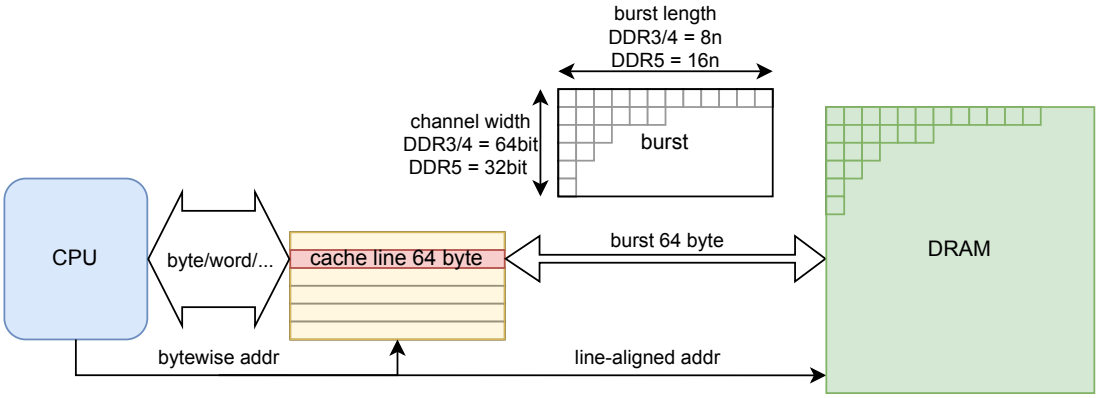
4x Add:
12 instructions

1a: Array of Structures vs Structure of Arrays

SIMD Pipeline (example addition):

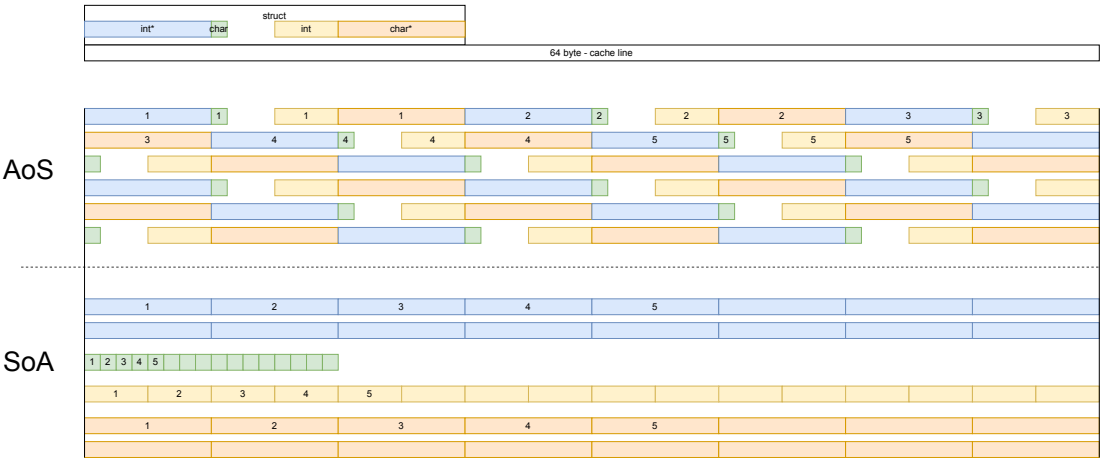


1a: Array of Structures vs Structure of Arrays



(recap from S02)

1a: Array of Structures vs Structure of Arrays



1a: Array of Structures vs Structure of Arrays

Describe the terms ‘array of structures’ and ‘structure of arrays’ in your own words and the benefits and drawbacks of each.

Array of Structures: structs put one after another into the array

- natural to programmers (OOP)
- can easily + efficiently interface indivial objects
- edge case: good for 2d/3d/4d specialized instruction dot products

Structure of Arrays: seperate struct members into parallel arrays per member

- when iterating over single/few members: improved efficiency of memory bandwidth and cache
- compiler can auto-vectorize code for SIMD
- slightly less memory useage

1b: Parallel Solvers

When solving hard problems, what simple approach to parallelization can often be surprisingly (and embarrassingly) competitive with complex, parallelized solvers?

1b: Parallel Solvers

When solving hard problems, what simple approach to parallelization can often be surprisingly (and embarrassingly) competitive with complex, parallelized solvers?

Multithreaded Portfolio Solvers:

- different cheap single-threaded heuristics
- multiple attempts with different parameters & random seeds
- run several in parallel
- return best solution

1b: Parallel Solvers

When solving hard problems, what simple approach to parallelization can often be surprisingly (and embarrassingly) competitive with complex, parallelized solvers?

Multithreaded Portfolio Solvers:

- different cheap single-threaded heuristics
- multiple attempts with different parameters & random seeds
- run several in parallel
- return best solution

Drawbacks & Benefits

- typically different heuristics $\Longrightarrow$ fast
- different approaches $\Longrightarrow$ usually good
- little data sharing $\Longrightarrow$
  - little inter-process communication
  - poor L3 cache hit rate & potential memory data rate bottleneck
  - large RAM required
- typically bad bounds (if any)

1c: Amdahl's Law

State Amdahl's Law and explain what it implies for where you should spend optimization effort.

1c: Amdahl's Law

State Amdahl's Law and explain what it implies for where you should spend optimization effort.

$$S_{overall} = \frac{t_{old}}{t_{optimized}} = \frac{1}{(1-f_{section}) + \frac{f_{section}}{S_{section}}}$$

f fraction of old runtime;
S Speedup; t runtime;

the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used. [1]

1c: Amdahl's Law

State Amdahl's Law and explain what it implies for where you should spend optimization effort.

$$S_{overall} = \frac{t_{old}}{t_{optimized}} = \frac{1}{(1-f_{section}) + \frac{f_{section}}{S_{section}}}$$

f fraction of old runtime;
S Speedup; t runtime;

the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used. [1]

Don't optimize what looks bad, instead **measure and optimize** where it matters!
Optimizing 10% by 10× is less relevant than optimizing 60% by 1.3×

1d: Profiler Families

Profiling tools fall into two big families. Name both, describe each mechanism, and give the main trade-off

1d: Profiler Families

Profiling tools fall into two big families. Name both, describe each mechanism, and give the main trade-off

Sampling

- interrupt program at regular intervals
- then record info (call stack, branch & cache misses, . . .)
- aggregate info into stochastic profile
- cannot get per-line infos (execution time, branching, cache, ..)
- may miss very small functions
- cannot measure test coverage

Instrumentation

- change program execution (emulation or inserting monitoring code)
- attach any measurement instrument
- non-uniformly alters execution time (large monitoring overhead)
- skews measurements (cache & branch hit rate worsens)
- $\Longrightarrow$ bad for performance analysis

2a: Union-Find in Theory & Practice

State the theoretical amortized complexity of Union-Find (find/unite) with union-by-rank + path compression, and the complexity to assume in practice. Justify.

Background: data structure for storing a partition, see Lecture 04.

2a: Union-Find in Theory & Practice

State the theoretical amortized complexity of Union-Find (find/unite) with union-by-rank + path compression, and the complexity to assume in practice. Justify.

Background: data structure for storing a partition, see Lecture 04.

- theory: $\mathcal{O}(\alpha(n))$ **amortized**, with α = inverse Ackermann Function
  - $\alpha(n) \leqslant 5 \quad \forall n \in \mathbb{N} \mid n \leqslant 10^{80}$ (which is basically constant)
  - one $\mathcal{O}(n)$ array $\implies$ little data: 10^7 elements $\approxeq 8$ MiB fit in L3-Cache
  - for **VERY** large n memory scaling (latency, data rate, ...) is more significant
- practice: $\mathcal{O}(1)$

2b: Heuristics for Branch & Bound

Why can heuristics that find good solutions be a valuable addition to branch & bound-based algorithms?

2b: Heuristics for Branch & Bound

Why can heuristics that find good solutions be a valuable addition to branch & bound-based algorithms?

- Background: in B&B, we can prune nodes whose bound is worse than the incumbent
- Heuristics: inform about good solutions - they can be the incumbent
- $\Longrightarrow$ earlier, more agressive and effective pruning
- can help with finding + measuring performance of cutting planes
- can help by warm-starting variable values

3: Delivery Company

Read the problem statement, then apply the five-component recipe — for each component extract the relevant phrase(s) and write the corresponding formal object.

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   each parcel packed once:
   $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   van capacity:
   $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost:
   $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   each parcel packed once:
   $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   van capacity:
   $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost:
   $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns):
   $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   each parcel packed once:
   $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   van capacity:
   $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost:
   $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   each parcel packed once:
   $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   van capacity:
   $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost:
   $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   - each parcel packed once: $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   - van capacity: $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost: $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   - each parcel packed once: $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   - van capacity: $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost: $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

A delivery company must assign every parcel from a given batch to exactly one of its available vans. Each parcel has a known weight, and each van has a fixed weight capacity that the total weight of the parcels loaded into it must not exceed. Loading a parcel into a particular van incurs a handling cost that depends on the parcel–van pair. The company wants to load every parcel while spending as little as possible on handling.

1. Entities (sets): parcels P, vans V
2. Parameters (given constants): weight $w_p \; \forall p \in P$, capacity $c_v \; \forall v \in V$, handling cost $h_{p,v} \; \forall p, v \in P \times V$
3. Variables (unknowns): $x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V$, =1 iff parcel p is assigned to van v
4. Constraints (relations):
   - each parcel packed once: $\sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P$
   - van weight capacity: $\sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V$
5. Objective: minimize total cost: $\min \sum_{p,v \in P \times V} h_{p,v} * x_{p,v}$

3: Delivery Company

$$
\begin{array}{lll}
\text{minimize} & \sum_{p,v \in P \times V} h_{p,v} * x_{p,v} & \text{(minimize cost)} \\
\text{subject to} & \sum_{v \in V} x_{p,v} = 1 \quad \forall p \in P & \text{(each parcel packed)} \\
& \sum_{p \in P} x_{p,v} * w_p \leqslant c_v \quad \forall v \in V & \text{(van weight capacity)} \\
& x_{p,v} \in \mathbb{B} \quad \forall p, v \in P \times V &
\end{array}
$$

4a: Barrier Interior Point vs Simplex

What are the benefits of the barrier interior point algorithm over simplex? What are the benefits of the simplex algorithm? Why can both be a valuable part of a MIP solver and even both be used in solving the same MIP?

4a: Barrier Interior Point vs Simplex

What are the benefits of the barrier interior point algorithm over simplex? What are the benefits of the simplex algorithm? Why can both be a valuable part of a MIP solver and even both be used in solving the same MIP?

Barrier

- well parallelizable
- faster for "difficult" and large LPs
- more consistent runtime (polytime vs simplex' degenerate exptime)
- can handle nonlinear values (for SOCP, QP, ...)
- $\Longrightarrow$ good for the root relaxations

Simplex

- produces sparser, basic solutions (more integral)
- can be warm-started
- $\Longrightarrow$ fast resolve after slightly changing the model
- $\Longrightarrow$ good for child subproblems (=slightly changed parent)

4b: Big-M

State the general Big-M construction for gating a linear constraint by a binary. Then explain the main practical drawback of the Big-M method and the mitigations.

4b: Big-M

State the general Big-M construction for gating a linear constraint by a binary. Then explain the main practical drawback of the Big-M method and the mitigations.

$$y = 1 \implies \sum_i a_i x_i \leqslant b \quad \leftrightarrow \quad \sum_i a_i x_i \leqslant b + (1 - y)M \quad \text{(for a large } M\text{)}$$

Issues

- drastically worsens linear relaxation - example:
  - let $M = 1000; b = 0; a_1 = 1$
  - $y = 0.01$ would allow for $x_1 = 10$
  - but y is almost 0
- numerical issues for very large M

Workarounds:

- avoid wherever reasonable
- tighten M with domain knowledge
- (sometimes) use solver-indicators
- split sums to reduce M.

5a: Dijkstra on Time-Dependent Shortest Path

Time-dependent shortest paths (edge travel time depends on departure time) can still be solved with Dijkstra. Which condition must hold, and what does it mean intuitively?

5a: Dijkstra on Time-Dependent Shortest Path

Time-dependent shortest paths (edge travel time depends on departure time) can still be solved with Dijkstra. Which condition must hold, and what does it mean intuitively?

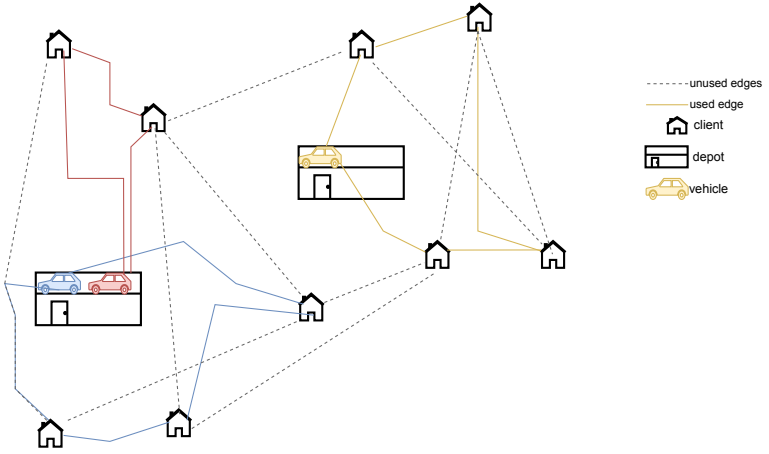
FIFO-condition

- lef $a_e(t) \subset (Time \times Time)$ be arrival time as a function of departure time
- $\forall e \in E : a_e(t)$ is monotonically increasing.
- "departing later cannot lead to earlier arrival"

5b: Vehicle Routing Problem and Variations

Vehicle Routing (VRP): give an informal description of the core problem, then name at least three constraints/objective variants that occur in practice (restrict to those in the lecture).

5b: Vehicle Routing Problem and Variations



5b: Vehicle Routing Problem and Variations

Vehicle Routing (VRP): give an informal description of the core problem, then name at least three constraints/objective variants that occur in practice (restrict to those in the lecture).

base description

- clients need to be served
- cost between locations
- vehicles stationed at depot(s)
- routes need to start/end at the depot
- minimize cost

variants

- capacities: CVPR
- time windows: VRPTW
- pickup+delivery
- shifts, max route length, overtime...
- multi depots
- heterogeneous fleet
- reloading, multi-trip
- optional clients / price collection
- soft objectives

6a: CDCL

$1\overline{2},\ 123,\ \overline{1}\overline{2}3,\ 12\overline{3}4,\ \overline{3}4,\ \overline{1}\overline{2}4,\ \overline{1}2$

Perform CDCL on the following formula until you have run into two conflicts or found the formula to be satisfied or unsatisfiable. When making a decision, prioritize variables with lower indices over those with higher indices, and set the variable to false in your decision.

level	literal	reason
1	$\overline{1}$	Δ
1	$\overline{2}$	$1\overline{2}$
1	3	123
1	4	$12\overline{3}4$
SAT		

6a: CDCL (fixed formula)

$1\overline{2},\ 123,\ \overline{12}3,\ 12\overline{3}4,\ \overline{34},\ \overline{12}4,\ \overline{1}2,$

6a: CDCL (fixed formula)

$1\overline{2},\ 123,\ \overline{12}3,\ 12\overline{3}4,\ \overline{34},\ \overline{12}4,\ \overline{1}2,$

level	literal	reason
1	$\overline{1}$	Δ
1	$\overline{2}$	$1\overline{2}$
1	3	123
1	$\overline{4}$	$\overline{34}$
1*	4	$12\overline{3}4$

$$12\overline{3}4 \diamond_4 \overline{34} = 12\overline{3}$$
$$12\overline{3} \diamond_3 123 = 12$$
$$12 \diamond_2 1\overline{2} = 1$$

6a: CDCL (fixed formula)

$1\overline{2},\ 123,\ \overline{12}3,\ 12\overline{3}4,\ \overline{34},\ \overline{12}4,\ \overline{1}2,\quad \underline{1},$

level	literal	reason
1	$\overline{1}$	Δ
1	$\overline{2}$	$1\overline{2}$
1	3	123
1	$\overline{4}$	$\overline{34}$
1*	4	$12\overline{3}4$

$$12\overline{3}4 \diamond_4 \overline{34} = 12\overline{3}$$
$$12\overline{3} \diamond_3 123 = 12$$
$$12 \diamond_2 1\overline{2} = 1$$

6a: CDCL (fixed formula)

$1\overline{2},\ 123,\ \overline{12}3,\ 12\overline{3}4,\ \overline{34},\ \overline{12}4,\ \overline{1}2,\quad \underline{1},$

level	literal	reason
1	$\overline{1}$	Δ
1	$\overline{2}$	$1\overline{2}$
1	3	123
1	$\overline{4}$	$\overline{34}$
1*	4	$12\overline{3}4$

$$12\overline{3}4 \diamond_4 \overline{34} = 12\overline{3}$$
$$12\overline{3} \diamond_3 123 = 12$$
$$12 \diamond_2 1\overline{2} = 1$$

level	literal	reason
0	1	1
0	2	$\overline{1}2$
0	3	$\overline{12}3$
0	$\overline{4}$	$\overline{34}$
0*	4	$\overline{12}4$

$$\overline{12}4 \diamond_4 \overline{34} = \overline{123}$$
$$\overline{123} \diamond_3 \overline{12}3 = \overline{12}$$
$$\overline{12} \diamond_2 \overline{1}2 = \overline{1}$$
$$\overline{1} \diamond_1 1 = \bot$$

6b: Integer Variables and VSIDS

Why is branching by the VSIDS (SAT decision) heuristic alone not sufficient to reach a complete solution in CP-SAT? What mechanism closes the gap?

6b: Integer Variables and VSIDS

Why is branching by the VSIDS (SAT decision) heuristic alone not sufficient to reach a complete solution in CP-SAT? What mechanism closes the gap?

Problem:

- VSIDS: choose highest scoring variable for branching
- CP-SAT integers: Order abstraction (vars for $x \leqslant k$ and $x = k$)
- CP-SAT encoding boolean literals lazily created
- required: exact assignment
- but: solver works on abstraction

6b: Integer Variables and VSIDS

Why is branching by the VSIDS (SAT decision) heuristic alone not sufficient to reach a complete solution in CP-SAT? What mechanism closes the gap?

Problem:

- VSIDS: choose highest scoring variable for branching
- CP-SAT integers: Order abstraction (vars for $x \leqslant k$ and $x = k$)
- CP-SAT encoding boolean literals lazily created
- required: exact assignment
- but: solver works on abstraction

Solution:

- SAT branching: decides all existing literals (ex: $\overline{x \leqslant 2} \wedge x \leqslant 7$)
- result: interval info (ex: $x \in [3; 7]$)
- abstraction refinement required $\implies$ create new literals (ex: $x \leqslant 5$)
- different refinement ideas (min-first / max-first, propagation, ...)
- resolve, until every variable is fixed

Exam

- place: SN 19.1
- time: July 30th, 12:30-14:30 - be there 12:15!
- permitted aids: Dictionary (any $\leftrightarrow$ german)
- required: black/blue permanent pen, ruler
- language: **german** (legally required, as far as we know)