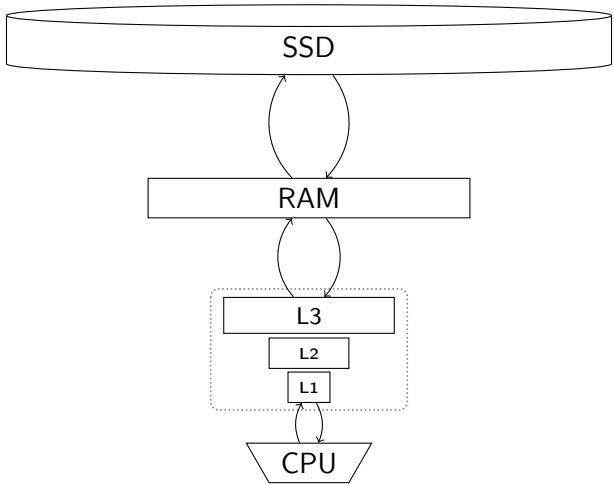
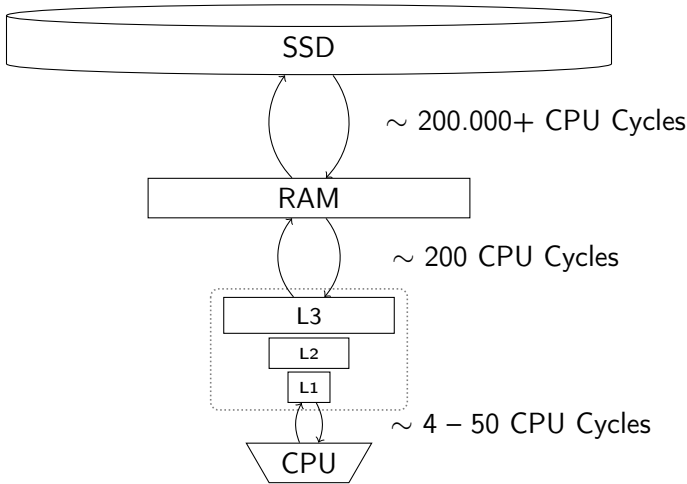
Adiar 1.1 : Zero-suppressed Decision Diagrams in External Memory

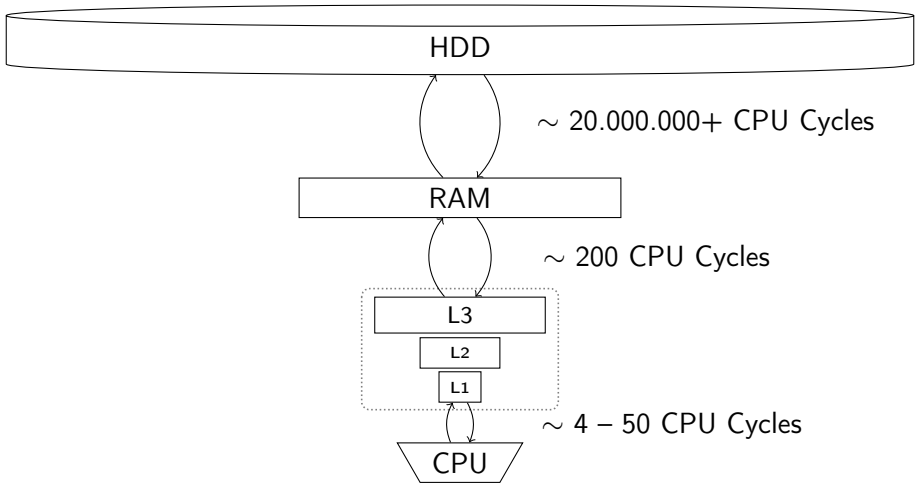
---

Steffan Christ Sølvsten and Jaco van de Pol

NFM 2023


SSD
RAM
L3
L2
L1
CPU


SSD
~ 200.000+ CPU Cycles
RAM
~ 200 CPU Cycles
L3
L2
L1
~ 4 – 50 CPU Cycles
CPU


HDD
~ 20.000.000+ CPU Cycles
RAM
~ 200 CPU Cycles
L3
L2
L1
~ 4 – 50 CPU Cycles
CPU

Adiar

Binary Decision Diagrams
in External Memory

github.com/ssoelvsten/adiar

Adiar

Multi-terminal Decision Diagrams
in External Memory

`github.com/ssoelvsten/adiar`

Adiar

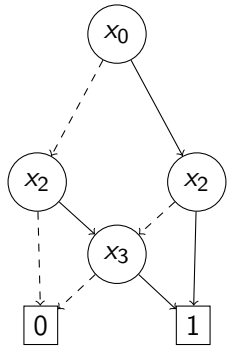
Quantum Multi-valued Decision Diagrams
in External Memory

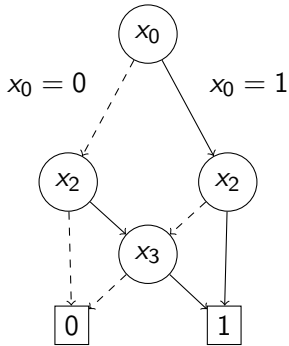
github.com/ssoelvsten/adiar

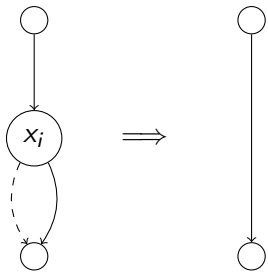
Adiar

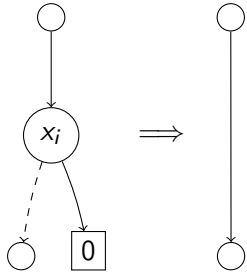
Zero-suppressed Decision Diagrams
in External Memory

`github.com/ssoelvsten/adiar`


x_0
x_2
x_2
x_3
0
1


x_0
$x_0 = 0$
$x_0 = 1$
x_2
x_2
x_3
0
1





BDD: $f : \mathbb{B}^n \to \mathbb{B}$

ZDD: $A \subseteq \mathbb{B}^n$

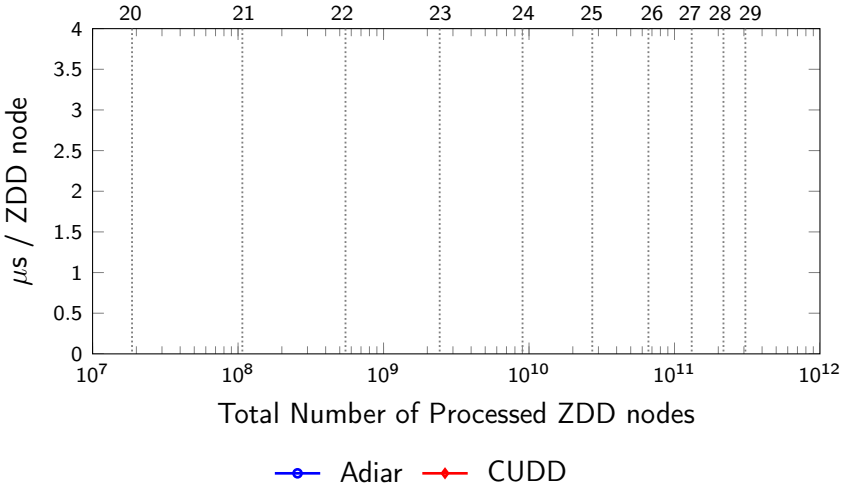
```
bdd bdd_apply(bdd f, bdd g, bool_op o)
```

```
bdd bdd_apply(bdd f, bdd g, bool_op o)
```

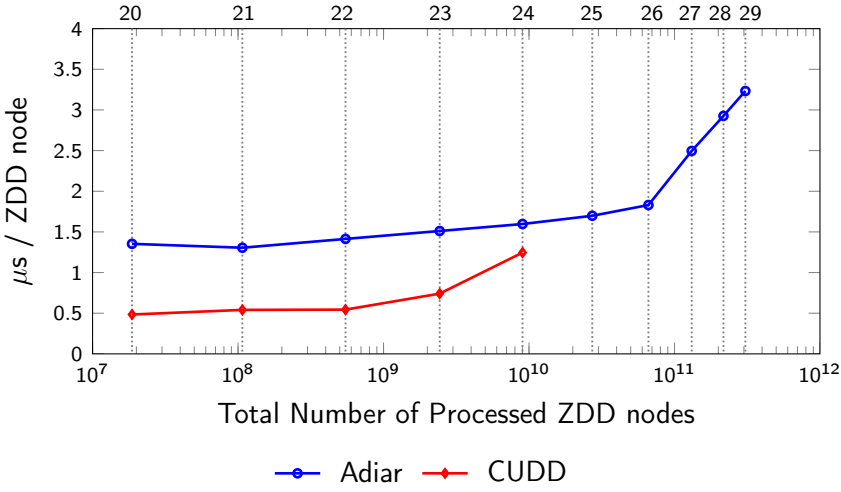
```
zdd zdd_binop(zdd A, zdd B, bool_op o)
```

```
bdd bdd_apply(bdd f, bdd g, bool_op o)  {
    return prod2<bdd_policy>(f, g, o);
}

zdd zdd_binop(zdd A, zdd B, bool_op o)  {
    return prod2<zdd_policy>(A, B, o);
}
```

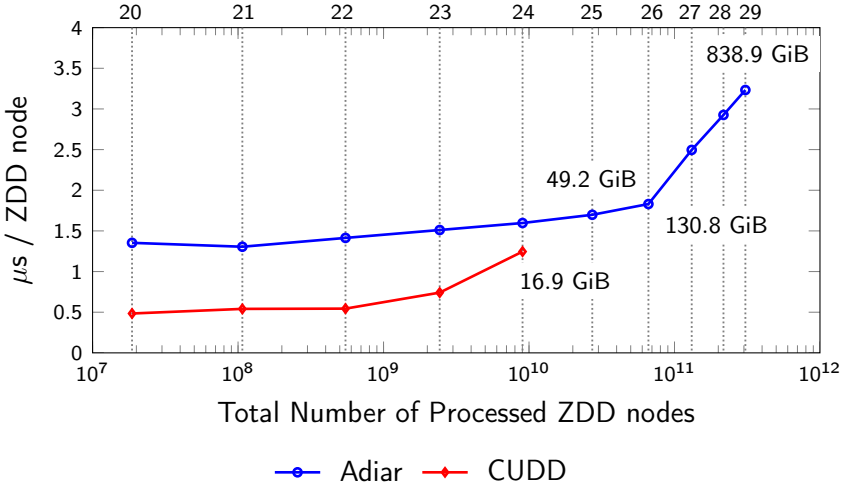
```
bdd bdd_apply(bdd f, bdd g, bool_op o)  {
    return prod2<bdd_policy>(f, g, o);
}

zdd zdd_binop(zdd A, zdd B, bool_op o)  {
    return prod2<zdd_policy>(A, B, o);
}
```



Running time for *3D Tic-Tac-Toe* with 300 GiB of RAM.



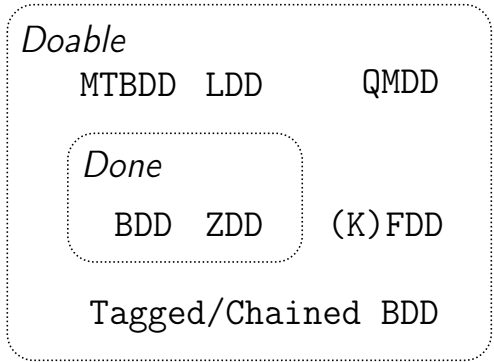
Running time for *3D Tic-Tac-Toe* with 300 GiB of RAM.

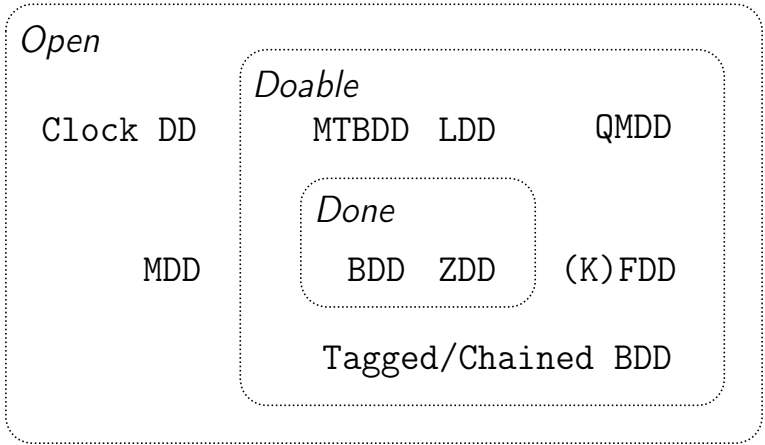


Running time for *3D Tic-Tac-Toe* with 300 GiB of RAM.

Done

BDD ZDD


Doable
MTBDD
LDD
QMDD
Done
BDD
ZDD
(K)FDD
Tagged/Chained BDD


Open
Clock DD
MDD
Doable
MTBDD LDD
QMDD
Done
BDD ZDD
(K)FDD
Tagged/Chained BDD

Steffan Christ Sølvsten

soelvsten@cs.au.dk

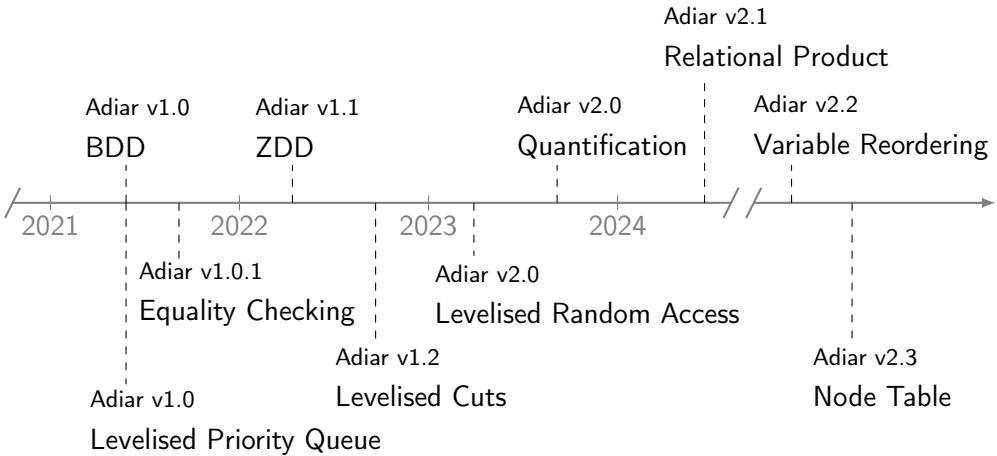
ssoelvsten.github.io

Adiar

github.com/ssoelvsten/adiar

ssoelvsten.github.io/adiar

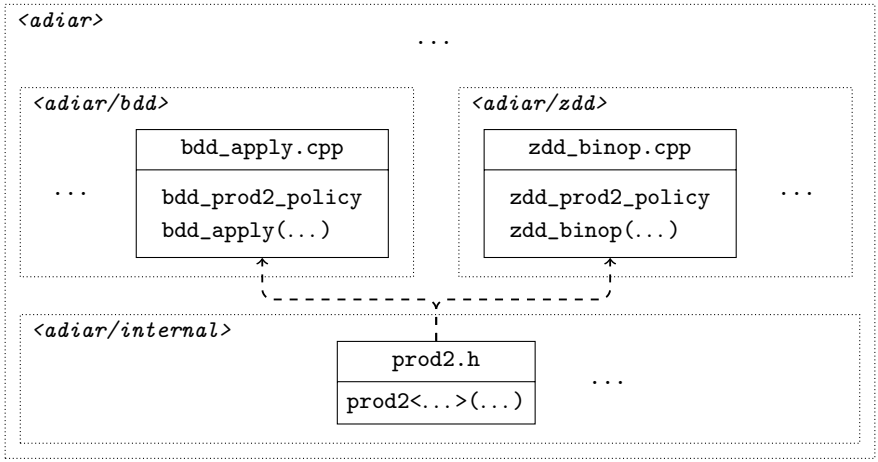
AARHUS
UNIVERSITY

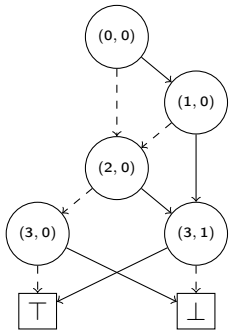

Adiar v2.1
Relational Product
Adiar v1.0
BDD
Adiar v1.1
ZDD
Adiar v2.0
Quantification
Adiar v2.2
Variable Reordering
2021
2022
2023
2024
Adiar v1.0.1
Equality Checking
Adiar v2.0
Levelised Random Access
Adiar v1.2
Levelised Cuts
Adiar v2.3
Node Table
Adiar v1.0
Levelised Priority Queue

Function	Operation Semantics
ZDD Constructors	
`zdd_empty()`	$\emptyset$
`zdd_null()`	$\{\emptyset\}$
`zdd_singleton(var)`	$\{x_{\textbf{\textit{var}}}\}$
`zdd_vars(vars)`	$\{\bigcup_{i \in \textbf{\textit{vars}}} \{x_i\}\}$
`zdd_singletons(vars)`	$\{\{x_i\} \mid i \in \textbf{\textit{vars}}\}$
`zdd_powerset(vars)`	$\mathcal{P}(\textbf{\textit{vars}})$
`zdd_sized_set(vars,`k`,`$\odot$`)`	$\{s \in \mathcal{P}(\textbf{\textit{vars}}) \mid \lvert s\rvert \odot k\}$
ZDD Manipulation	
`zdd_binop(`A`,`B`,`$\otimes$`)`	$\{x \mid x \in A \otimes x \in B\}$
`zdd_change(`A`, vars)`	$\{(a \setminus \textbf{\textit{vars}}) \cup (\textbf{\textit{vars}} \setminus a) \mid a \in A\}$
`zdd_complement(`A`, dom)`	$\mathcal{P}(\textbf{\textit{dom}}) \setminus A$
`zdd_expand(`A`, vars)`	$\bigcup_{a \in A} \{a \cup v \mid v \in \mathcal{P}(\textbf{\textit{vars}})\}$
`zdd_offset(`A`, vars)`	$\{a \in A \mid \textbf{\textit{vars}} \cap a = \emptyset\}$
`zdd_onset(`A`, vars)`	$\{a \in A \mid \textbf{\textit{vars}} \subseteq a\}$
`zdd_project(`A`, vars)`	$\bigcup_{a \in A} \{a \cap \textbf{\textit{vars}}\}$

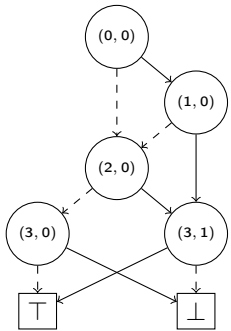
Function	Operation Semantics
Counting	
`zdd_size(`A`)`	$\lvert A\rvert$
`zdd_nodecount(`A`)`	# ZDD Nodes in A
`zdd_varcount(`A`)`	# Non-empty Levels in A
Predicates	
`zdd_equal(`A`, `B`)`	$A = B$
`zdd_unequal(`A`, `B`)`	$A \neq B$
`zdd_subseteq(`A`, `B`)`	$A \subseteq B$
`zdd_disjoint(`A`, `B`)`	$A \cap B = \emptyset$
Set elements	
`zdd_contains(`A`, `a`)`	$a \in A$
`zdd_minelem(`A`)`	$a \in A$ s.t. $\forall a' \in A \,.\, a \leq a'$
`zdd_maxelem(`A`)`	$a \in A$ s.t. $\forall a' \in A \,.\, a' \leq a$
Conversion	
`zdd_from(`f`, dom)`	$\{x \in \mathcal{P}(\textbf{\textit{dom}}) \mid f(x) = \top\}$
`bdd_from(`A`, dom)`	$\vec{x} : \mathcal{P}(\textbf{\textit{dom}}) \mapsto \vec{x} \in A$

Operations provided by Adiar in `<adiar/zdd.h>`.


<adiar>
...
<adiar/bdd>
bdd_apply.cpp
bdd_prod2_policy
bdd_apply(...)
...
<adiar/zdd>
zdd_binop.cpp
zdd_prod2_policy
zdd_binop(...)
...
<adiar/internal>
prod2.h
prod2<...>(...)
...



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

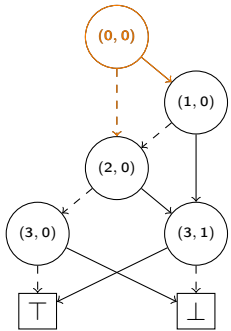


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: Q_{count}:

[

]

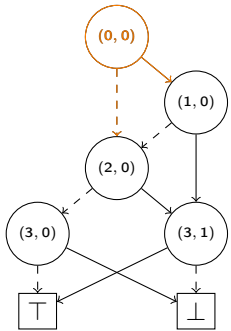


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: Q_{count}:

[

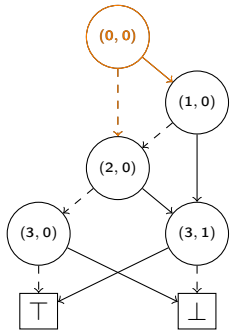
]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: Q_{count}:

$$\left[\begin{array}{ll} ((0,0) \xrightarrow{\top} (1,0), & 1) \\ ((0,0) \xrightarrow{\perp} (2,0), & 1) \end{array} \right]$$

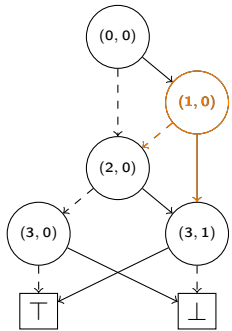


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1, 0)$	0	0

Priority Queue: Q_{count}:

$$[\quad ((0,0) \xrightarrow{\top} (1,0),\quad 1)\quad ,$$
$$((0,0) \xrightarrow{\bot} (2,0),\quad 1)\quad ,$$
$$]$$



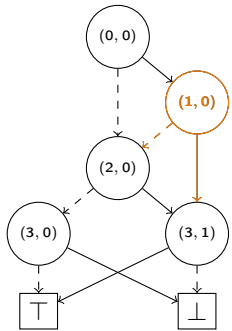
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1, 0)$	0	0

Priority Queue: Q_{count}:

$$[\quad ((0,0) \xrightarrow{\top} (1,0), \quad 1) \quad ,$$
$$((0,0) \xrightarrow{\bot} (2,0), \quad 1) \quad ,$$

$]$



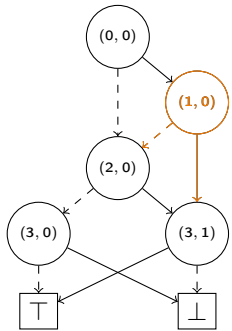
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1, 0)$	1	0

Priority Queue: Q_{count}:

$$[\quad ((0,0) \xrightarrow{\perp} (2,0), \quad 1) \quad ,$$

$$]$$



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1,0)$	1	0

Priority Queue: Q_{count}:

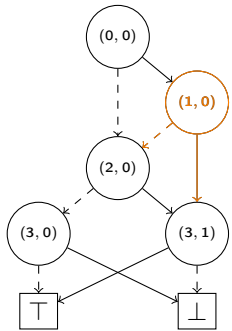
[

$((0,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(2, 0)$	0	0

Priority Queue: Q_{count}:

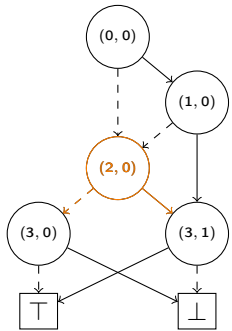
[

$((0,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

]

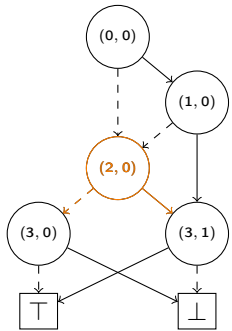


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(2, 0)$	0	0

Priority Queue: Q_{count}:

$$
\begin{bmatrix}
((0,0) \xrightarrow{\perp} (2,0), & 1) & , \\
((1,0) \xrightarrow{\perp} (2,0), & 1) & , \\
 & & \\
((1,0) \xrightarrow{\top} (3,1), & 1) & , \\
 & &]
\end{bmatrix}
$$



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(2,0)$	1	0

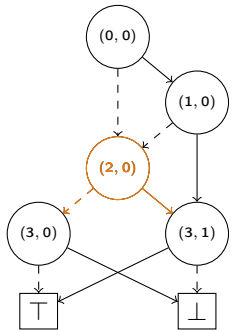
Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

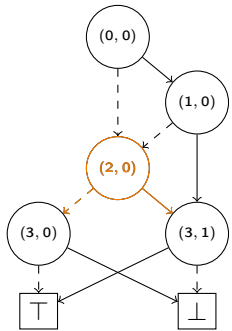
Seek	Sum	Result
$(2, 0)$	2	0

Priority Queue: Q_{count}:

[

$((1, 0) \xrightarrow{\top} (3, 1), \quad 1)$,

]



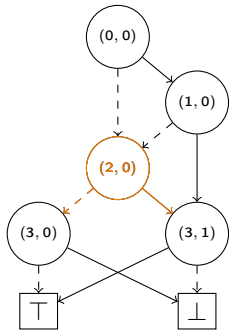
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(2, 0)$	2	0

Priority Queue: Q_{count}:

[

$$((2,0) \xrightarrow{\perp} (3,0), \quad 2) \quad ,$$
$$((1,0) \xrightarrow{\top} (3,1), \quad 1) \quad ,$$
$$((2,0) \xrightarrow{\top} (3,1), \quad 2) \quad]$$



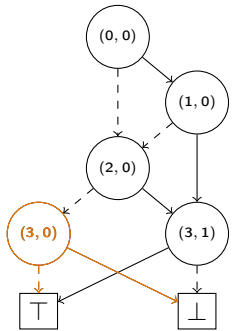
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3, 0)$	0	0

Priority Queue: Q_{count}:

[

$((2,0) \xrightarrow{\perp} (3,0), \quad 2)$,
$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,
$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



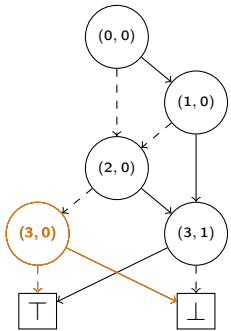
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3, 0)$	0	0

Priority Queue: Q_{count}:

[

$((2,0) \xrightarrow{\perp} (3,0), \quad 2)$,
$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,
$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

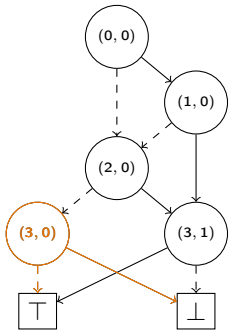
Seek	Sum	Result
$(3, 0)$	2	0

Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

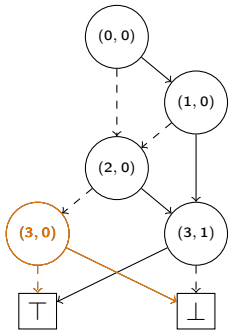
Seek	Sum	Result
$(3, 0)$	2	2

Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

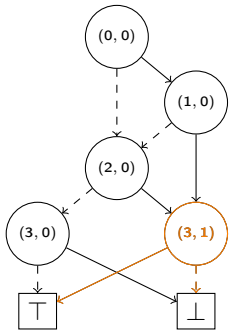
Seek	Sum	Result
$(3,1)$	0	2

Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

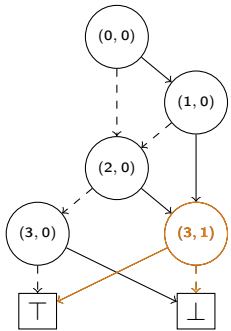
Seek	Sum	Result
$(3,1)$	0	2

Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



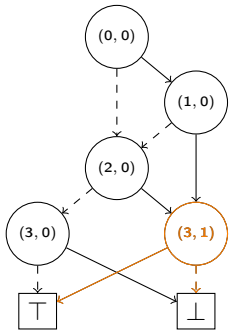
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3,1)$	1	2

Priority Queue: Q_{count}:

[

$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]



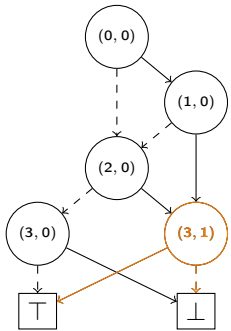
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3, 1)$	3	2

Priority Queue: Q_{count}:

[

]



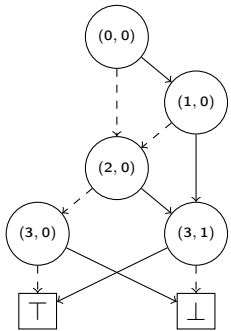
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3, 1)$	3	5

Priority Queue: Q_{count}:

[

]



(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Result

5

Priority Queue: Q_{count}:

[

]

Steffan Christ Sølvsten

soelvsten@cs.au.dk

ssoelvsten.github.io

Adiar

github.com/ssoelvsten/adiar

ssoelvsten.github.io/adiar

AARHUS
UNIVERSITY