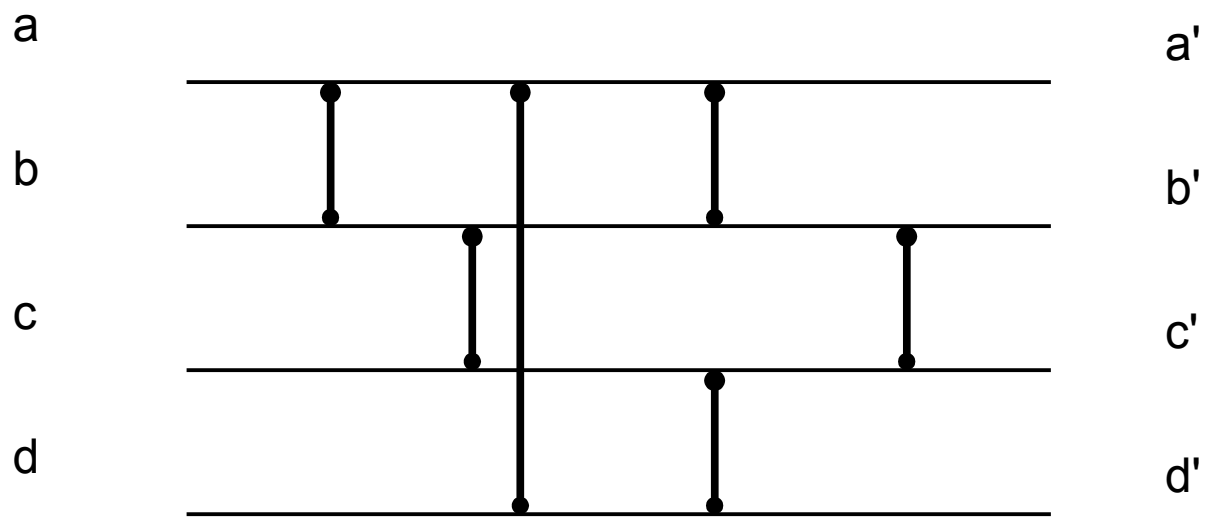
Optimal Sorting Networks

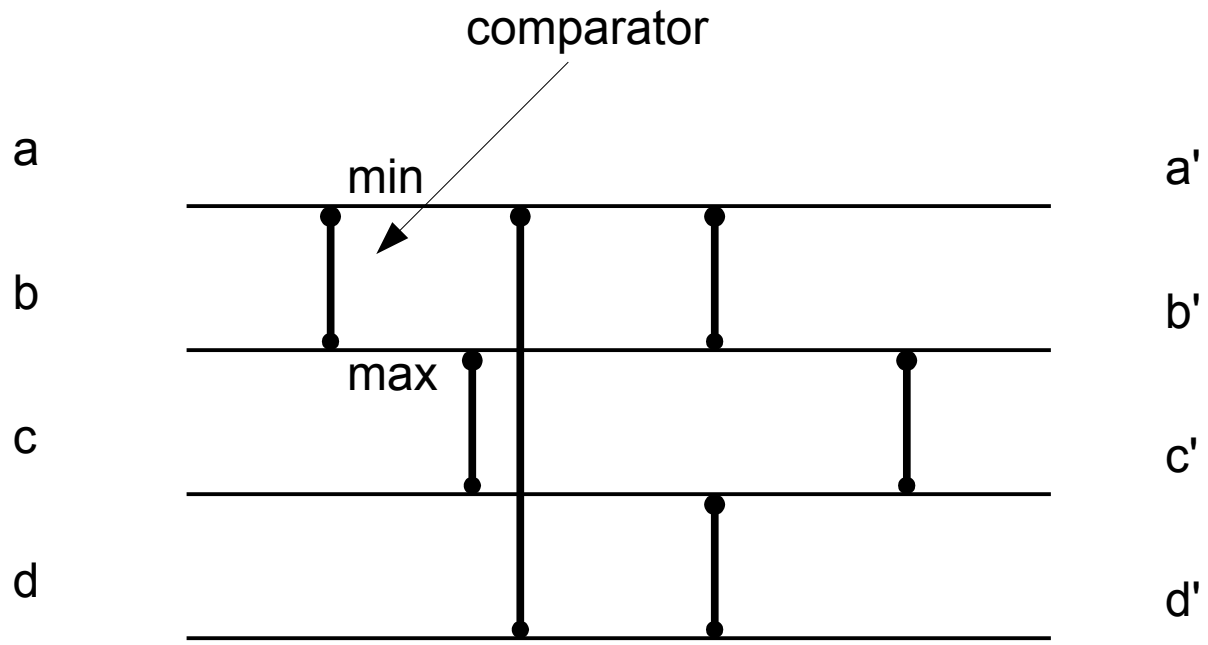
Daniel Bundala & Jakub Závodný
University of Oxford

Sorting in Hardware

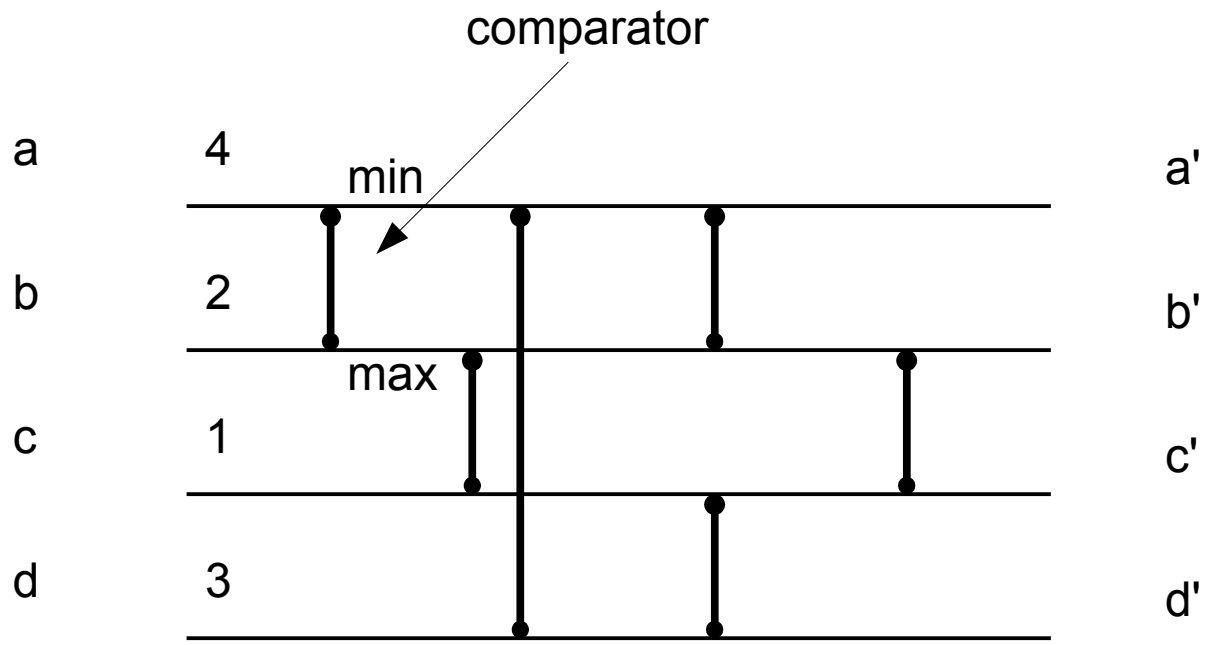
Sorting in Hardware



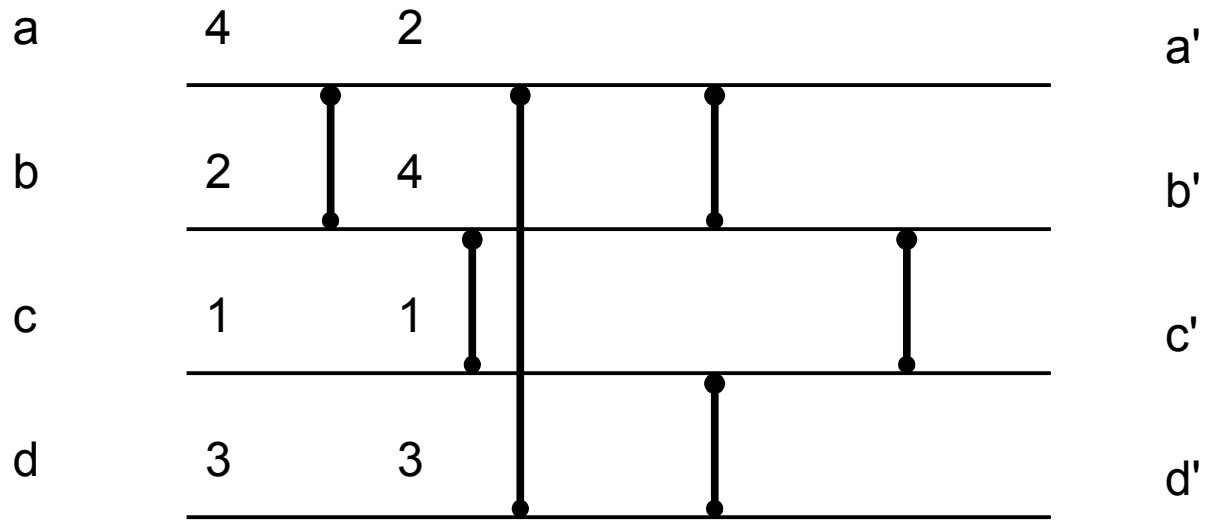
Sorting in Hardware



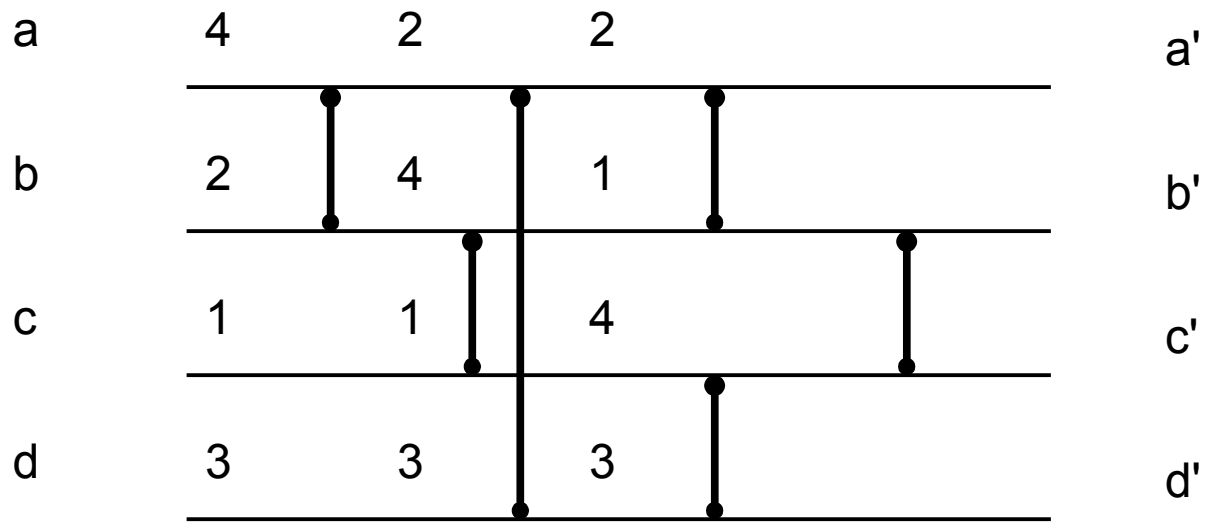
Sorting in Hardware



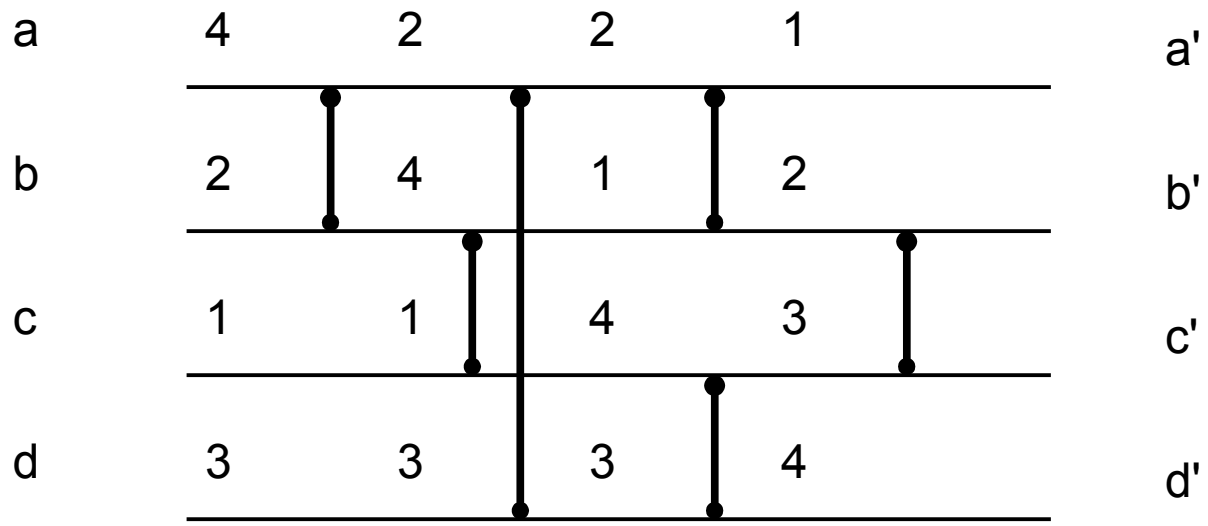
Sorting in Hardware



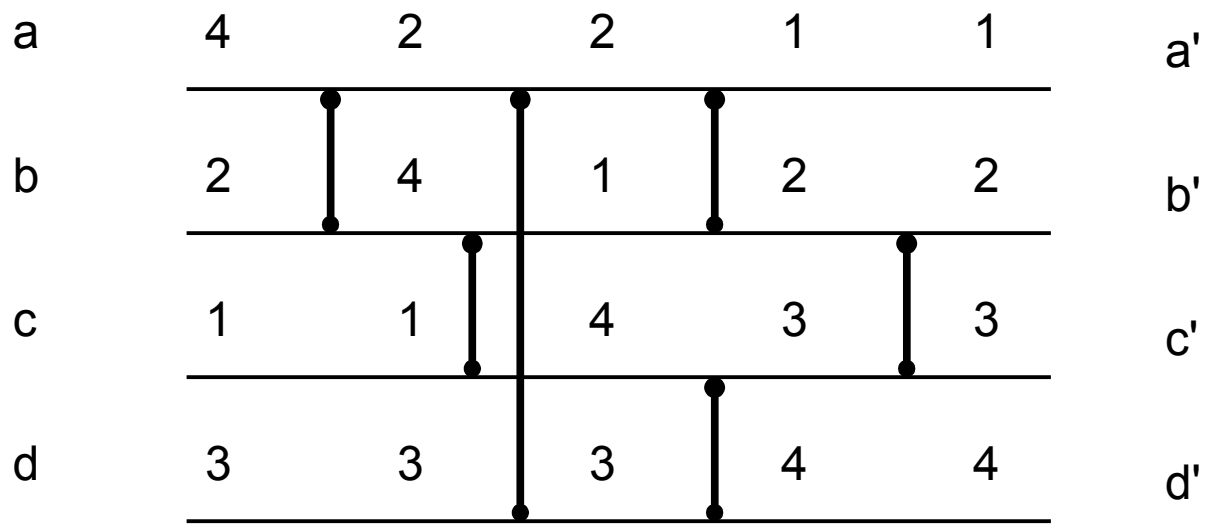
Sorting in Hardware



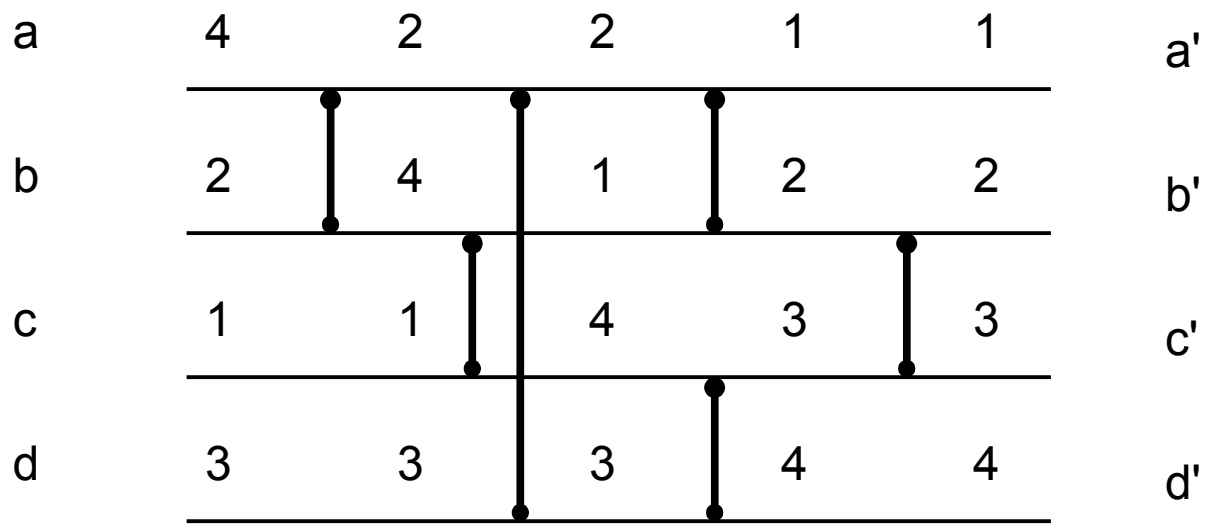
Sorting in Hardware



Sorting in Hardware

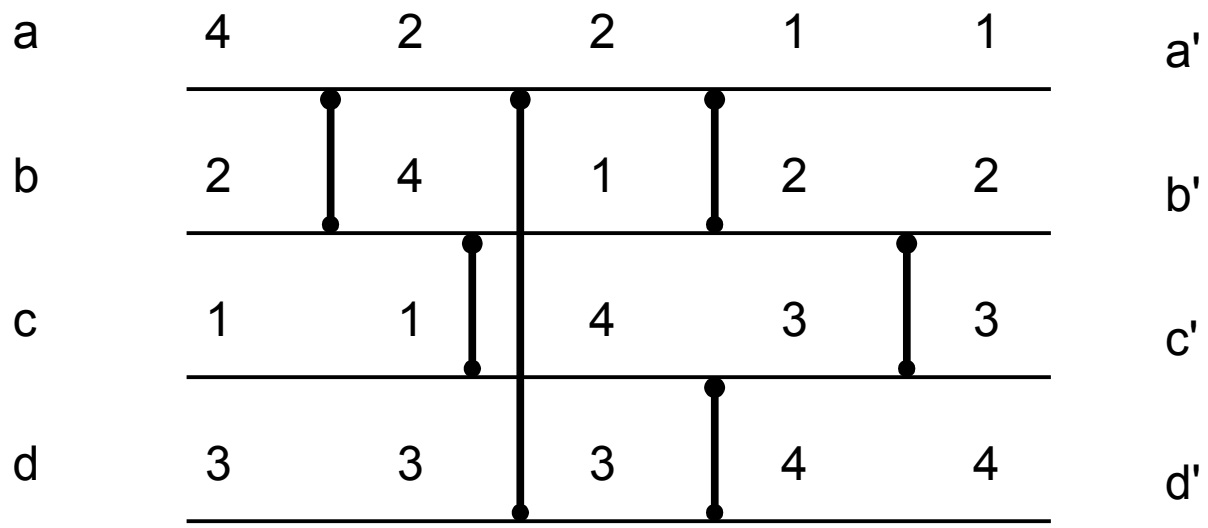


Sorting in Hardware



Sorting network – sorts every input

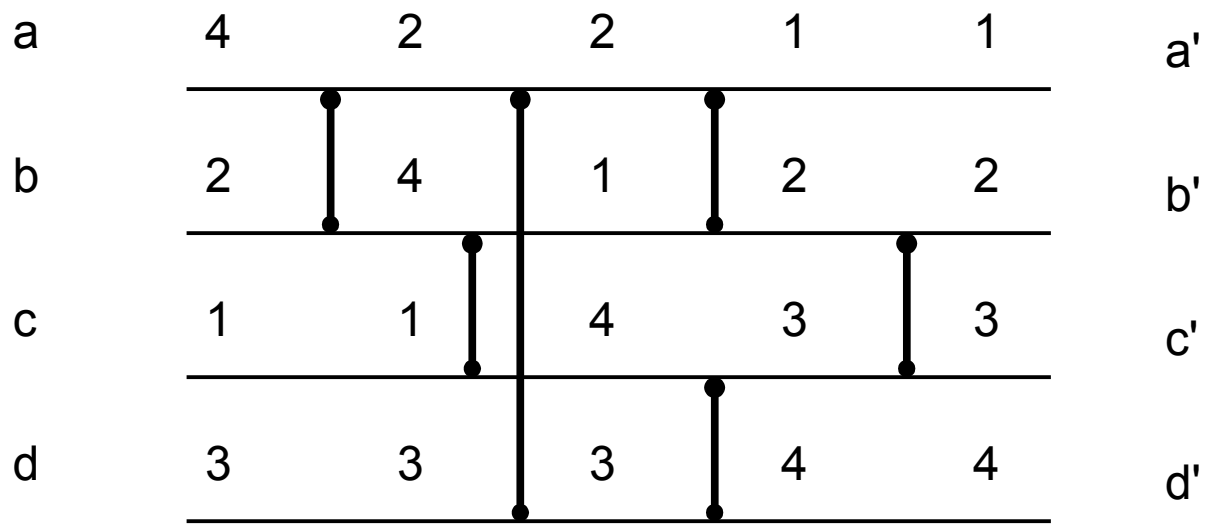
Sorting in Hardware



Sorting network – sorts every input

Time/Depth: 4

Sorting in Hardware



Sorting network – sorts every input

Time/Depth: 4

What is the fastest way to sort n inputs?

General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

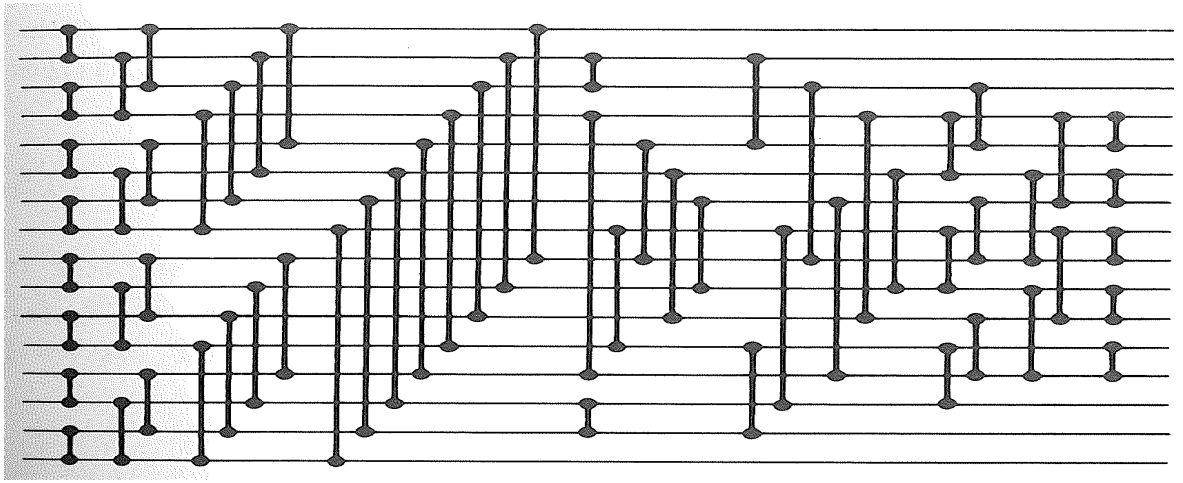
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

Knuth (1973) – The Art of Computer Programming, Vol. 3



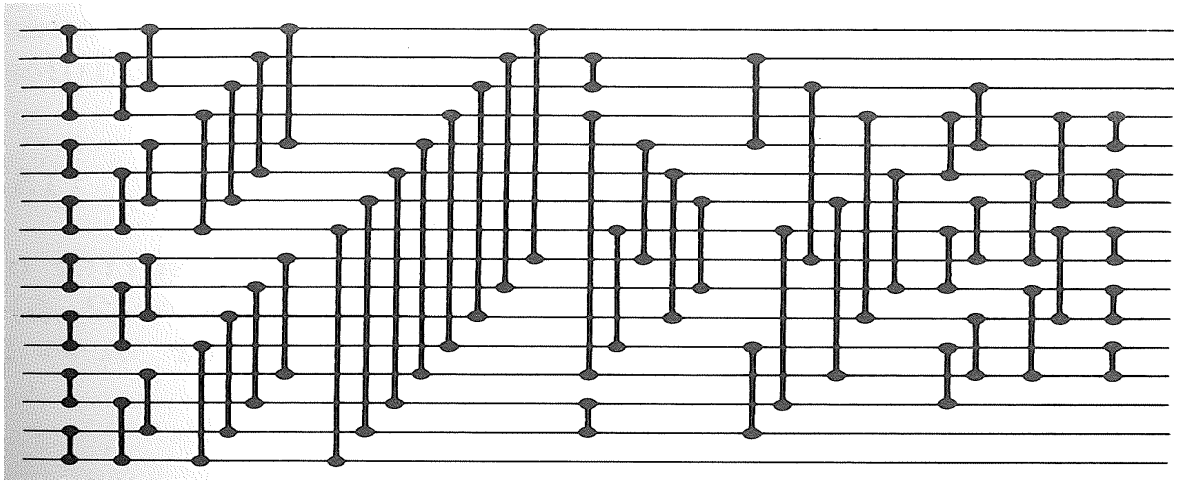
General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

Knuth (1973) – The Art of Computer Programming, Vol. 3

Floyd, Knuth (1973)



General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

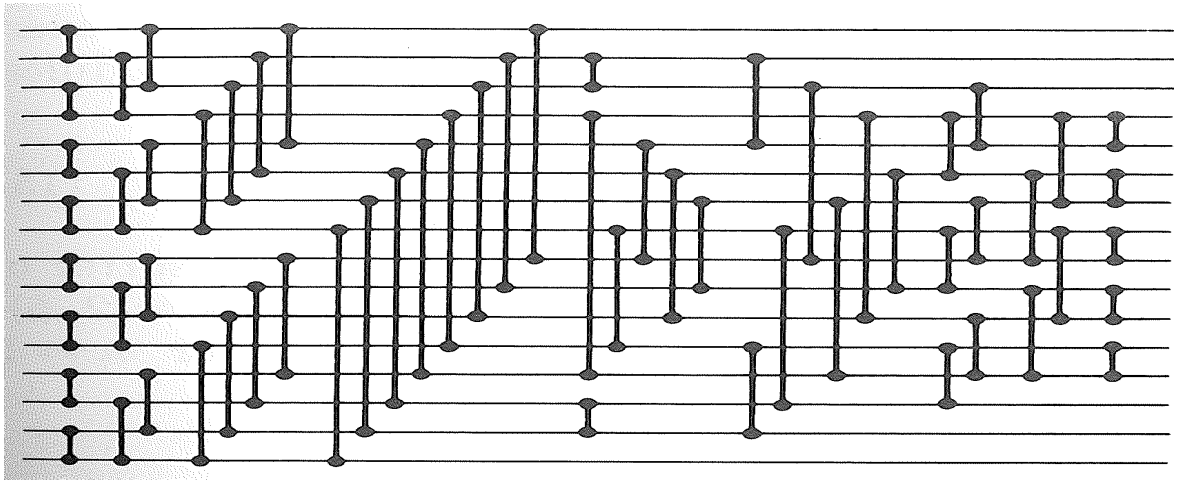
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

7 7

Knuth (1973) – The Art of Computer Programming, Vol. 3

Floyd, Knuth (1973)

Parberry (1989)



General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

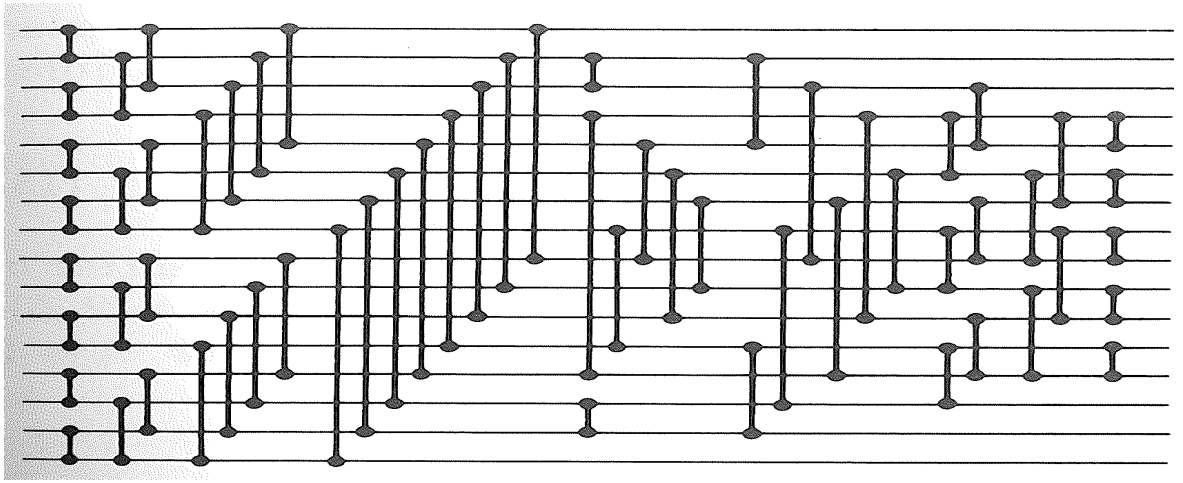
7 7 8 8 9 9 9 9

Knuth (1973) – The Art of Computer Programming, Vol. 3

Floyd, Knuth (1973)

Parberry (1989)

Bundala, Zavodny (2013)



General n: Ajtai, Komlos, Szemeredi (1983) – Θ (log n)

Specific n: Knuth (1973) – The Art of Computer Programming

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	6	6	6	6	6	6	6	6

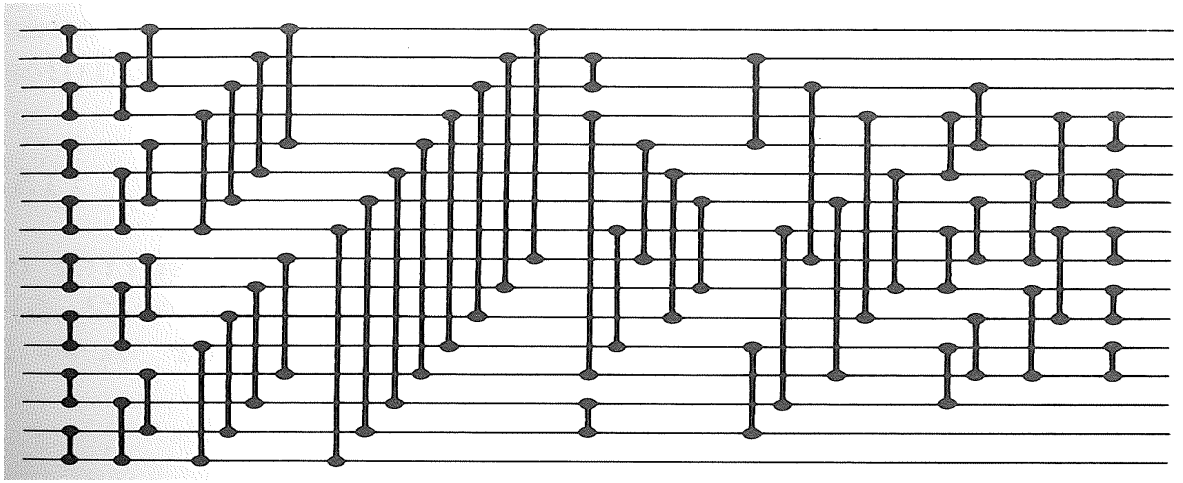
7 7 8 8 9 9 9 9

Knuth (1973) – The Art of Computer Programming, Vol. 3

Floyd, Knuth (1973)

Parberry (1989)

Bundala, Zavodny (2013)

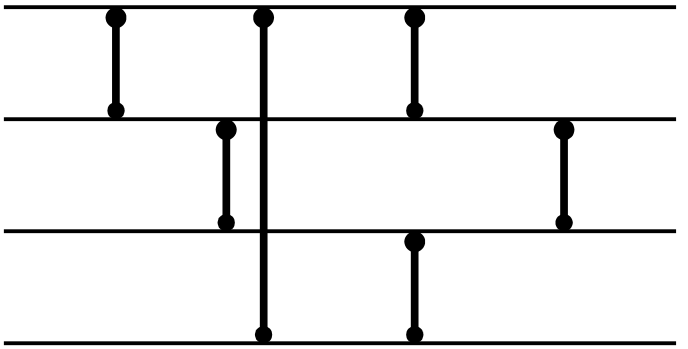


How to show that no network of depth 7 sorts 11 inputs?

How to show that no network of depth 7 sorts 11 inputs?

How to show that no network of depth 7 sorts 11 inputs?

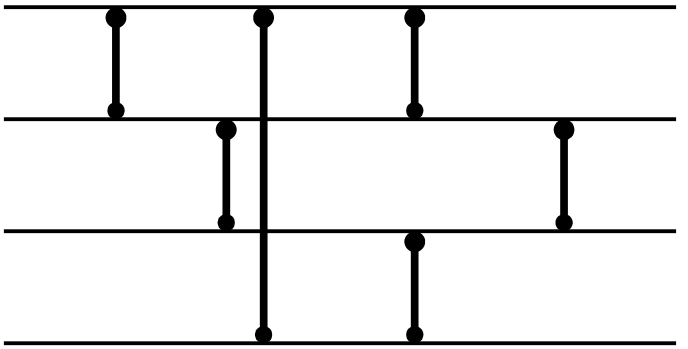
35696 candidates for each layer



How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

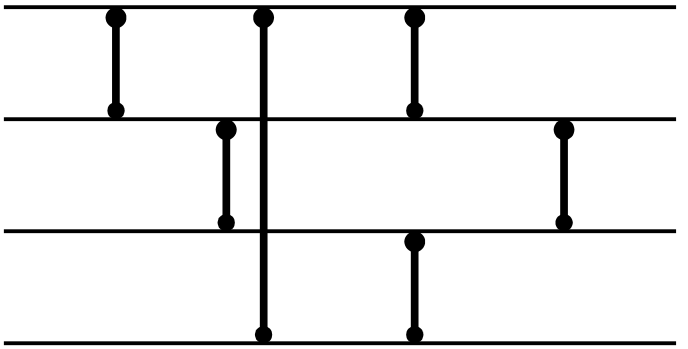
$35696^7 > 7 \times 10^{31}$ networks



How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

$35696^7 > 7 \times 10^{31}$ networks

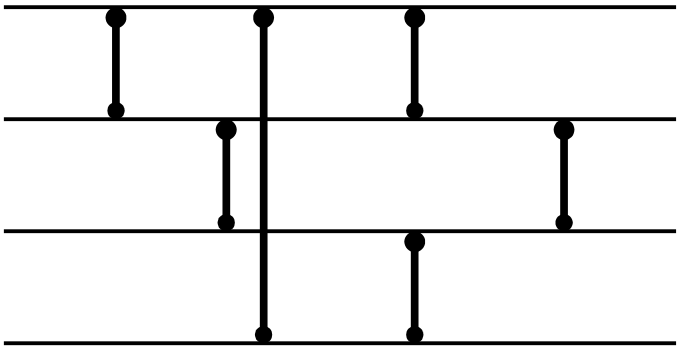


Build propositional formula satisfiable iff a depth 7 network sorts 11 inputs

How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

$35696^7 > 7 \times 10^{31}$ networks



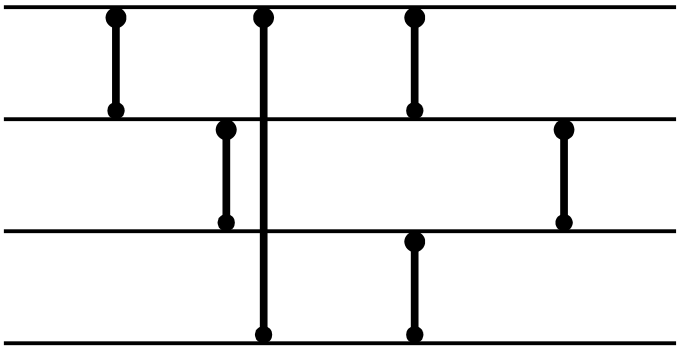
Build propositional formula satisfiable iff a depth 7 network sorts 11 inputs

Variables: C(i,j,l) – true if comparator (i,j) in level l

How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

$35696^7 > 7 \times 10^{31}$ networks



Build propositional formula satisfiable iff a depth 7 network sorts 11 inputs

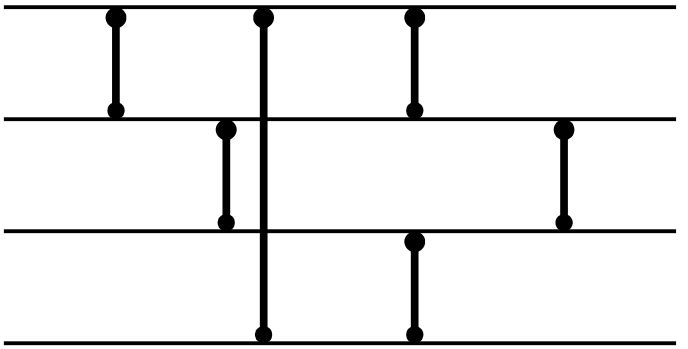
Variables: C(i,j,l) – true if comparator (i,j) in level l

Formula: $\left(\bigwedge_{\substack{\text{line} \\ 1 \le i \le 11}} \bigwedge_{\substack{\text{layer} \\ 1 \le l \le 7}} i \text{ used at most once at layer } l \right) \bigwedge \text{network sorts every input}$

How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

$35696^7 > 7 \times 10^{31}$ networks



Build propositional formula satisfiable iff a depth 7 network sorts 11 inputs

Variables: C(i,j,l) – true if comparator (i,j) in level l

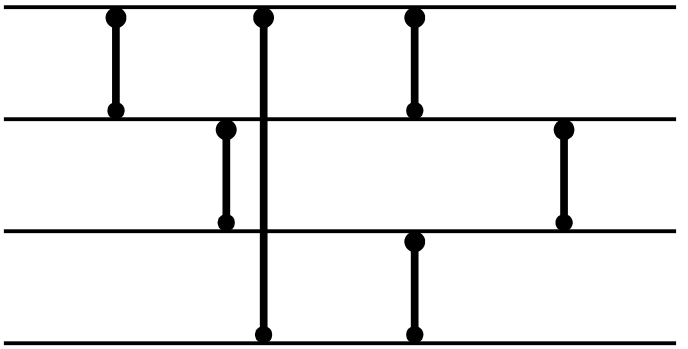
Formula: $\left(\bigwedge_{\substack{\text{line} \\ 1 \le i \le 11}} \bigwedge_{\substack{\text{layer} \\ 1 \le l \le 7}} i \text{ used at most once at layer } l \right) \bigwedge \text{network sorts every input}$

Lemma: Network sorts every input iff it sorts every binary input

How to show that no network of depth 7 sorts 11 inputs?

35696 candidates for each layer

$35696^7 > 7 \times 10^{31}$ networks



Build propositional formula satisfiable iff a depth 7 network sorts 11 inputs

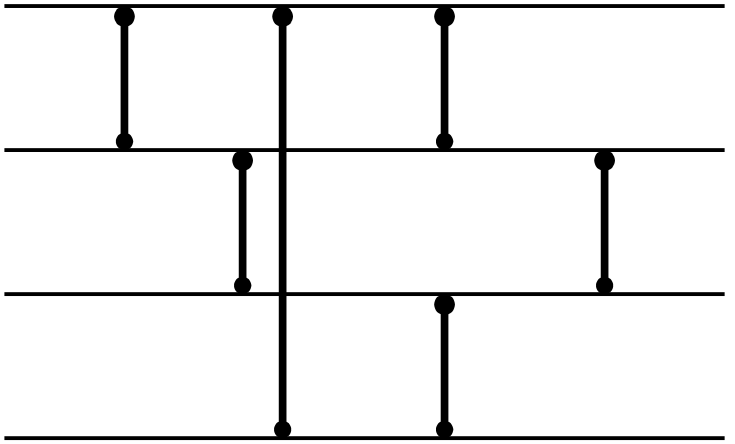
Variables: C(i,j,l) – true if comparator (i,j) in level l

Formula: $\left(\bigwedge_{\substack{\text{line} \\ 1 \le i \le 11}} \bigwedge_{\substack{\text{layer} \\ 1 \le l \le 7}} i \text{ used at most once at layer } l \right) \bigwedge \text{network sorts every input}$

Lemma: Network sorts every input iff it sorts every binary input

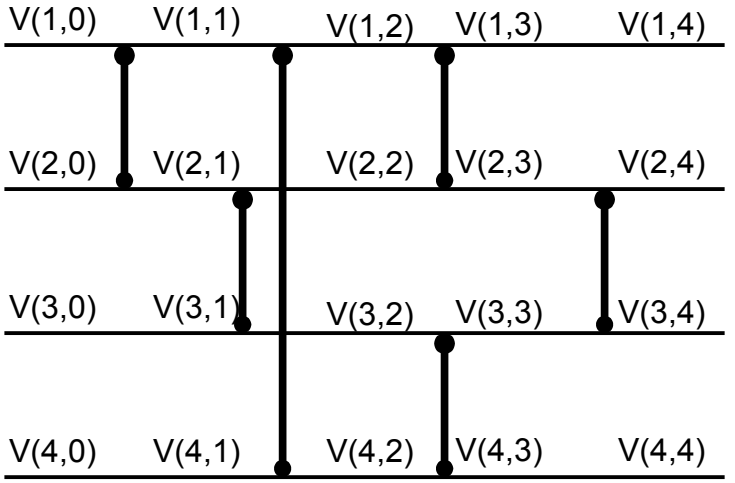
Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)

sorts(01101001011) =



Variables: V(i,l) – value of line i at level l

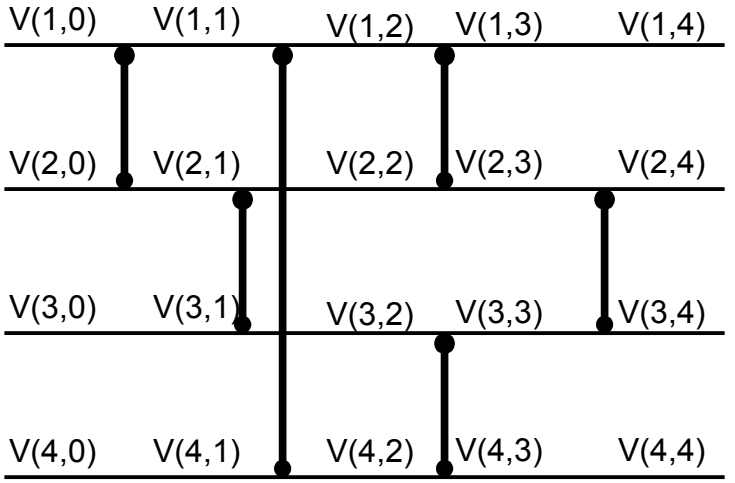
sorts(01101001011) =



Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

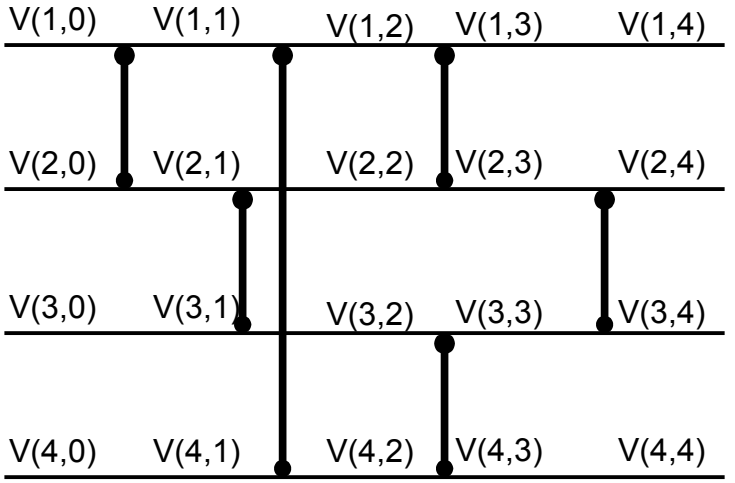


Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

if C(1,2,1) then V(1,1) = min(V(1,0),V(2,0))
V(2,1) = max(V(1,0),V(2,0))
if C(1,3,1) then V(1,1) = min(V(1,0),V(3,0))
V(3,1) = max(V(1,0),V(3,0))
...

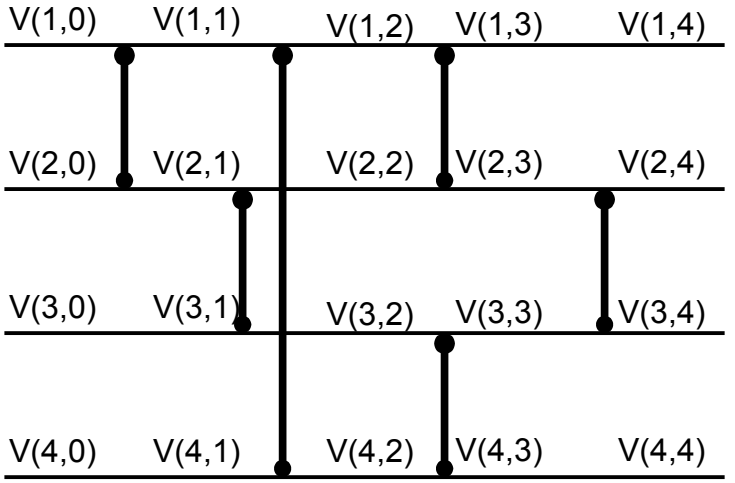


Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

if C(1,2,1) then V(1,1) = min(V(1,0),V(2,0))
V(2,1) = max(V(1,0),V(2,0))
if C(1,3,1) then V(1,1) = min(V(1,0),V(3,0))
V(3,1) = max(V(1,0),V(3,0))
...



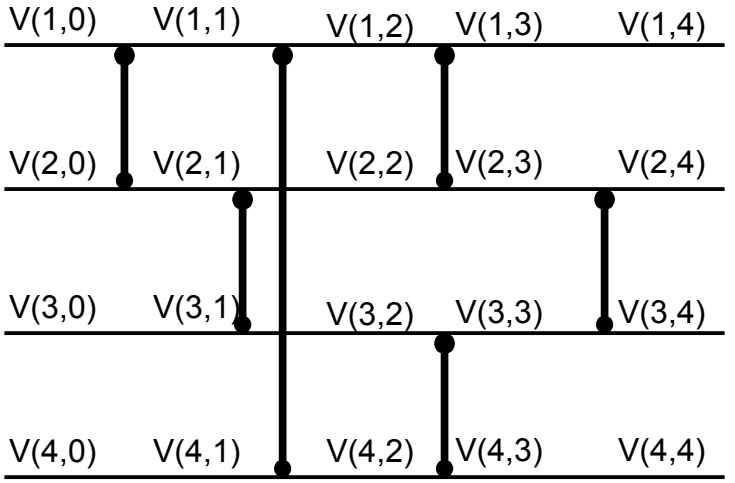
min(x,y) = x and y
max(x,y) = x or y

Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

if C(1,2,1) then V(1,1) = min(V(1,0),V(2,0))
V(2,1) = max(V(1,0),V(2,0))
if C(1,3,1) then V(1,1) = min(V(1,0),V(3,0))
V(3,1) = max(V(1,0),V(3,0))
...



min(x,y) = x and y
max(x,y) = x or y

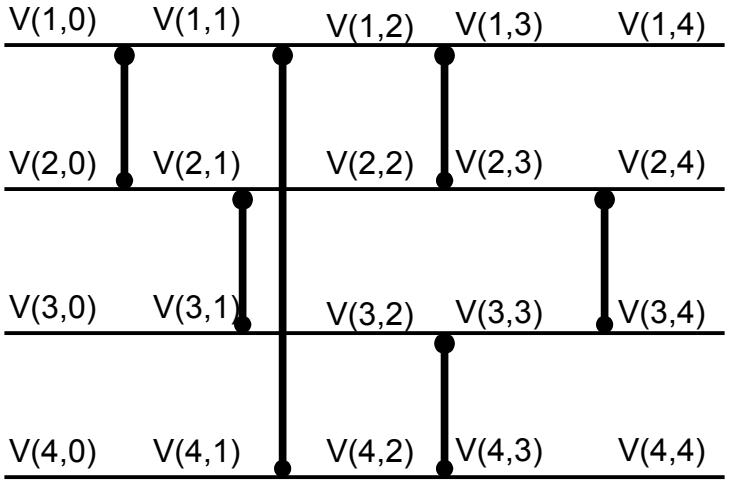
Can build formula F(n,d) which is satisfiable iff some network of depth d sorts n inputs

Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

if C(1,2,1) then V(1,1) = min(V(1,0),V(2,0))
V(2,1) = max(V(1,0),V(2,0))
if C(1,3,1) then V(1,1) = min(V(1,0),V(3,0))
V(3,1) = max(V(1,0),V(3,0))
...



min(x,y) = x and y
max(x,y) = x or y

Can build formula F(n,d) which is satisfiable iff some network of depth d sorts n inputs

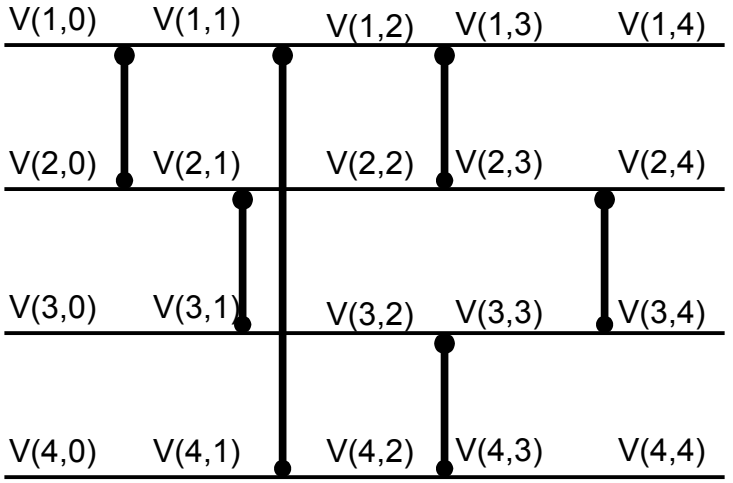
n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
d	2	2	4	4	5	5	6	6	7	7	8	8	8	8
UNSAT	<1s	<1s	<1s	<1s	2s	2s	15m	29m	-	-	-	-	-	-
d	3	3	5	5	6	6	7	7	8	8	9	9	9	9
SAT	<1s	<1s	<1s	<1s	<1s	1s	1m	16m	21m	-	-	-	-	-

Variables: V(i,l) – value of line i at level l

sorts(01101001011) =

V(_,0) = 01101001011 and
V(_,7) = 00000111111 and

if C(1,2,1) then V(1,1) = min(V(1,0),V(2,0))
V(2,1) = max(V(1,0),V(2,0))
if C(1,3,1) then V(1,1) = min(V(1,0),V(3,0))
V(3,1) = max(V(1,0),V(3,0))
...

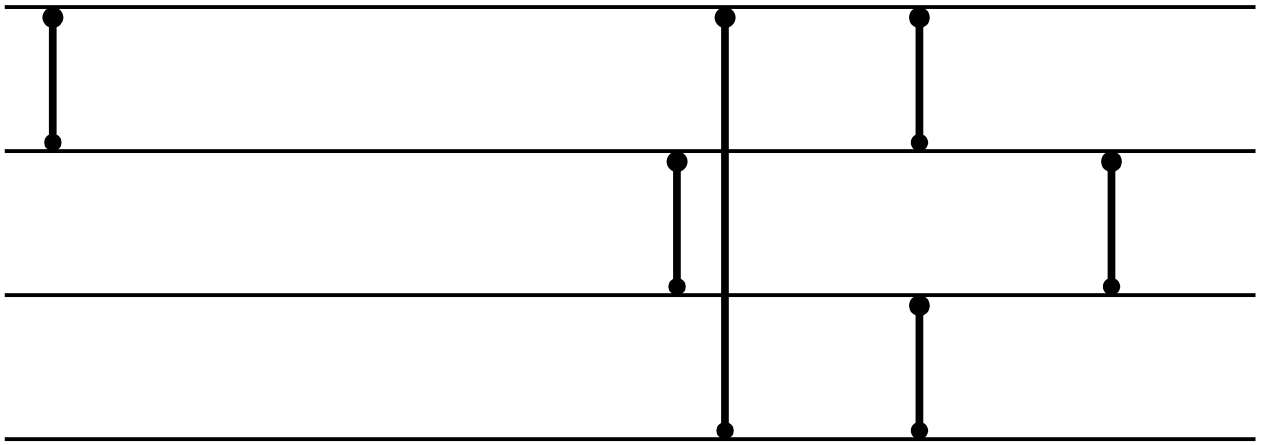


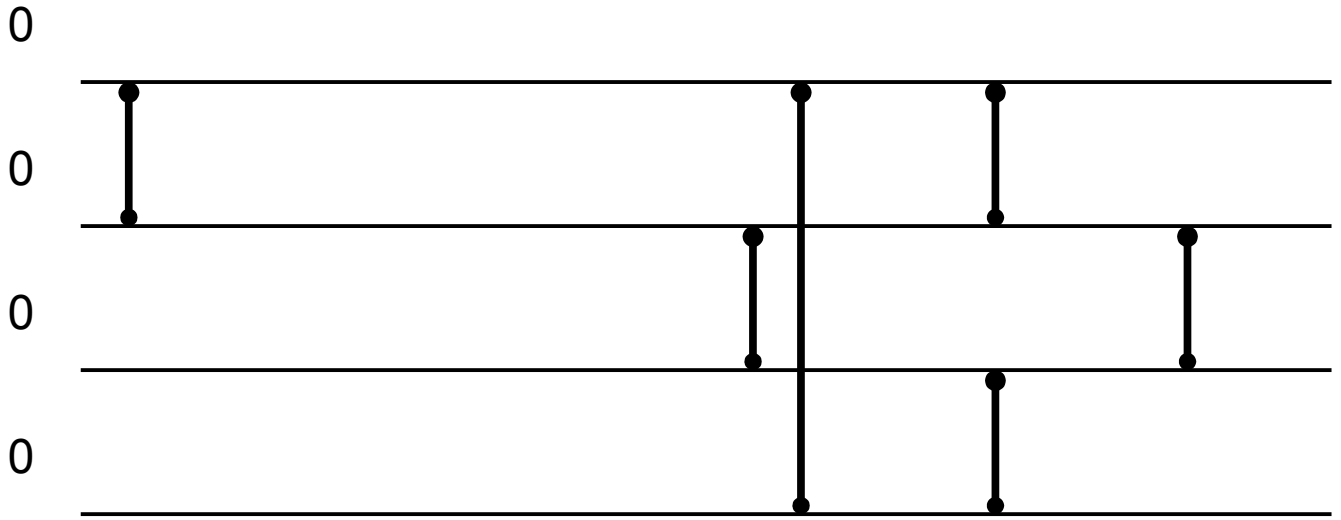
min(x,y) = x and y
max(x,y) = x or y

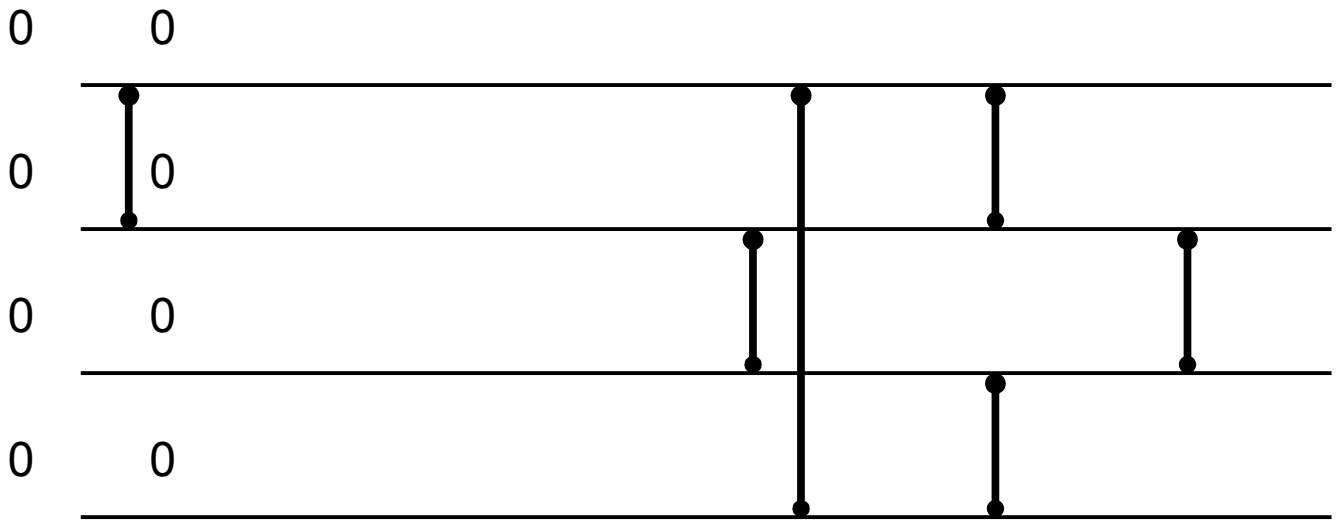
Can build formula F(n,d) which is satisfiable iff some network of depth d sorts n inputs

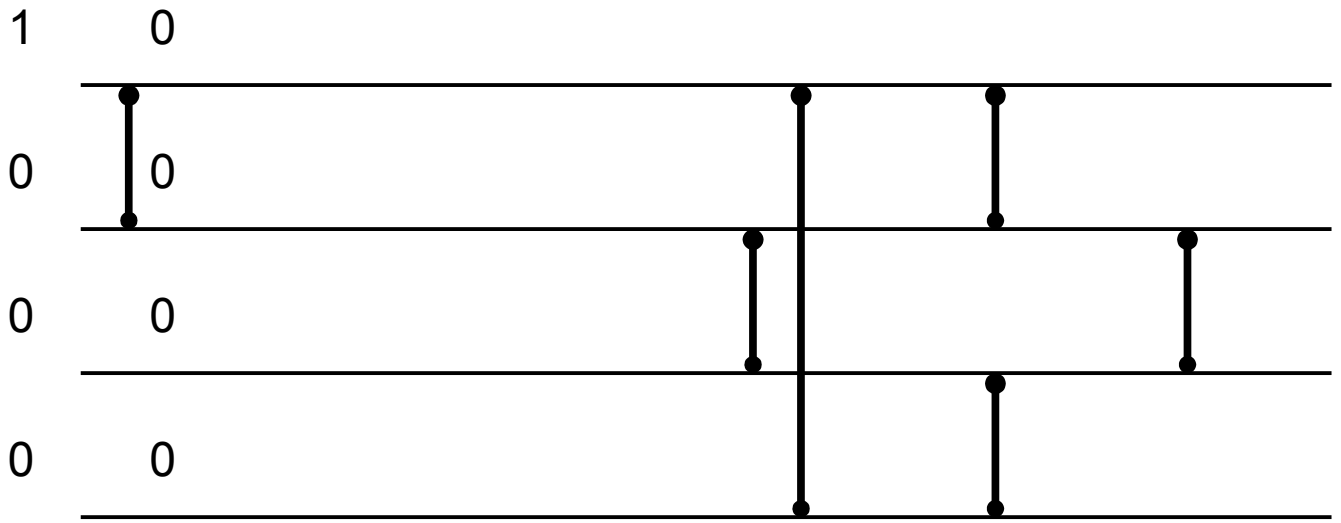
n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
d	2	2	4	4	5	5	6	6	7	7	8	8	8	8
UNSAT	<1s	<1s	<1s	<1s	2s	2s	15m	29m	-	-	-	-	-	-
d	3	3	5	5	6	6	7	7	8	8	9	9	9	9
SAT	<1s	<1s	<1s	<1s	<1s	1s	1m	16m	21m	-	-	-	-	-

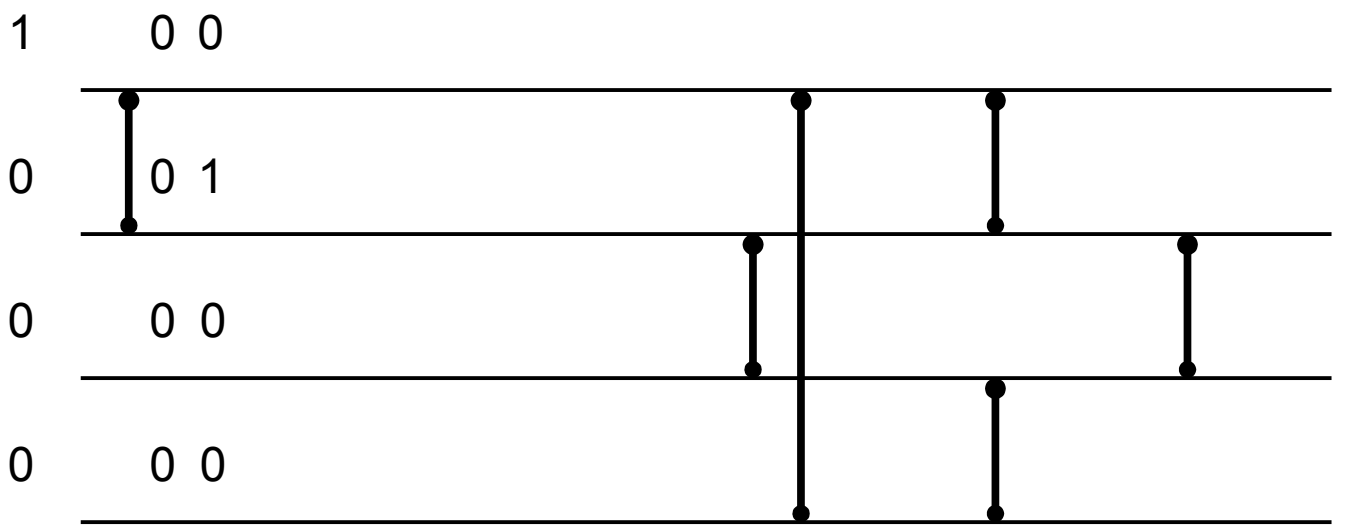
How to restrict set of possible networks?

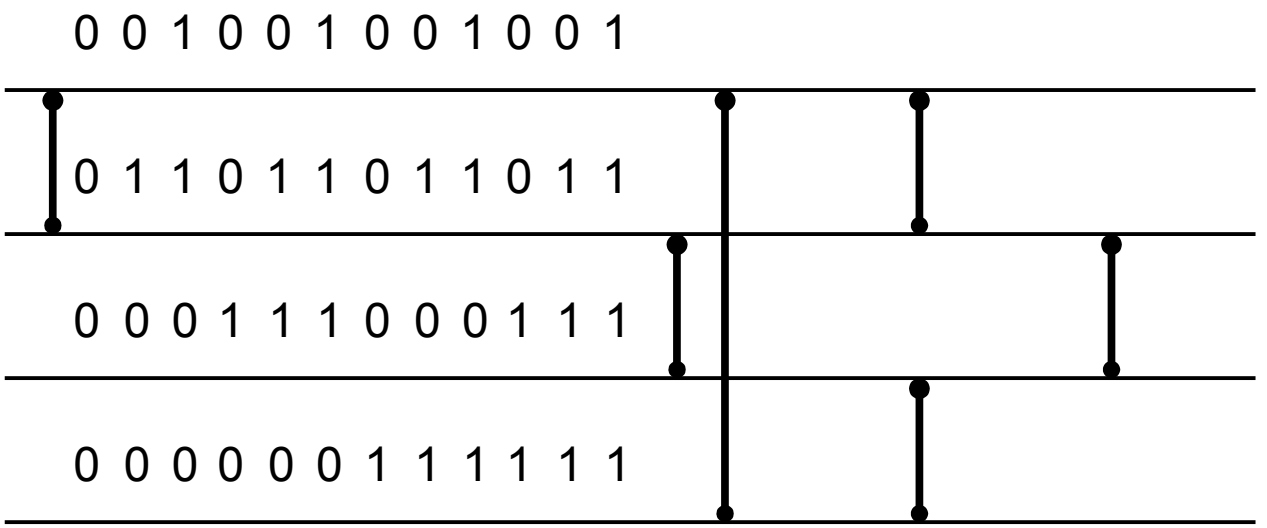


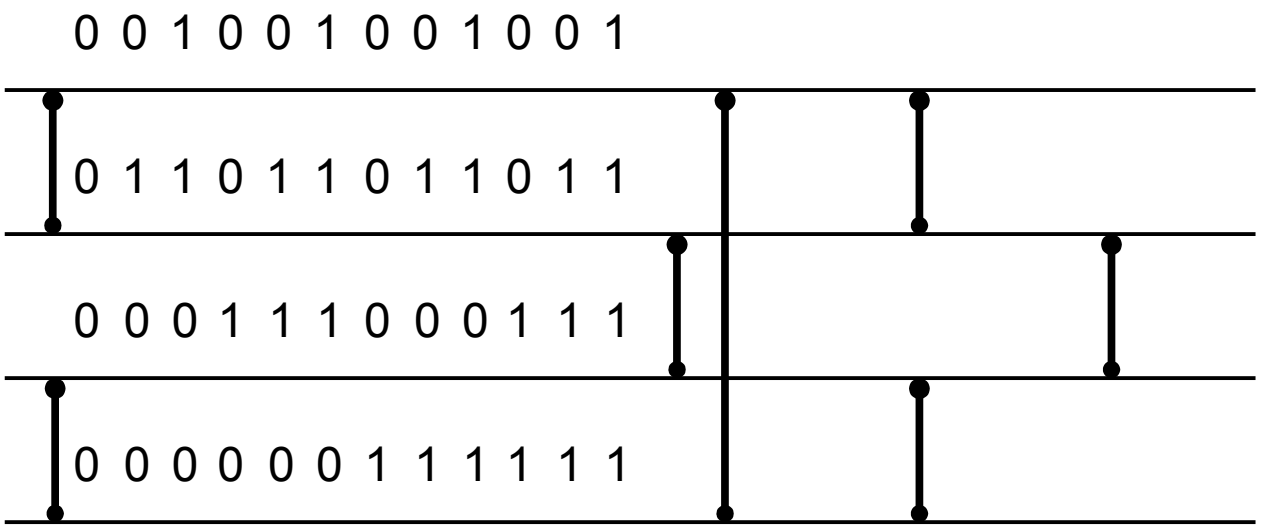

0
0
0
0

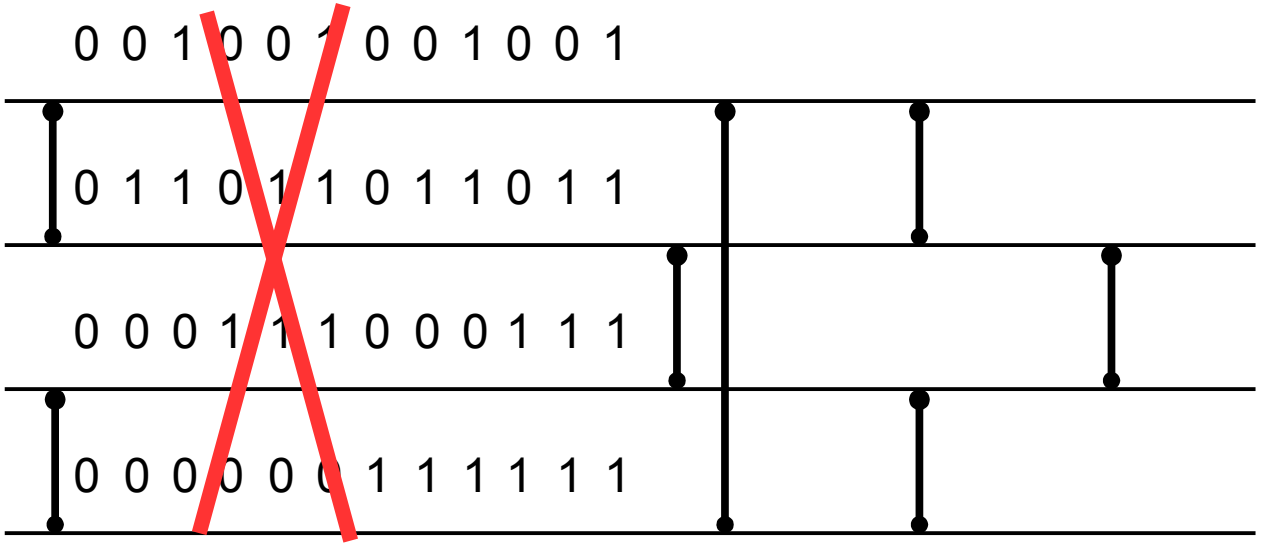

0
0
0
0
0
0
0
0

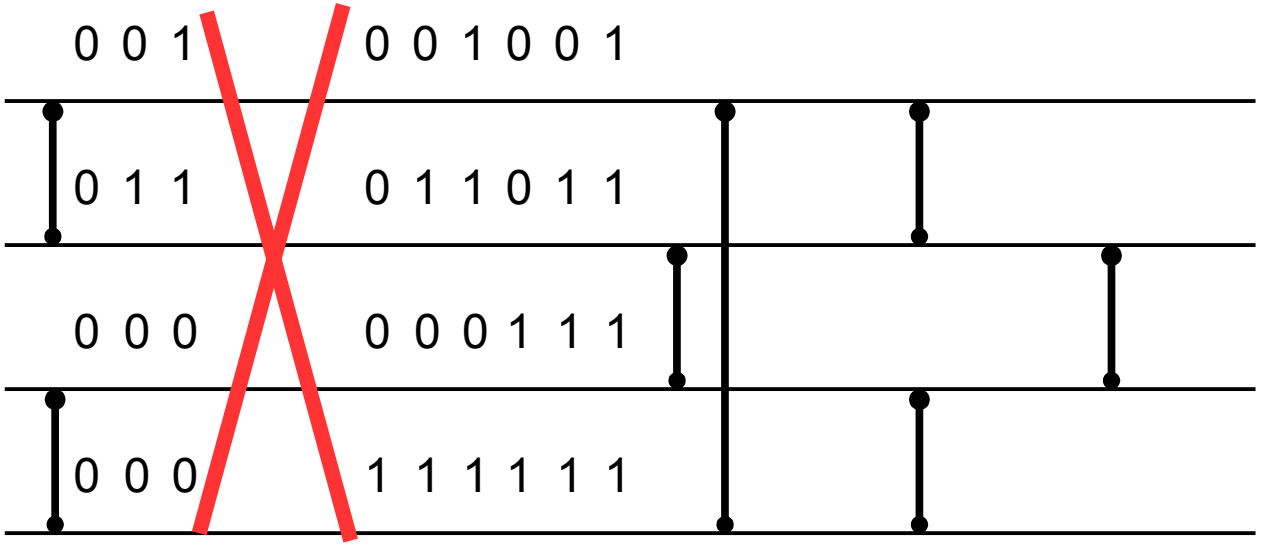

1
0
0
0
0
0
0
0

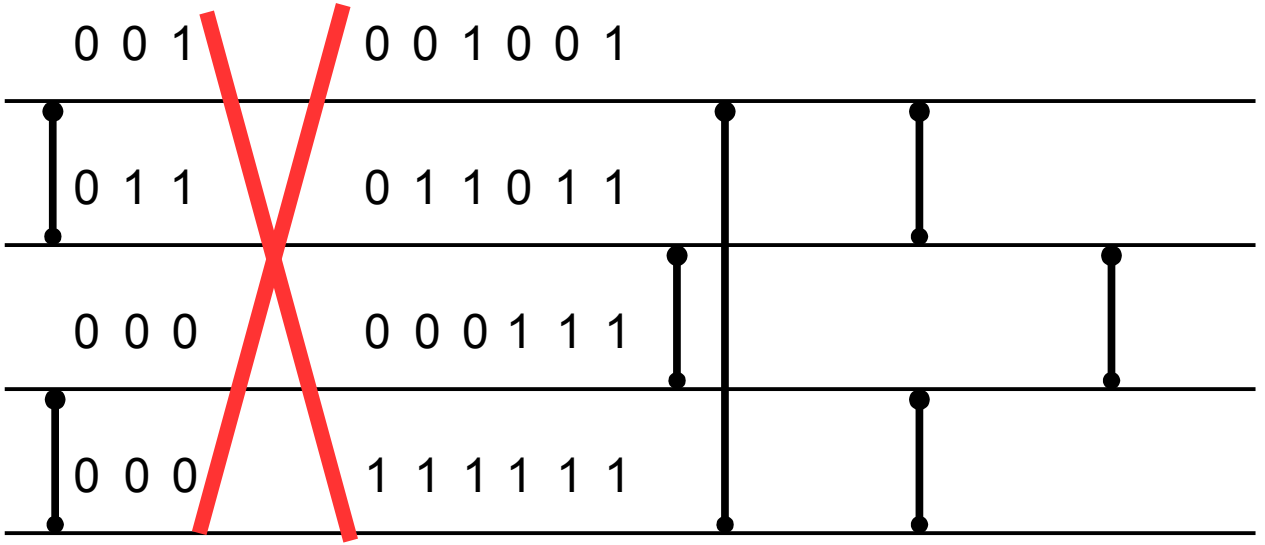

1
0 0
0
0 1
0
0 0
0
0 0


0 0 1 0 0 1 0 0 1 0 0 1
0 1 1 0 1 1 0 1 1 0 1 1
0 0 0 1 1 1 0 0 0 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1

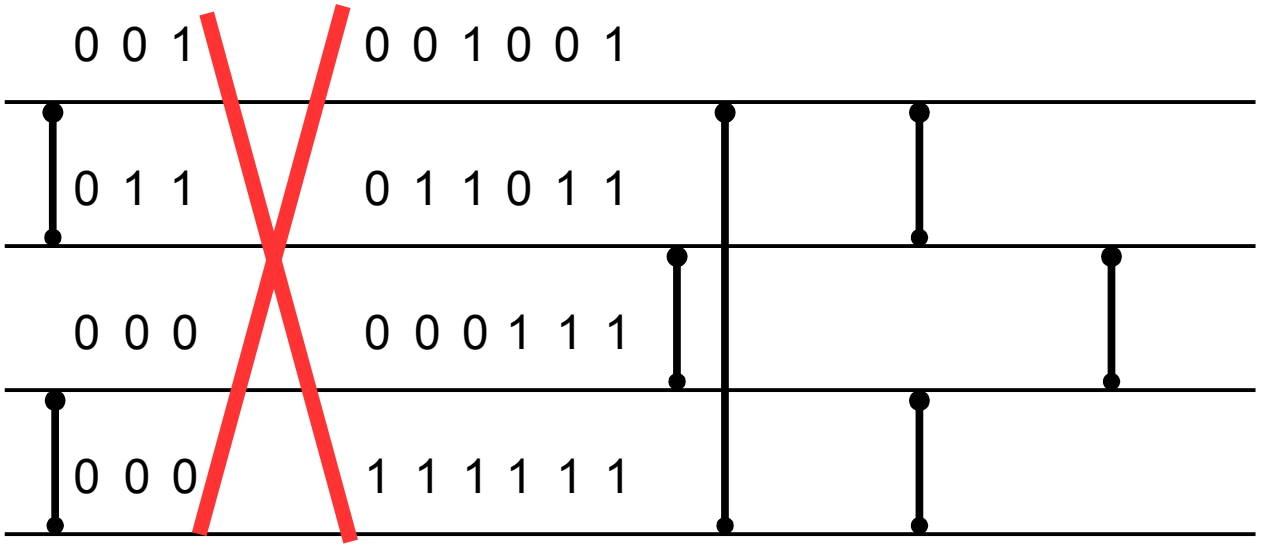

0 0 1 0 0 1 0 0 1 0 0 1
0 1 1 0 1 1 0 1 1 0 1 1
0 0 0 1 1 1 0 0 0 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1


0 0 1 0 0 1 0 0 1 0 0 1
0 1 1 0 1 1 0 1 1 0 1 1
0 0 0 1 1 1 0 0 0 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1

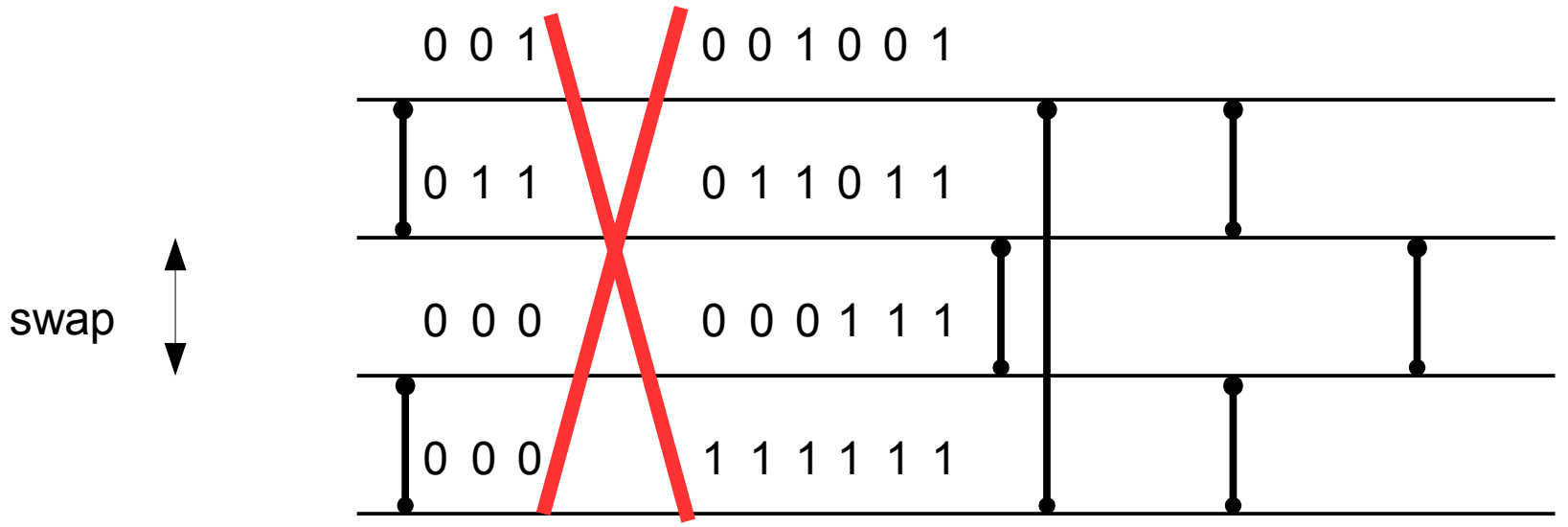

0 0 1
0 0 1 0 0 1
0 1 1
0 1 1 0 1 1
0 0 0
0 0 0 1 1 1
0 0 0
1 1 1 1 1 1



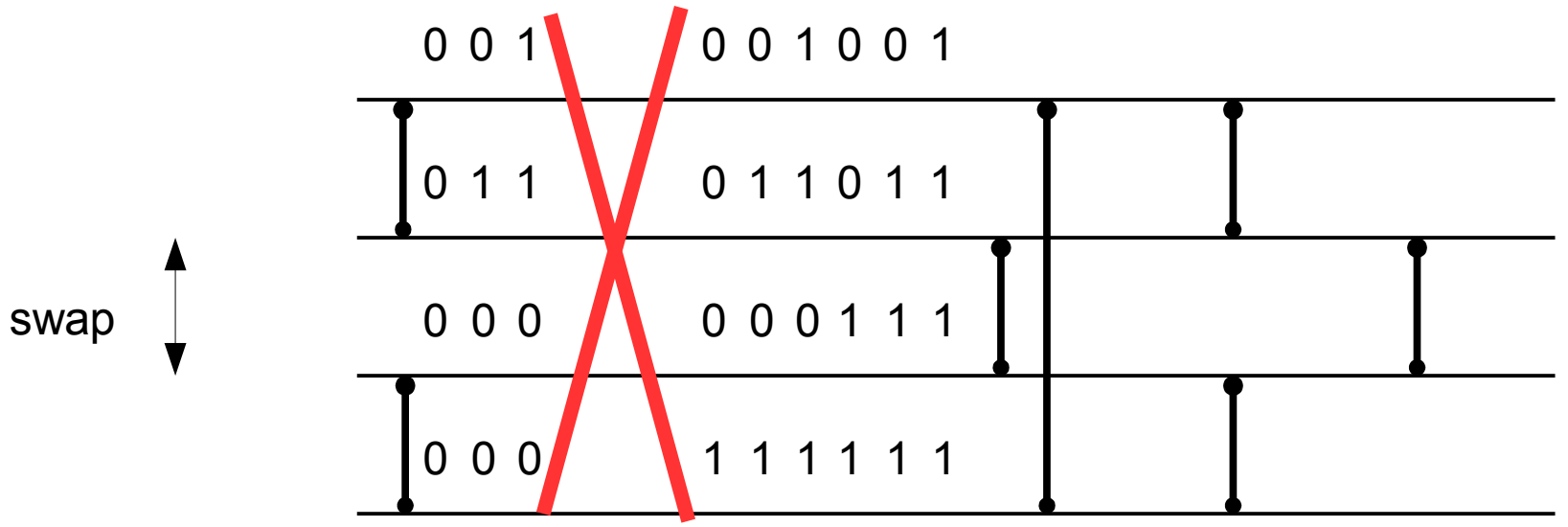
First layer is maximal



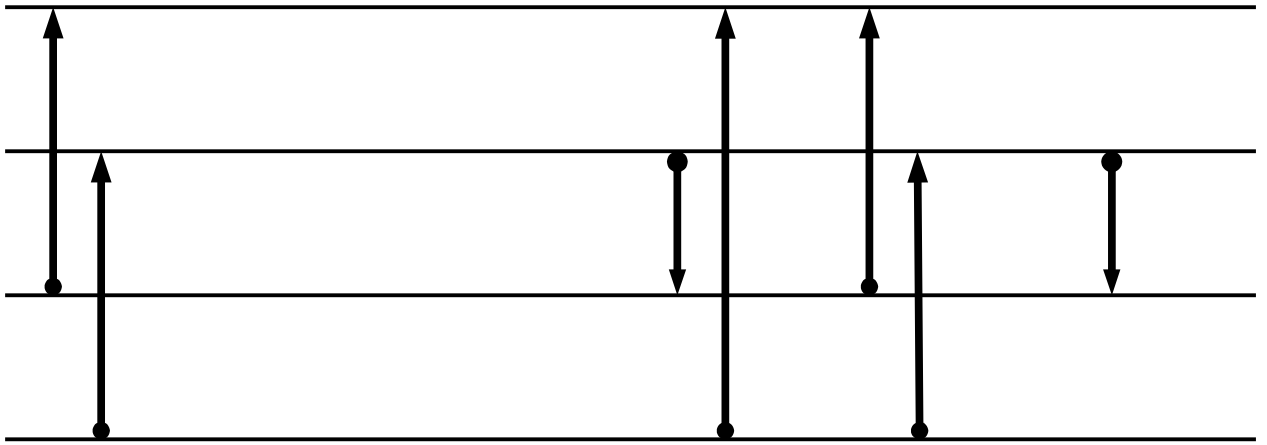
First layer is maximal

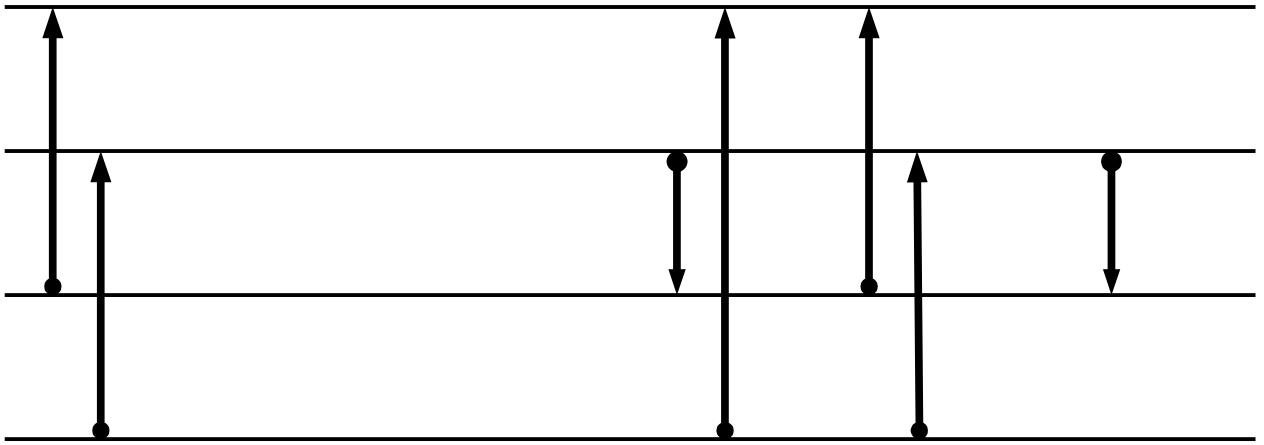


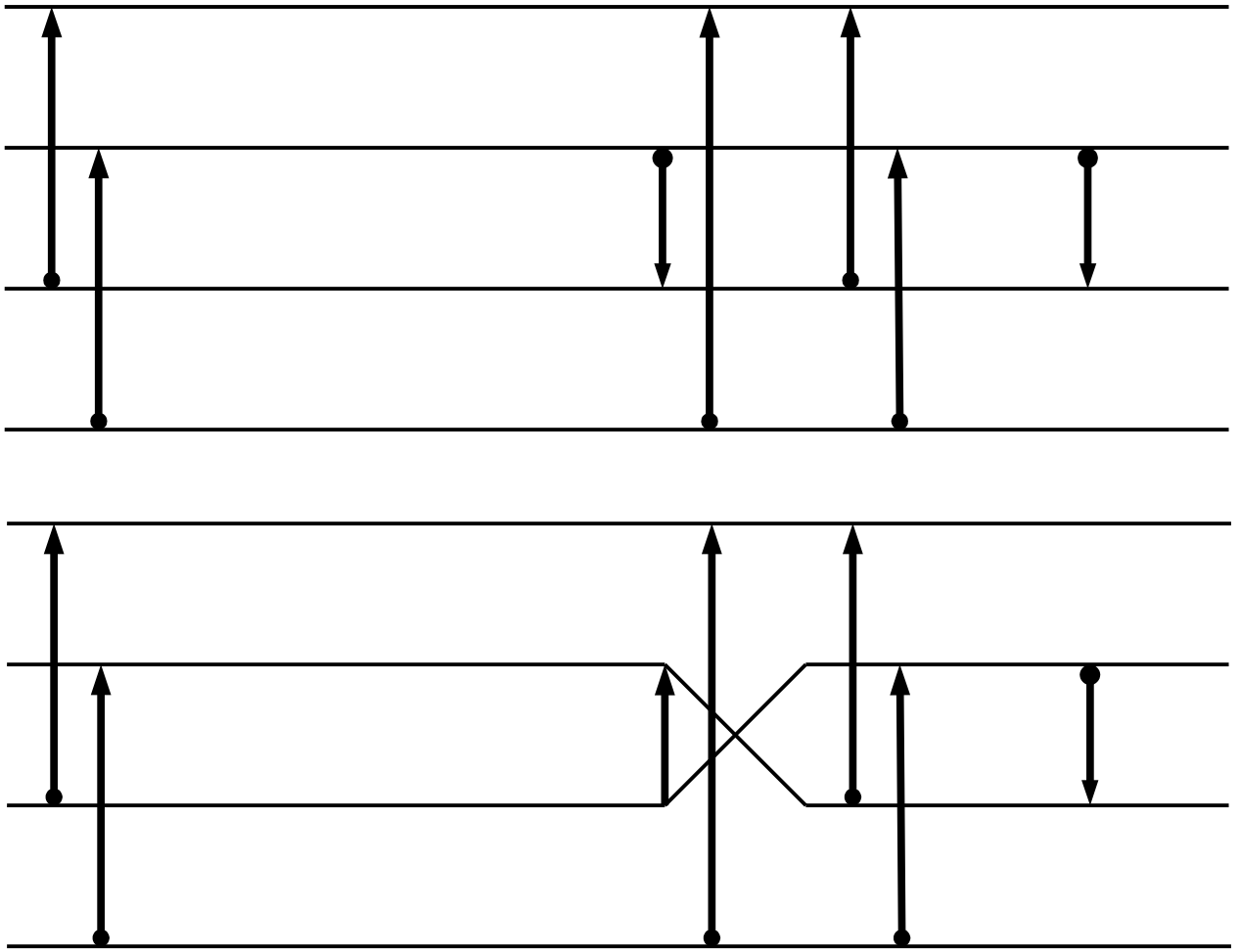
First layer is maximal

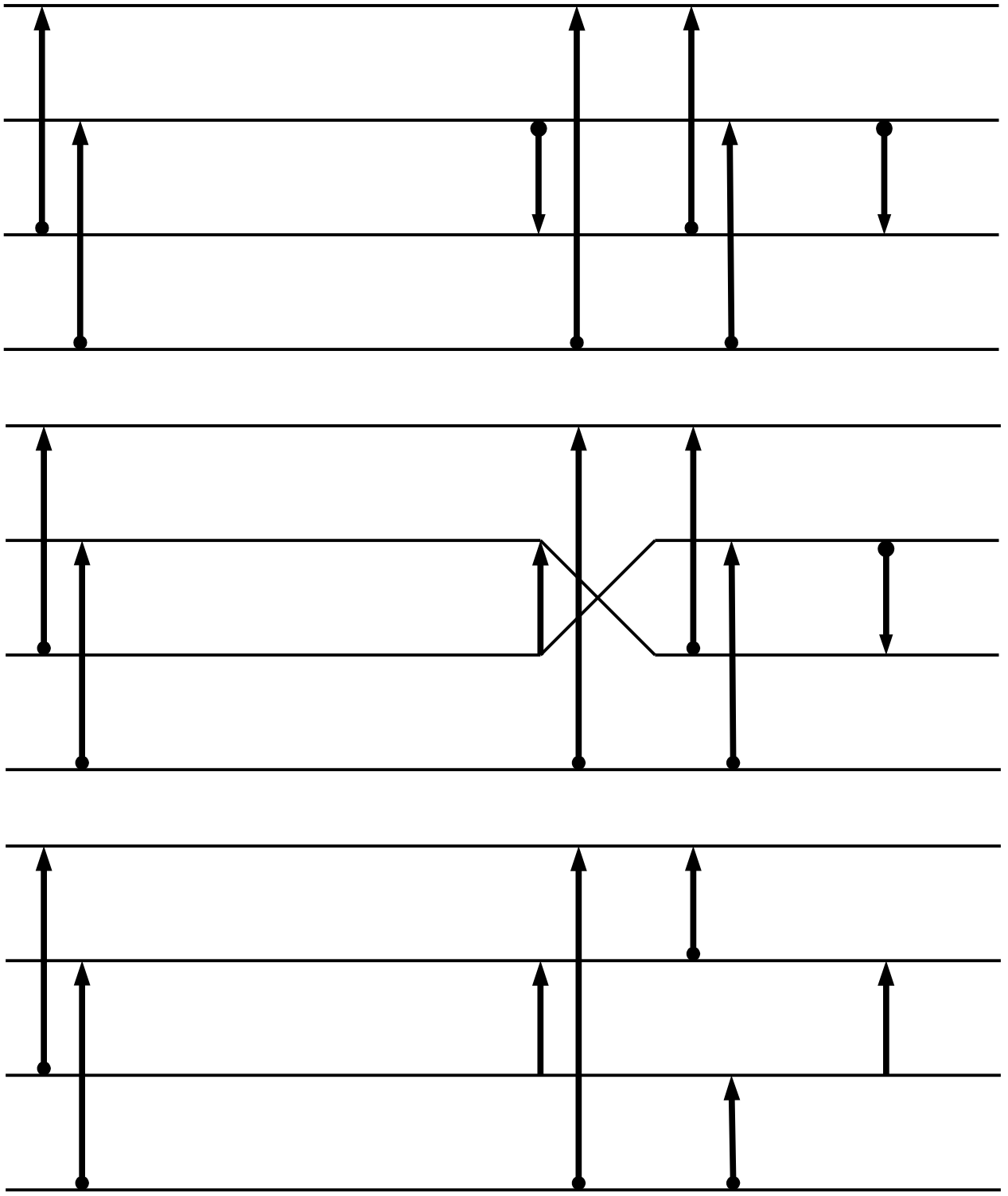


First layer is maximal

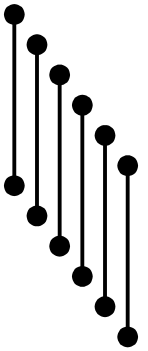


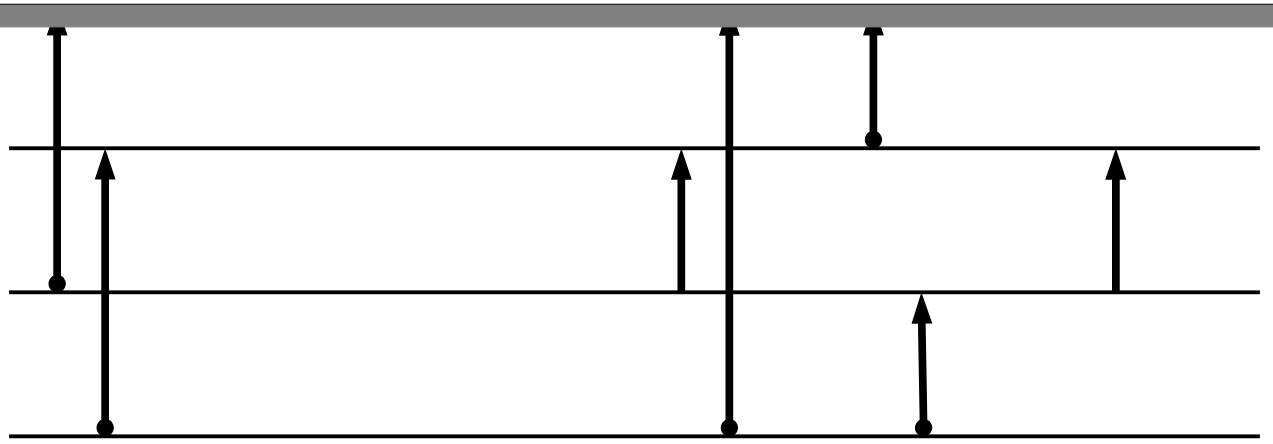




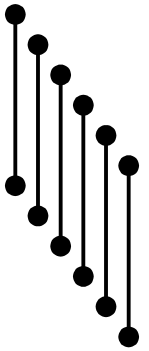


First layer is fixed:



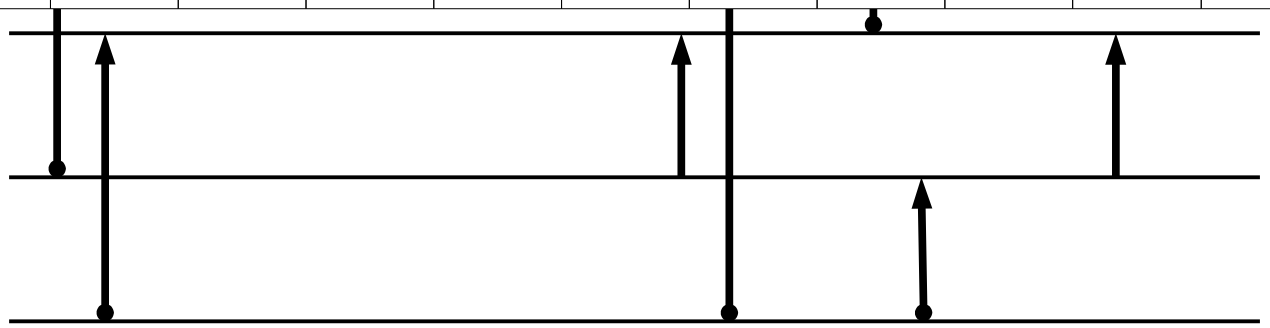


First layer is fixed:

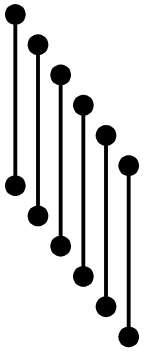


Can set C(i,j,1) variables for the first layer

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
d	2	2	4	4	5	5	6	6	7	7	8	8	8	8
UNSAT	<1s	<1s	<1s	<1s	2s	2s	15m	29m	-	-	-	-	-	-
First Fixed	<1s	<1s	<1s	<1s	<1s	<1s	2s	10s	13m	23m	-	-	-	-
d	3	3	5	5	6	6	7	7	8	8	9	9	9	9
SAT	<1s	<1s	<1s	<1s	<1s	1s	1m	16m	21m	-	-	-	-	-
First Fixed	<1s	<1s	<1s	<1s	<1s	<1s	1s	19s	4m	12m	-	-	-	-



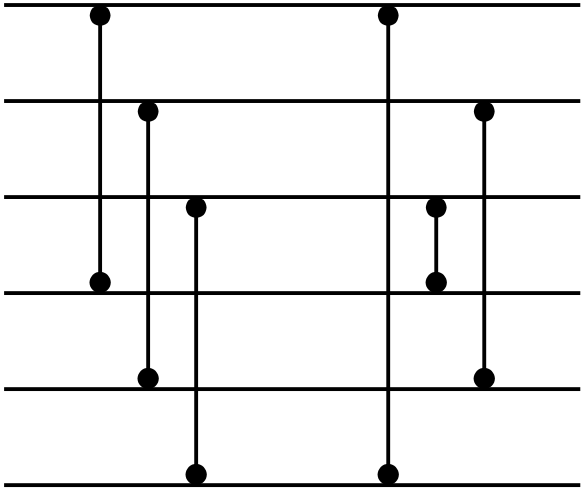
First layer is fixed:



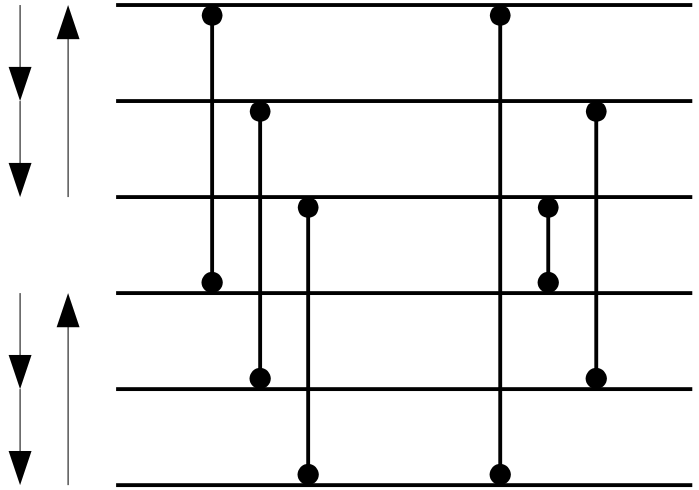
Can set C(i,j,1) variables for the first layer

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
d	2	2	4	4	5	5	6	6	7	7	8	8	8	8
UNSAT	<1s	<1s	<1s	<1s	2s	2s	15m	29m	-	-	-	-	-	-
First Fixed	<1s	<1s	<1s	<1s	<1s	<1s	2s	10s	13m	23m	-	-	-	-
d	3	3	5	5	6	6	7	7	8	8	9	9	9	9
SAT	<1s	<1s	<1s	<1s	<1s	1s	1m	16m	21m	-	-	-	-	-
First Fixed	<1s	<1s	<1s	<1s	<1s	<1s	1s	19s	4m	12m	-	-	-	-

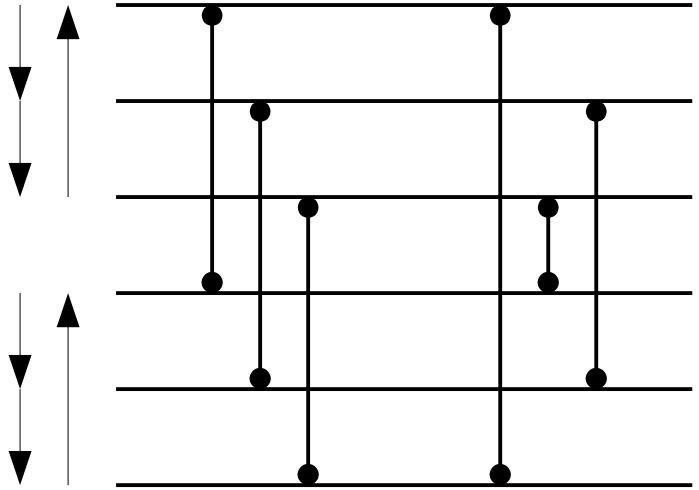
How can we restrict second layers for n=13?

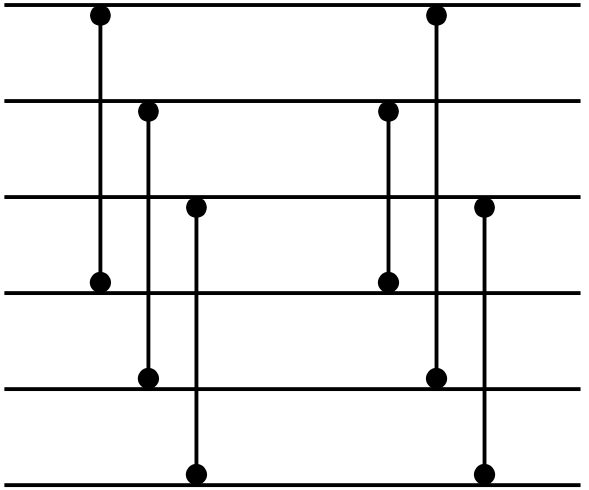


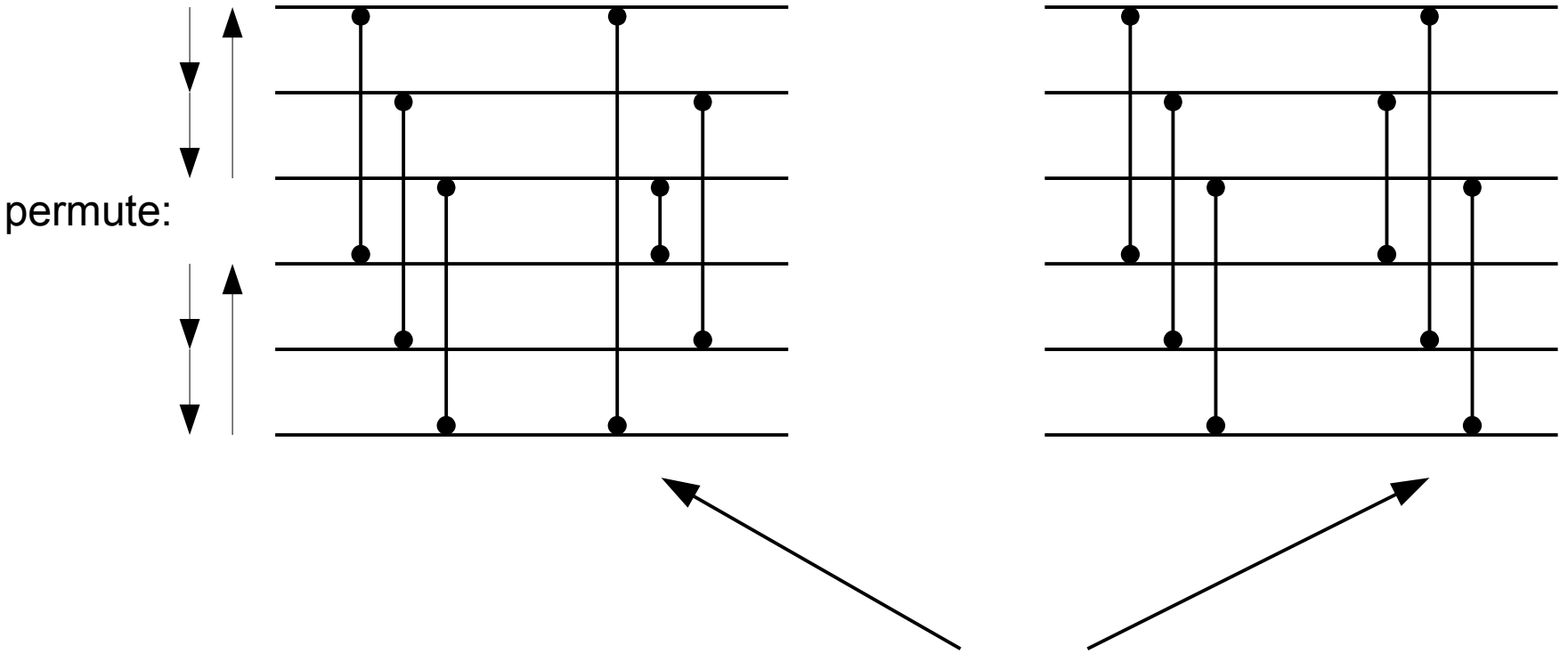
permute:

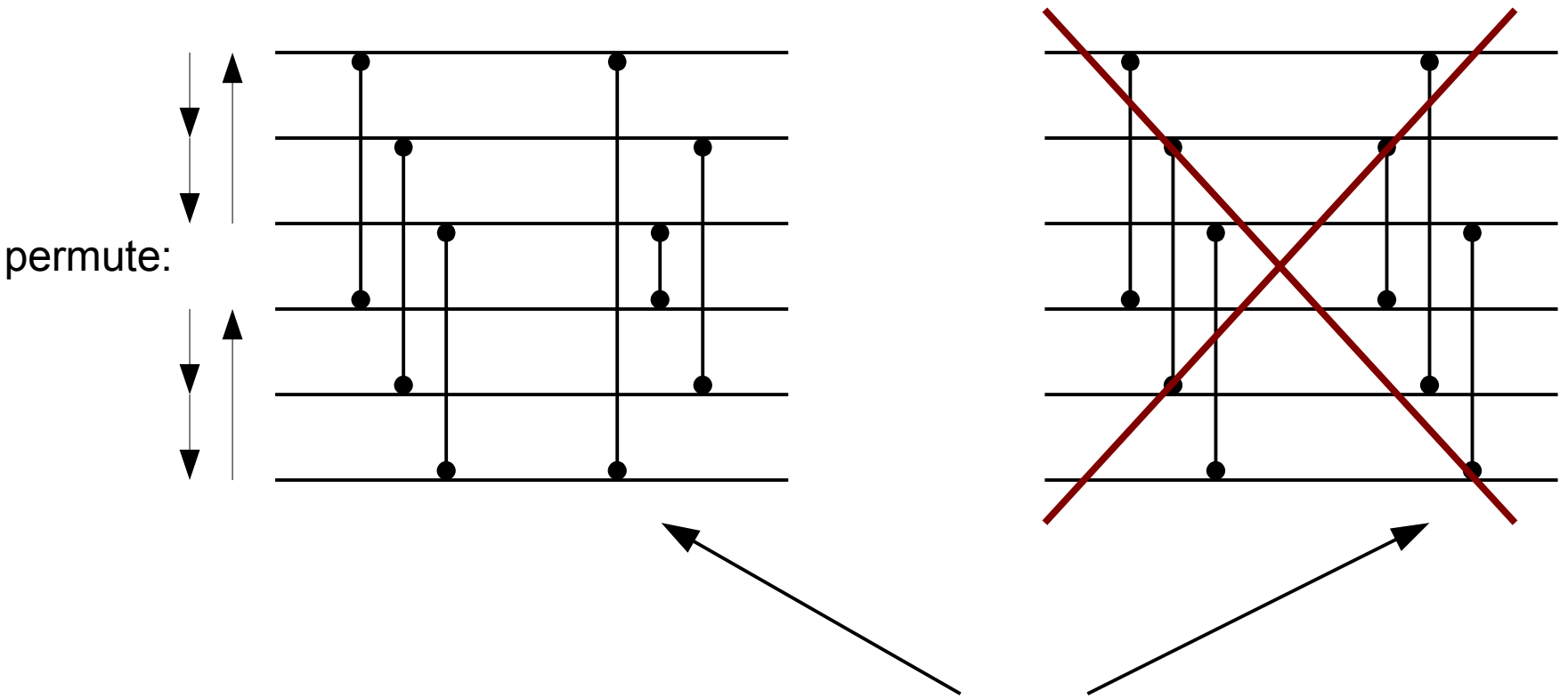


permute:

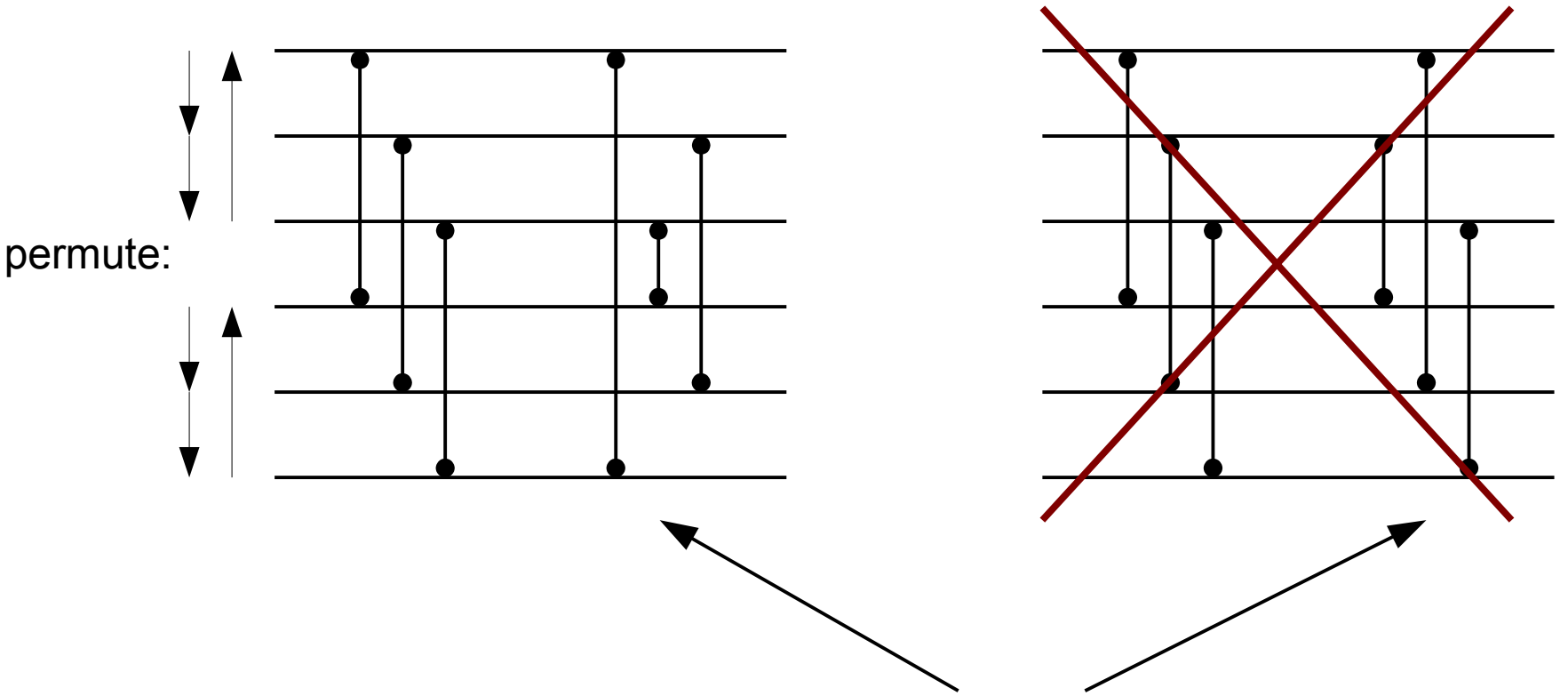





permute:
Equivalent second layers



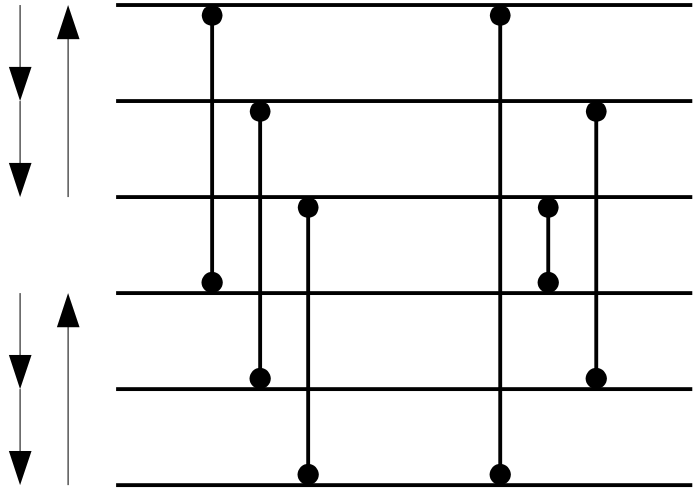
Equivalent second layers



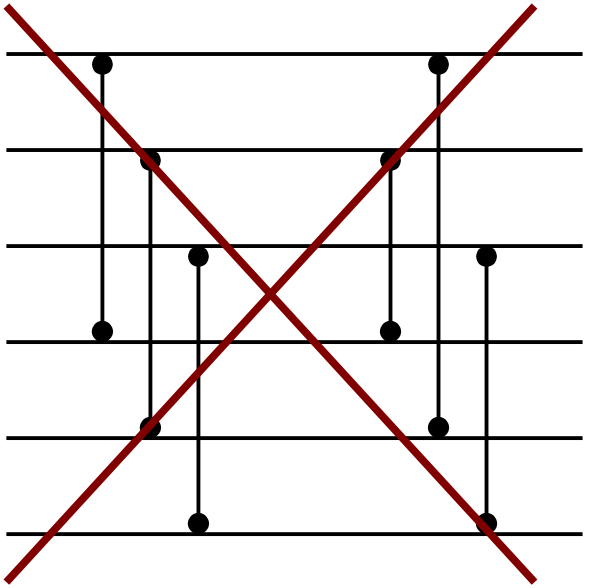
Equivalent second layers

For n=13 568504 candidate layers 4288 equivalence classes of second layers

permute:

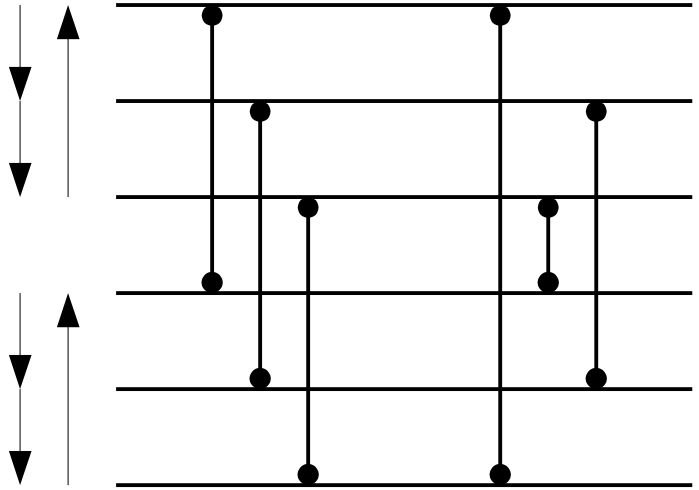


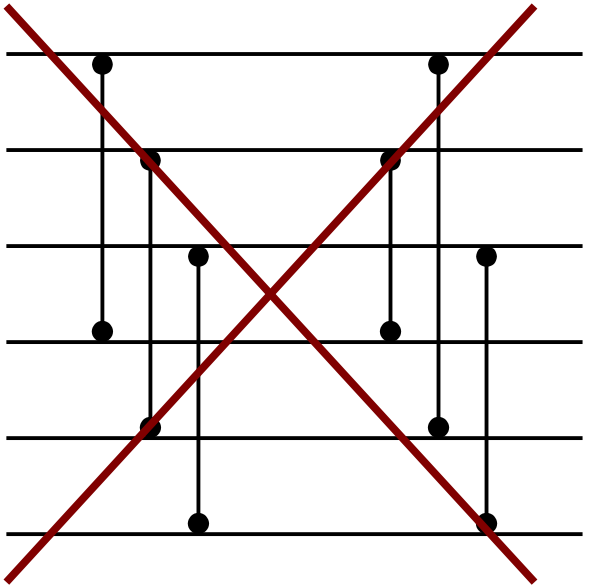
C = π(D)



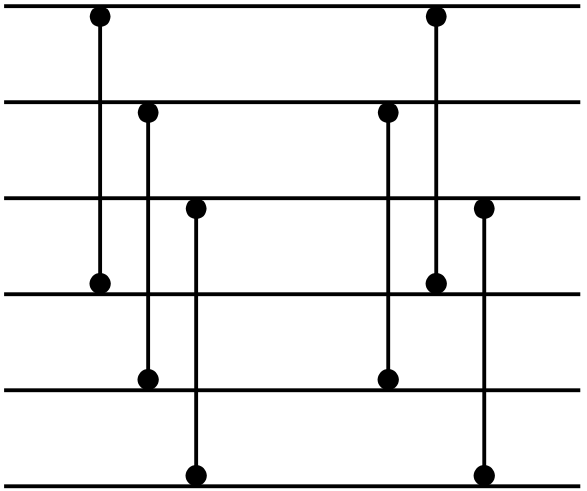
C = π(D)

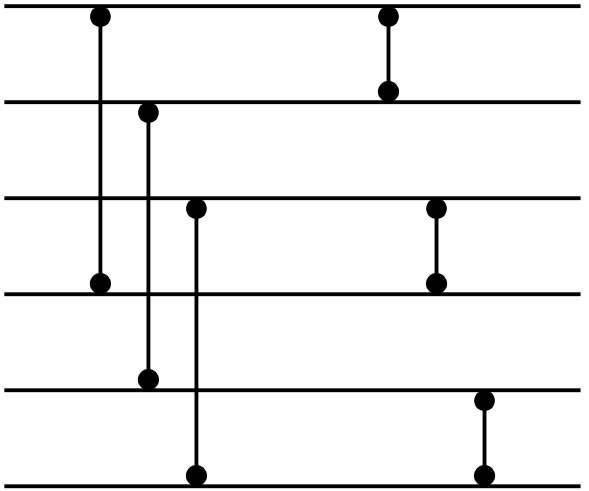
permute:



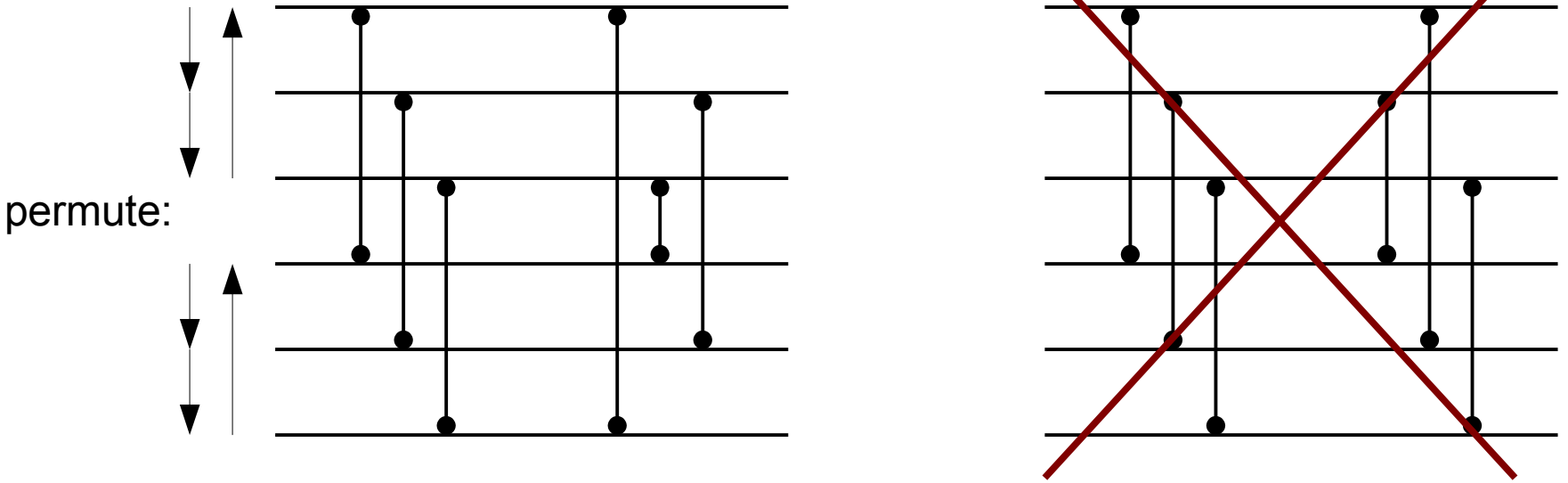


outputs(C) ⊆ outputs(D)

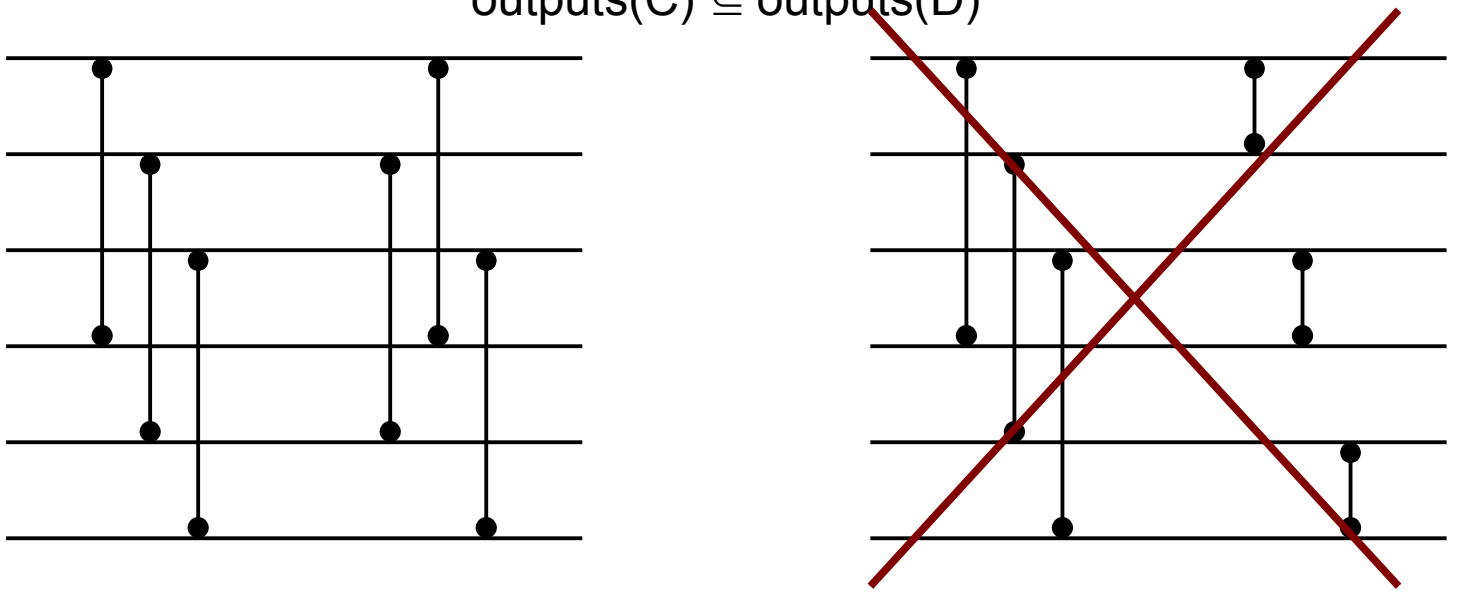




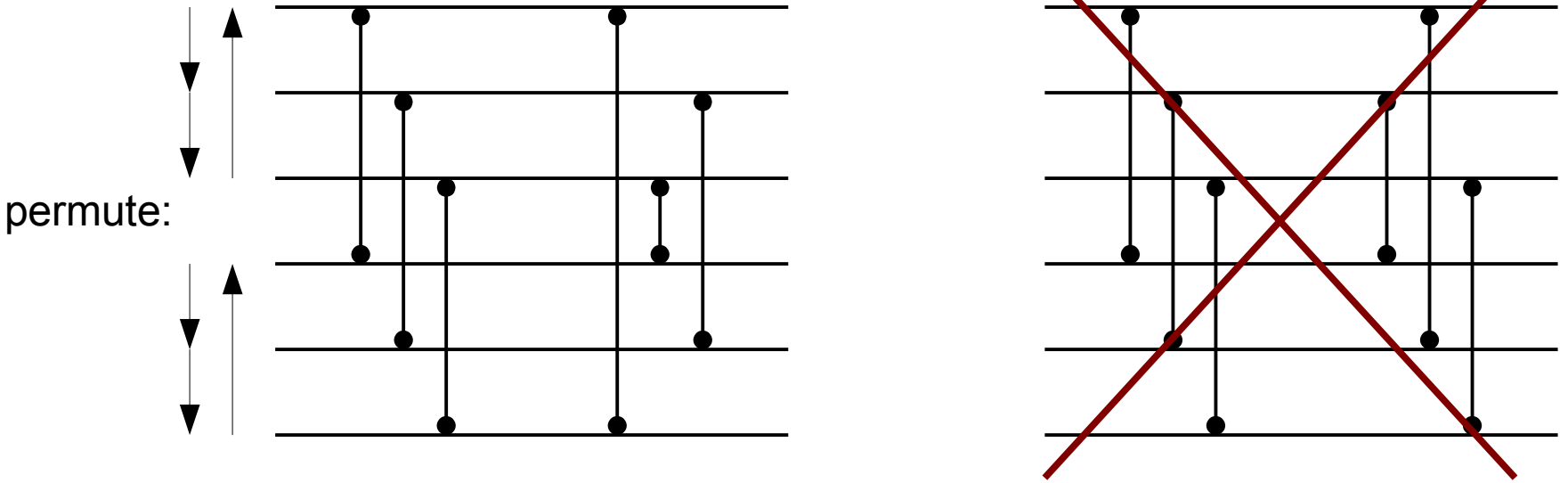
C = π(D)



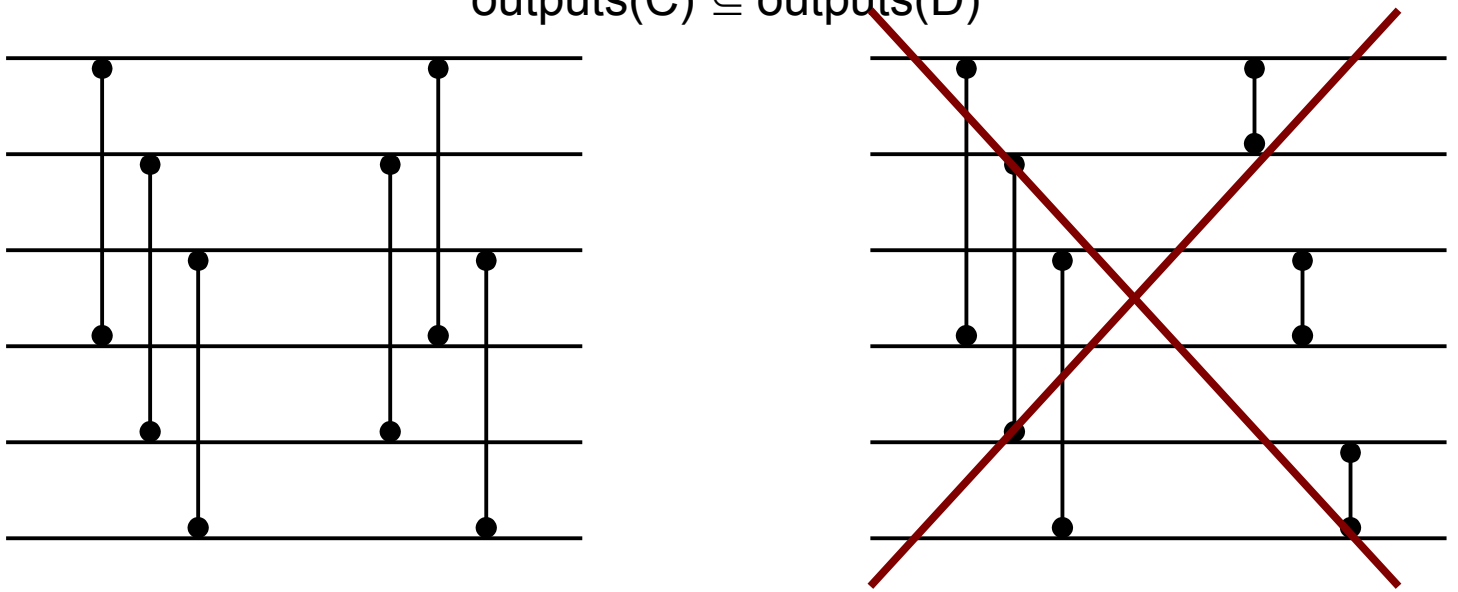
outputs(C) ⊆ outputs(D)



C = π(D)

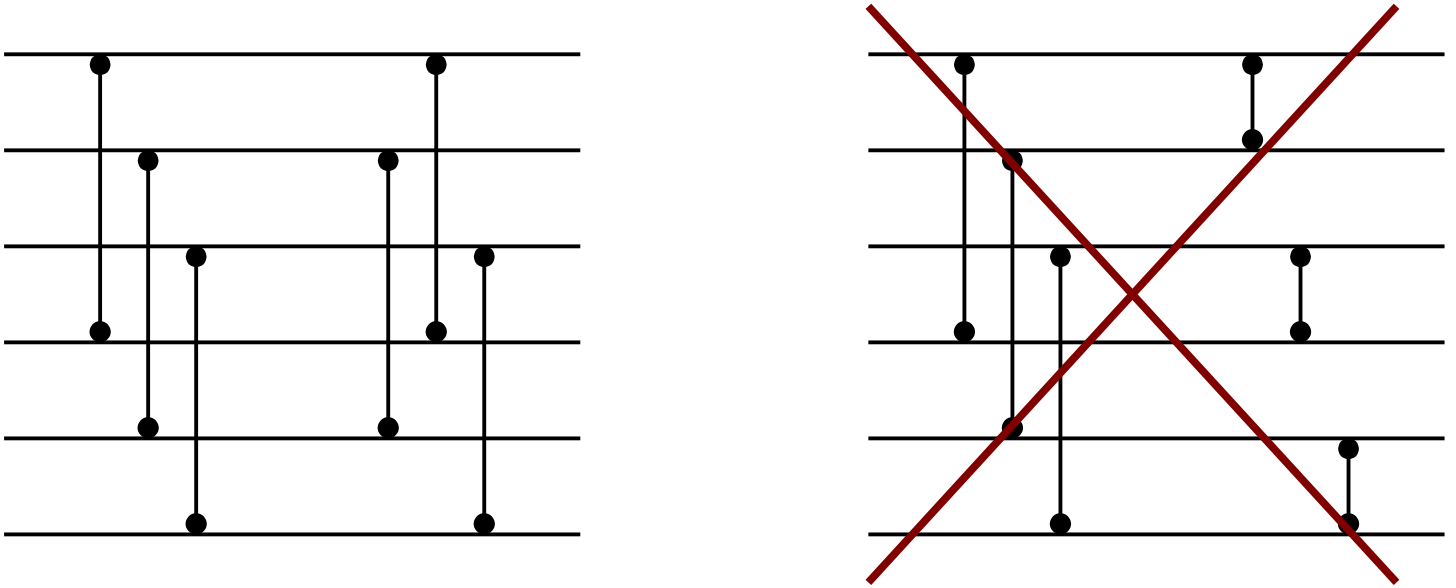


outputs(C) ⊆ outputs(D)



outputs(C) ⊆ π(outputs(D))

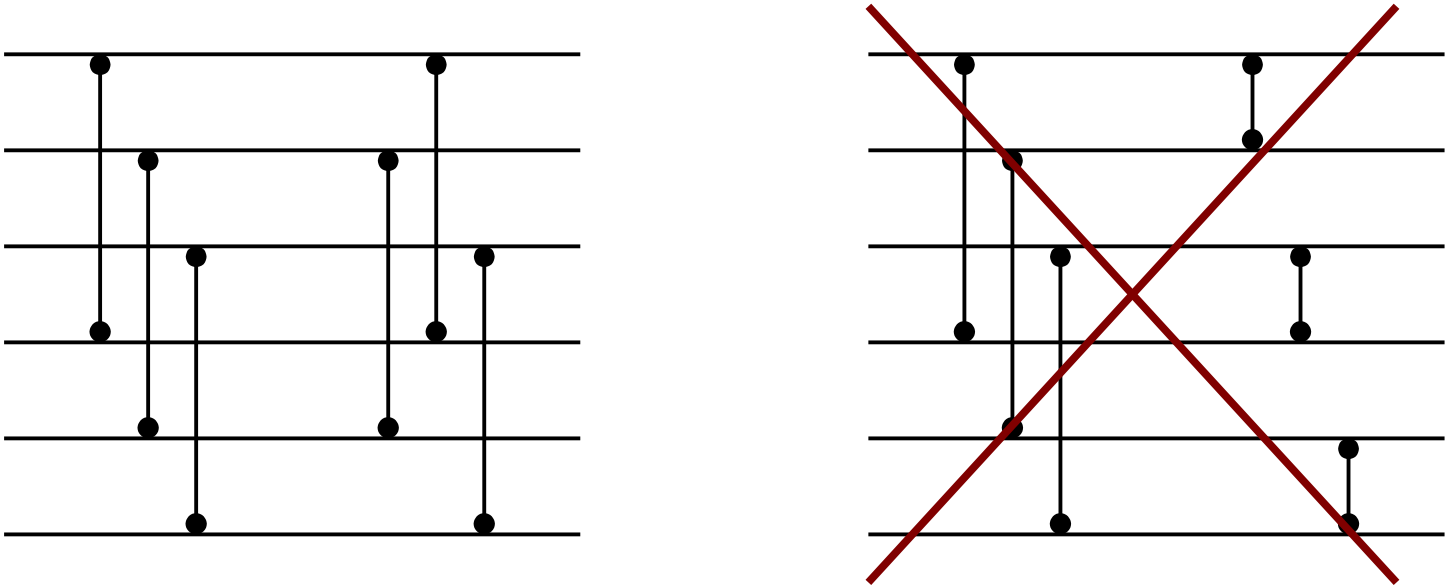
outputs(C) ⊆ π(outputs(D))



outputs(C) ⊆ π(outputs(D))

For n=13 568504 candidate layers

outputs(C) ⊆ π(outputs(D))

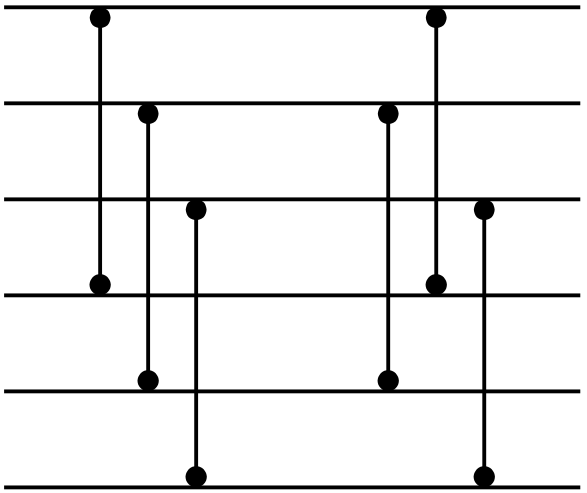


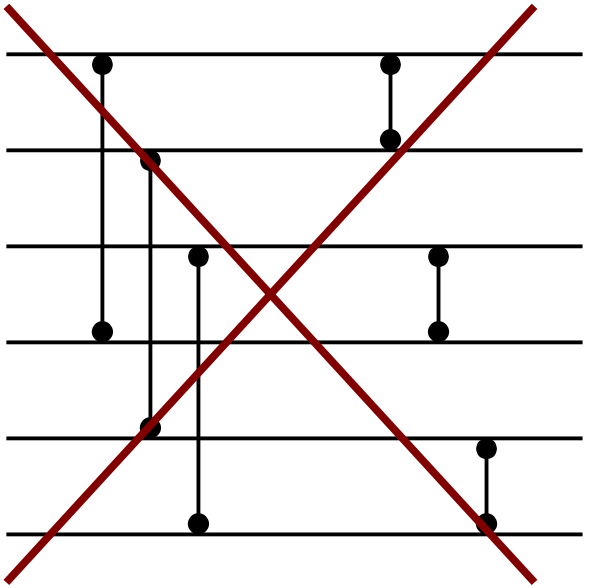
outputs(C) ⊆ π(outputs(D))

For n=13 568504 candidate layers

outputs(C) ⊆ π(outputs(D))

13! ~ 6×10^9 permutations





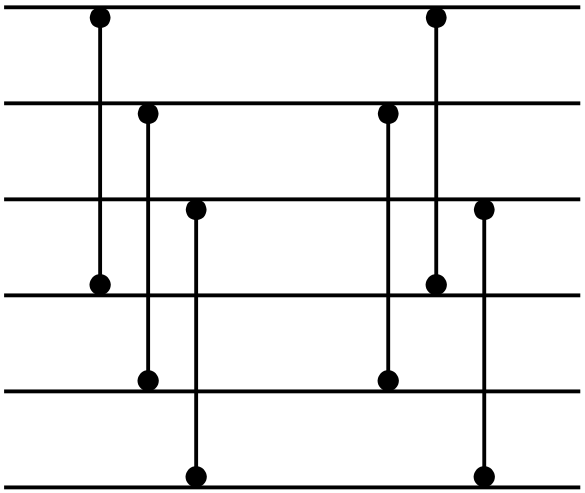
outputs(C) ⊆ π(outputs(D))

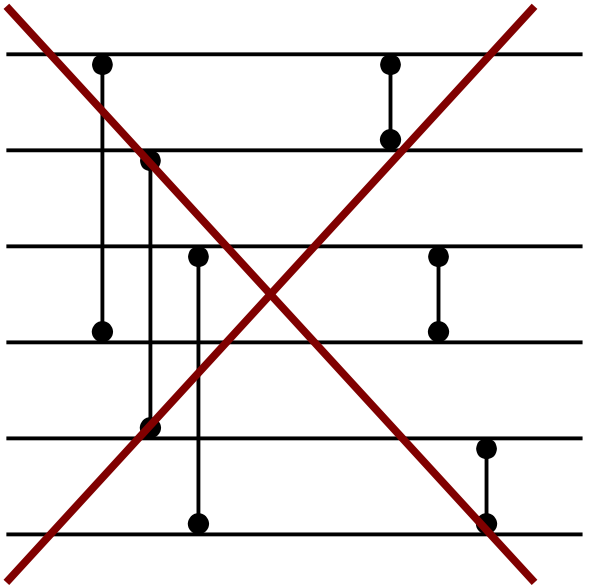
For n=13 568504 candidate layers

outputs(C) ⊆ π(outputs(D))

~1000 size of outputs(C)

13! ~ 6×10^9 permutations





outputs(C) ⊆ π(outputs(D))

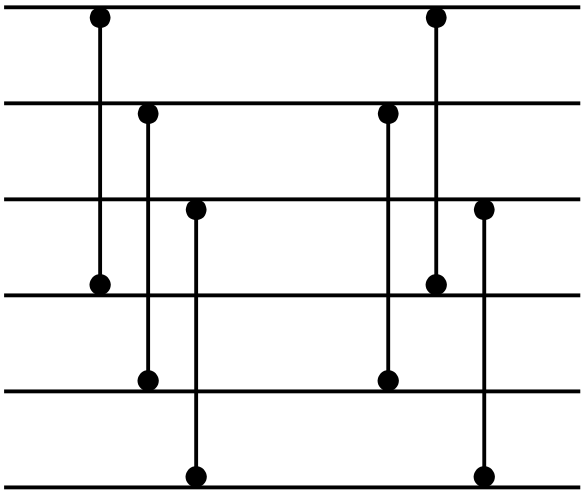
For n=13 568504 candidate layers

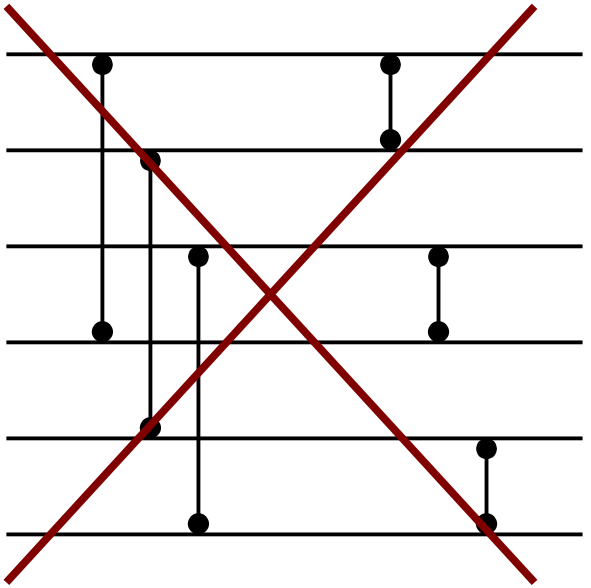
1069 minimal second layers
212 if you listen to Knuth
(118 if you are Knuth)

outputs(C) ⊆ π(outputs(D))

~1000 size of outputs(C)

13! ~ 6×10^9 permutations





outputs(C) ⊆ π(outputs(D))

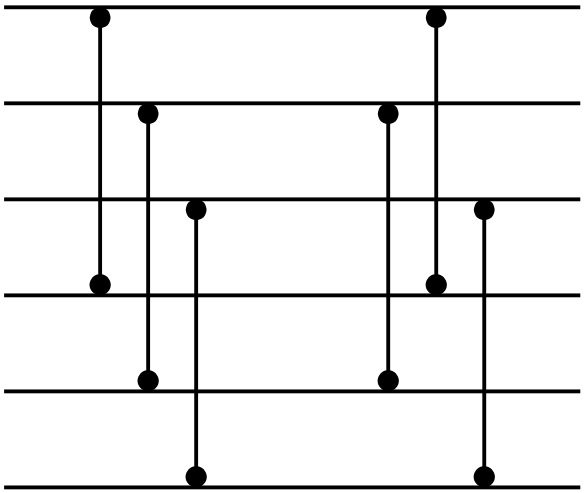
For n=13 568504 candidate layers

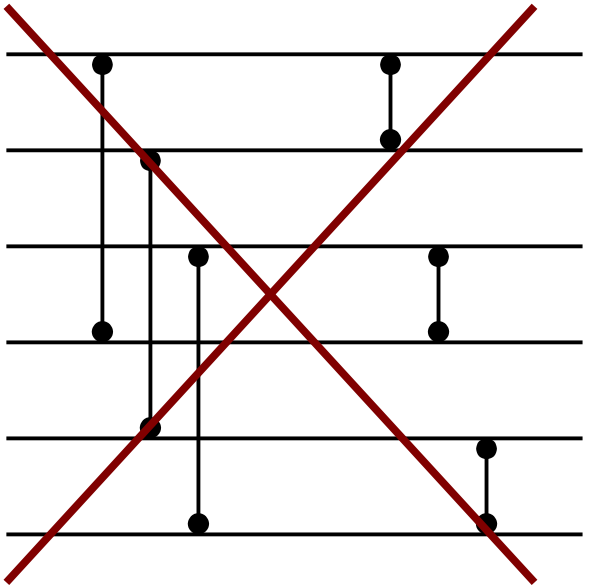
1069 minimal second layers
212 if you listen to Knuth
(118 if you are Knuth)

outputs(C) ⊆ π(outputs(D))

~1000 size of outputs(C)

13! ~ 6×10^9 permutations





outputs(C) ⊆ π(outputs(D))

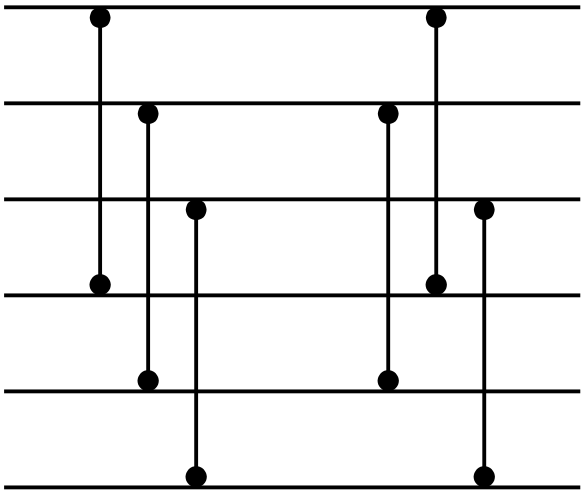
For n=13 568504 candidate layers

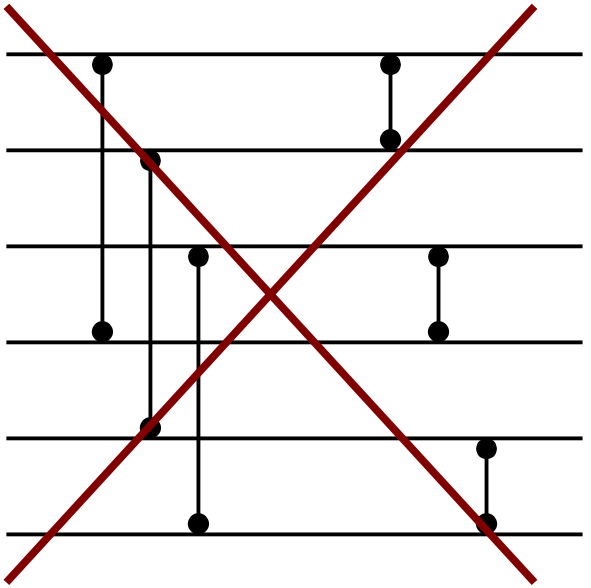
1069 minimal second layers
212 if you listen to Knuth
(118 if you are Knuth)

outputs(C) ⊆ π(outputs(D))

~1000 size of outputs(C)

13! ~ 6×10^9 permutations



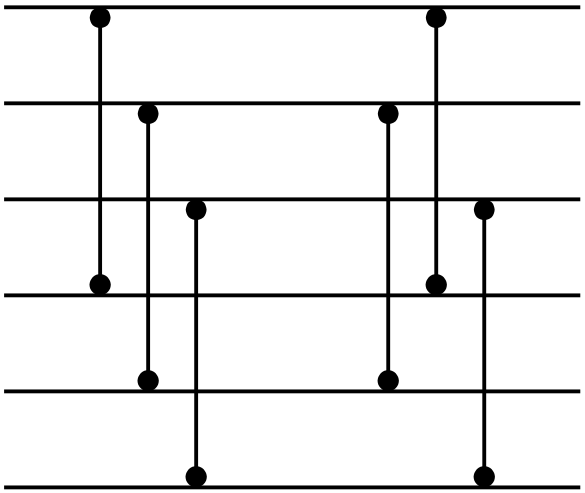


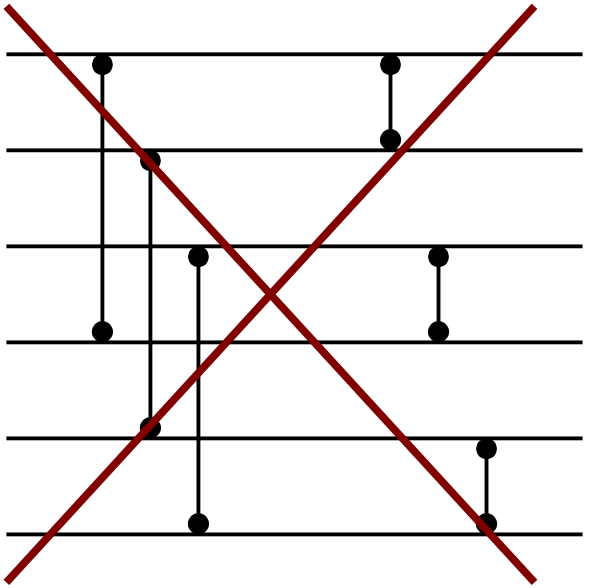
outputs(C) ⊆ π(outputs(D))

For n=13 568504 candidate layers 1069 minimal second layers
212 if you listen to Knuth
(118 if you are Knuth)

Goal: Show that no network of depth 8 sorts 13 inputs

Sufficient to fix the first layer and check 212 second layers





outputs(C) ⊆ π(outputs(D))

Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

Goal: Show that no network of depth 8 sorts 13 inputs

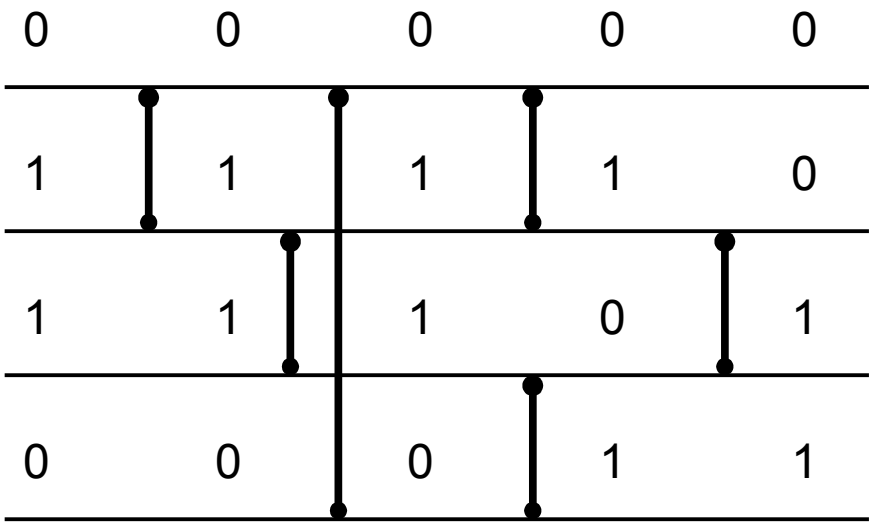
Solution: run 212 instances of SAT (one for each second layer)

Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)

Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

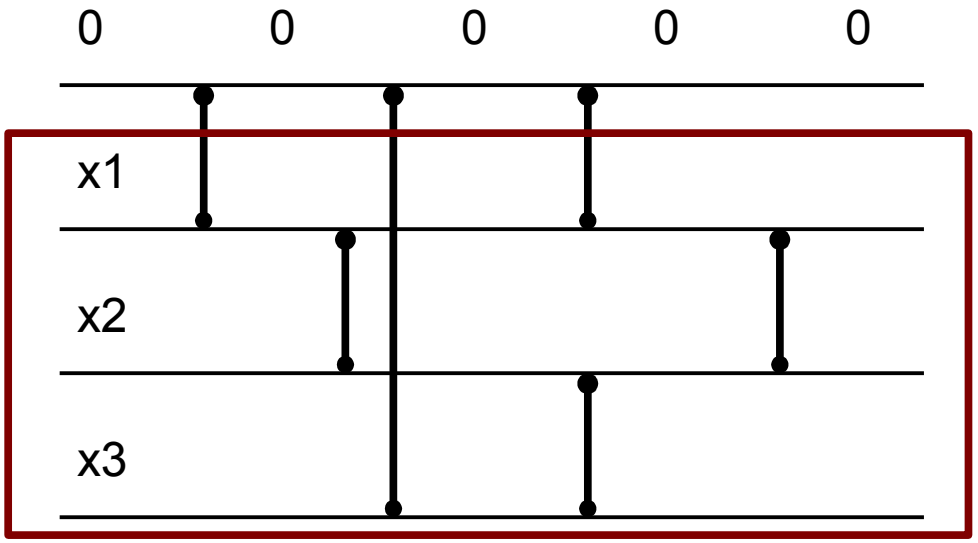
Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)



Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

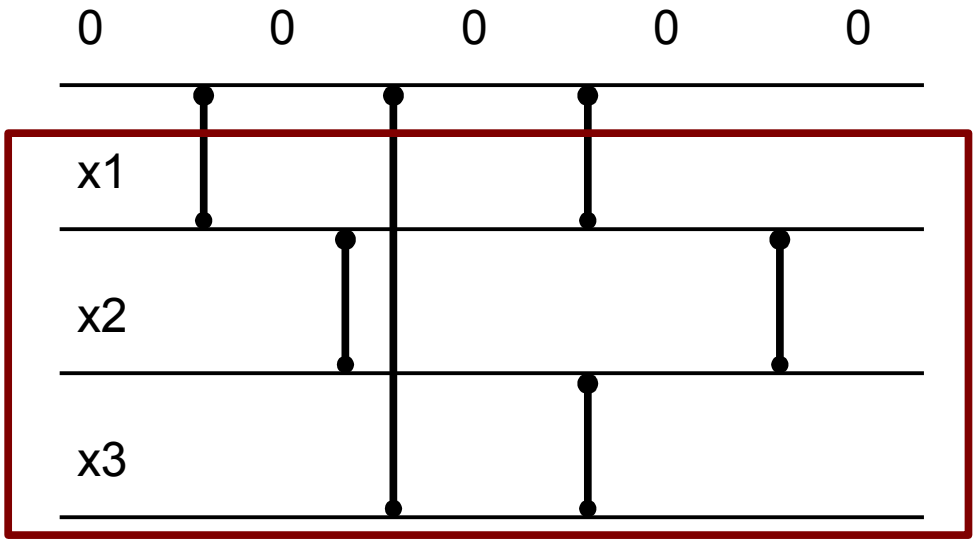
Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)



Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)

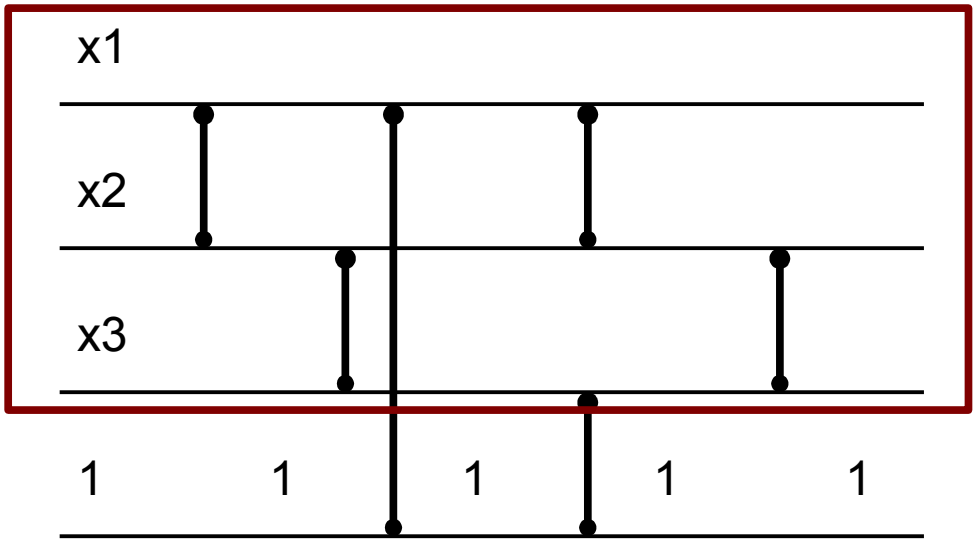


For n=13, try only inputs: 000x

Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)

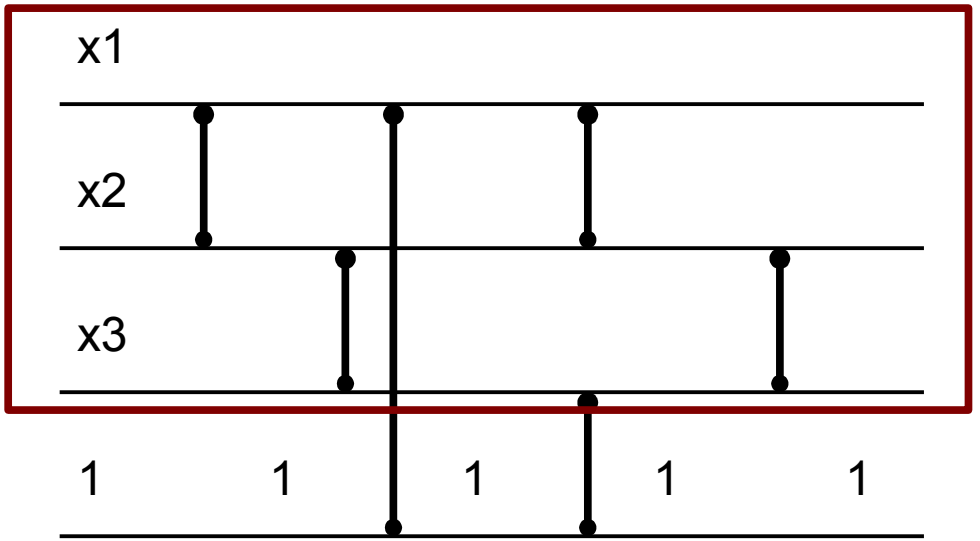


For n=13, try only inputs: 000x, y111, 0z11, 00w1

Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)



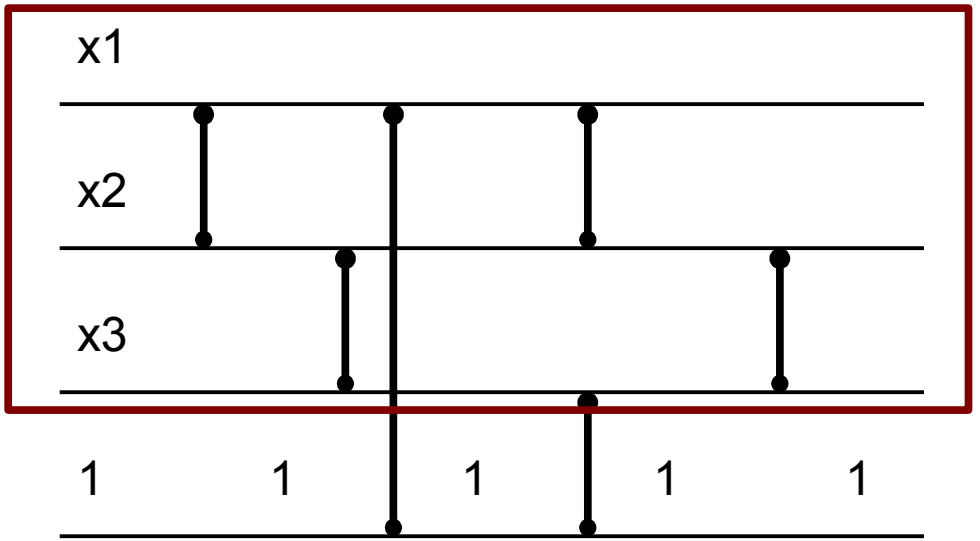
For n=13, try only inputs: 000x, y111, 0z11, 00w1

One instance: ~4min. Overall: 13 hours.

Goal: Show that no network of depth 8 sorts 13 inputs

Solution: run 212 instances of SAT (one for each second layer)

Network sorts every input:
sorts(00000000000) and sorts(00000000001) and sorts(00000000010) and ... and sorts(11111111111)

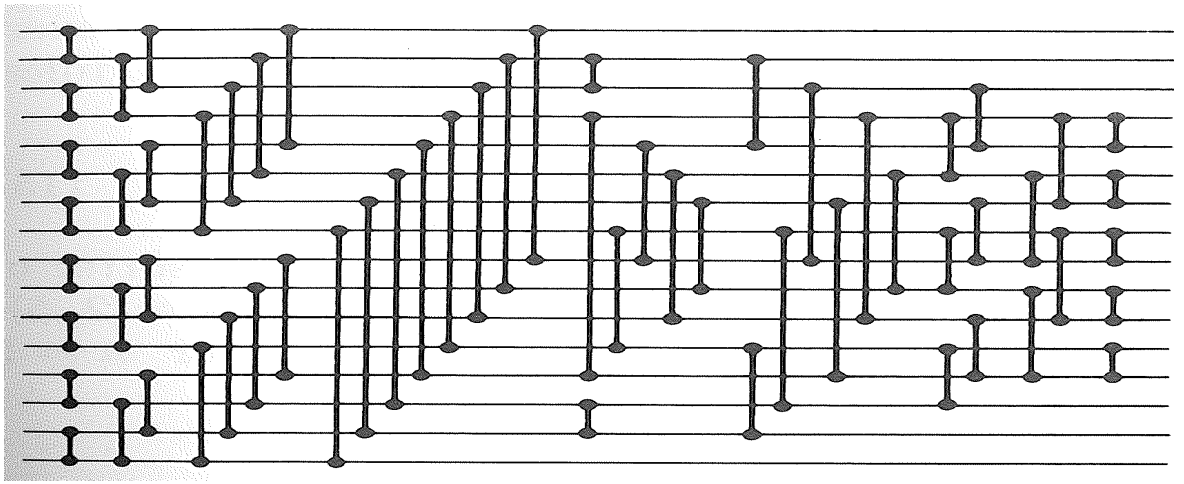


For n=13, try only inputs: 000x, y111, 0z11, 00w1

One instance: ~4min. Overall: 13 hours.

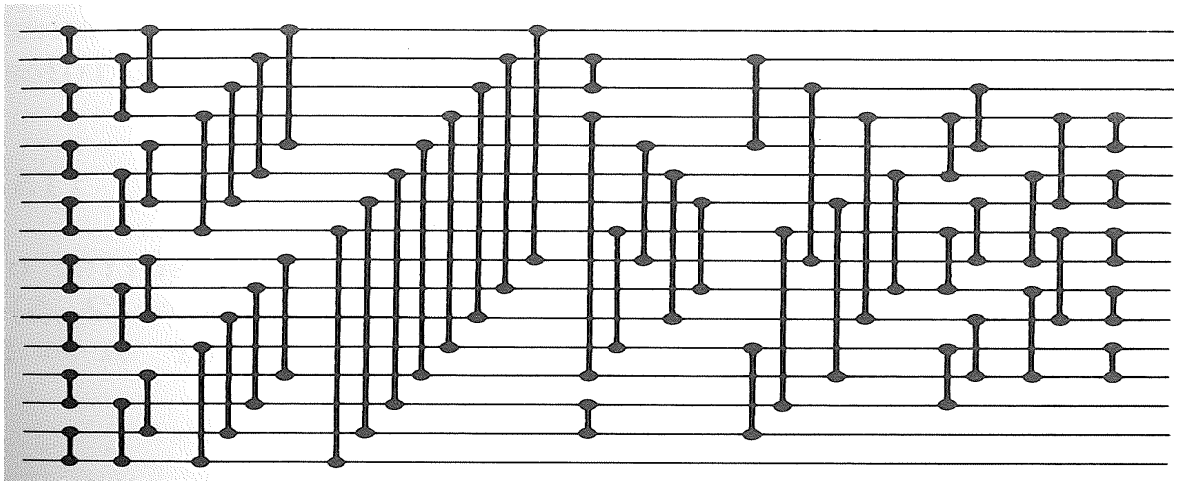
Optimal depth for 13 inputs is 9

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**



n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**

We reconstructed all previous results in 15 seconds



n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper Bound	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower Bound	0	1	3	3	5	5	6	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**

We reconstructed all previous results in 15 seconds