Лекции 3-4

Динамическое программирование

- **Динамическое программирование** в математике и теории вычислительных систем — метод решения задач с оптимальной подструктурой и перекрывающимися подзадачами, который намного эффективнее, чем решение «в лоб» (brute force).

Идея динамического программирования

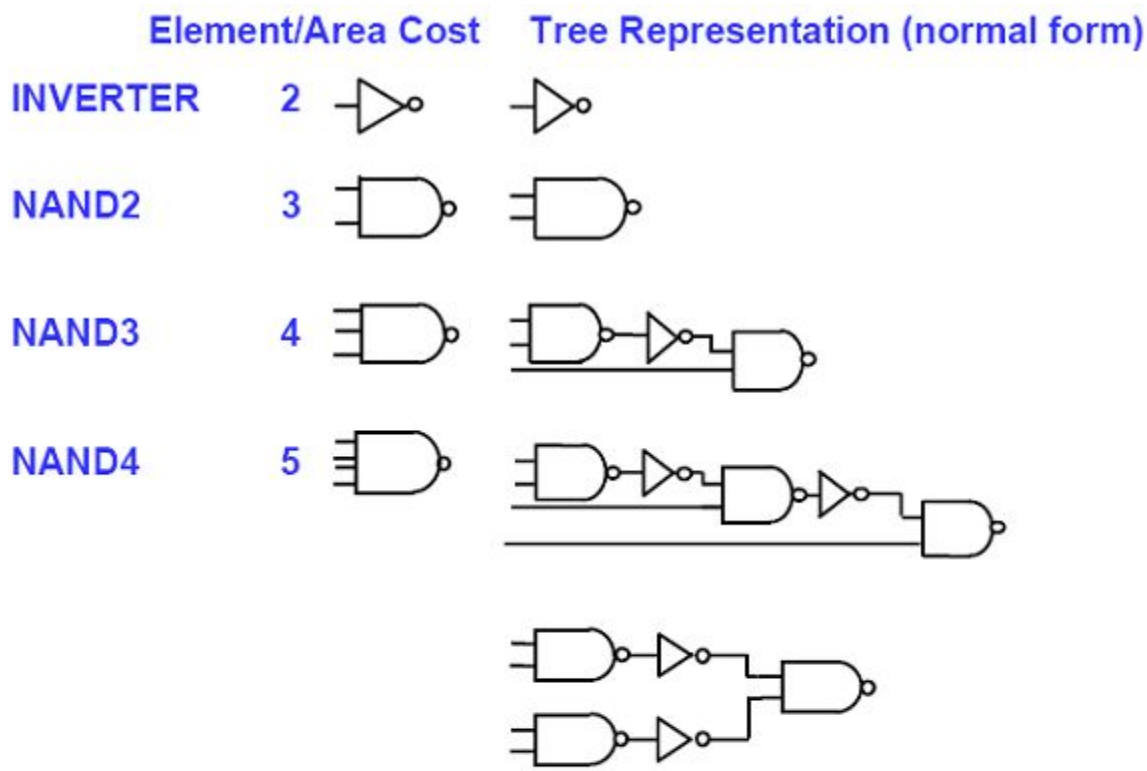
Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

- Разбиение задачи на подзадачи меньшего размера.
- Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
- Использование полученного решения подзадач для конструирования решения исходной задачи.

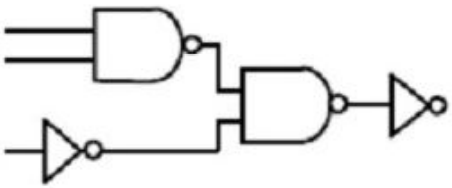
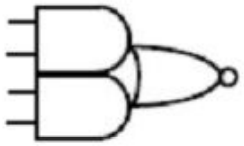
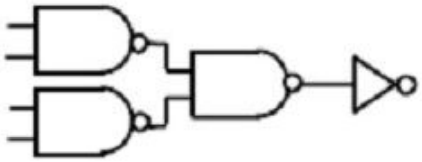
Подзадачи решаются делением их на подзадачи ещё меньшего размера и т. д., пока не приходят к тривиальному случаю задачи, решаемой за константное время

Принцип оптимальности (Беллман): подстратегия оптимальной стратегии должна быть оптимальной

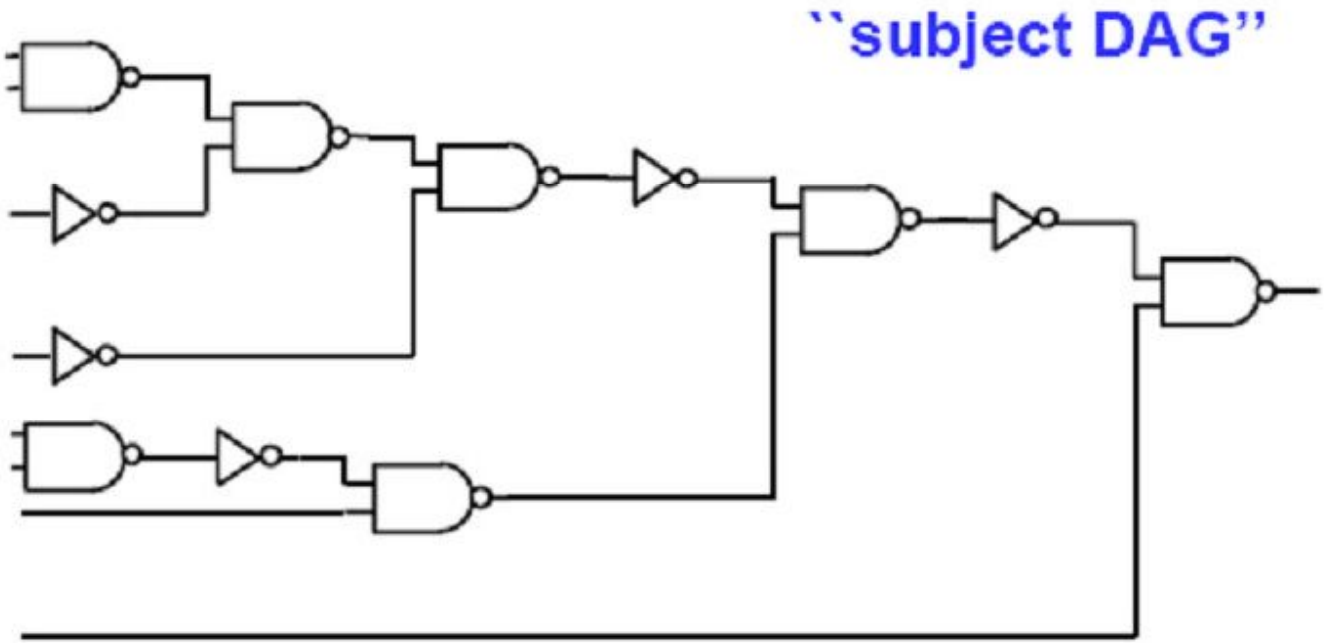
Пример: технологический мэппинг. Примеры библиотечных элементов-1.



Примеры библиотечных элементов-2.

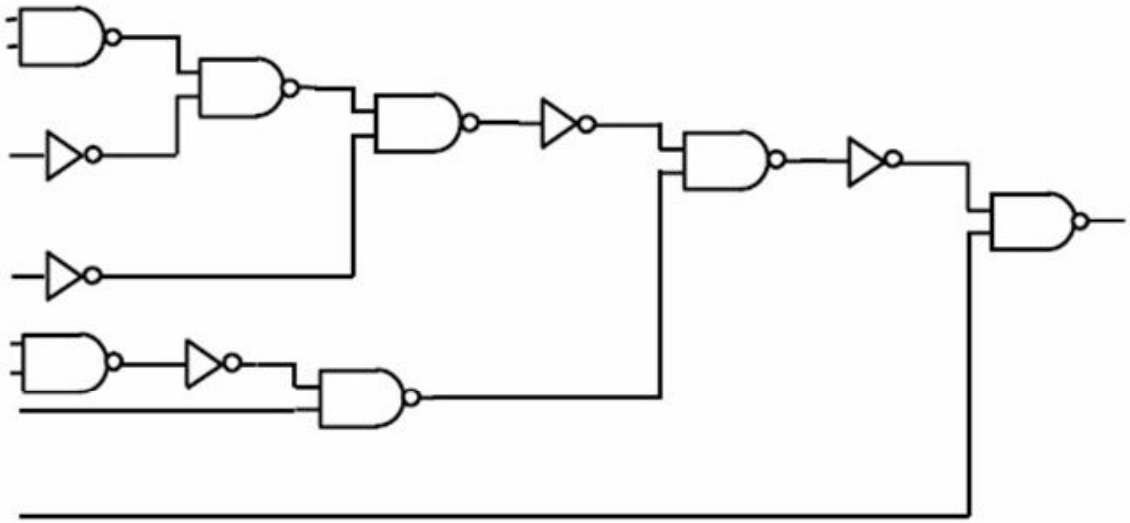
	Element/Area Cost		Tree Representation (normal form)
AOI21	4		
AOI22	5		

Исходная схема

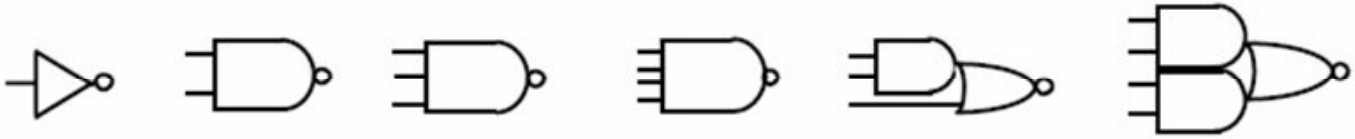


Формулировка проблемы

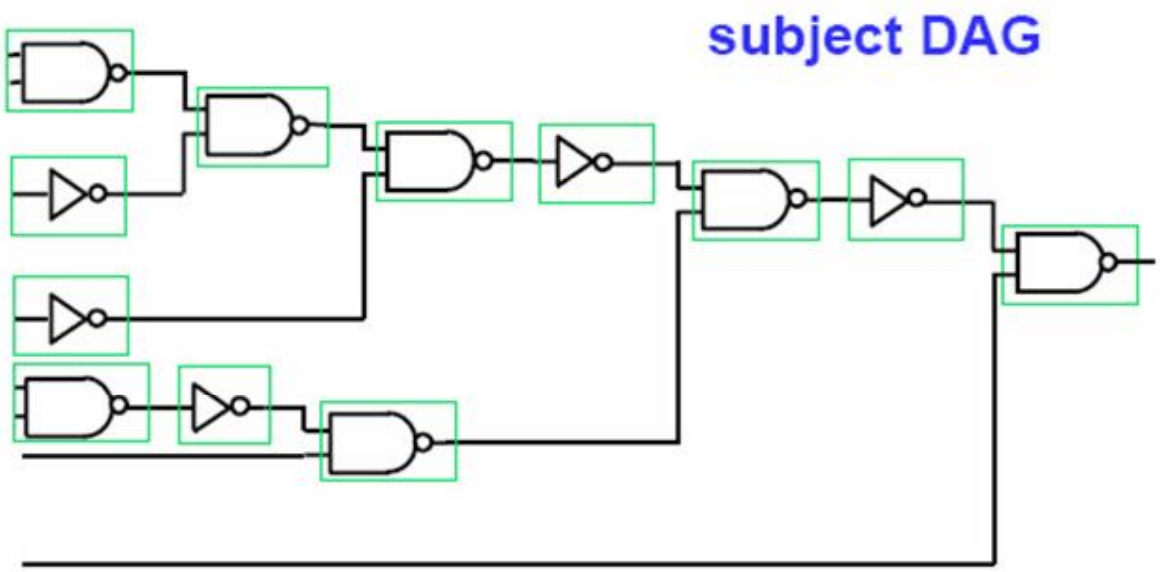
Find an ``optimal'' (in area, delay, power) mapping of this circuit (DAG)



into this library

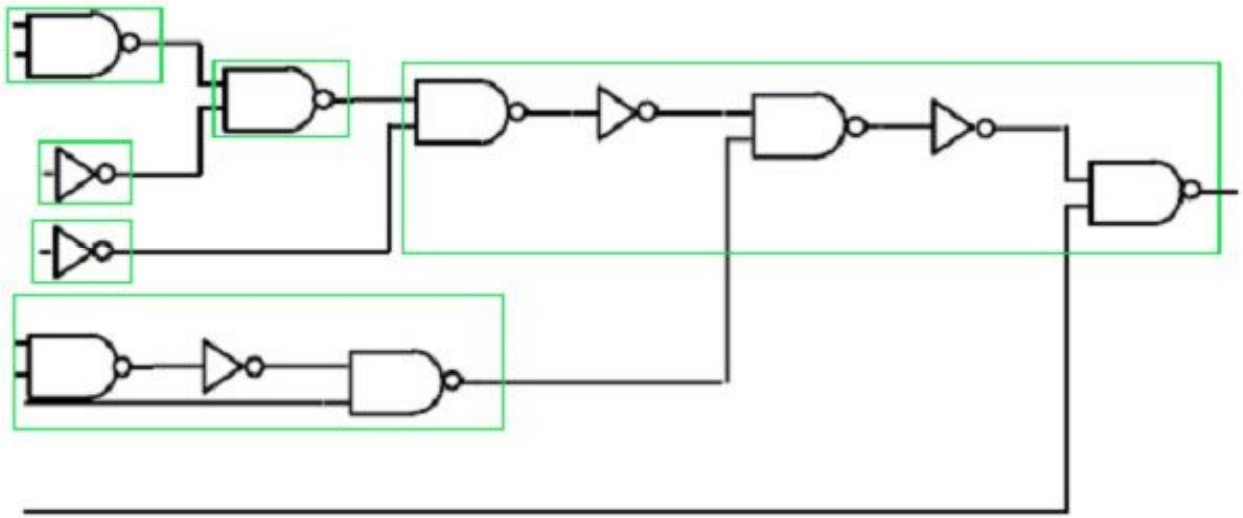


Тривиальное решение



7	NAND2 (3) =	21
5	INV (2) =	10
	Area cost	31

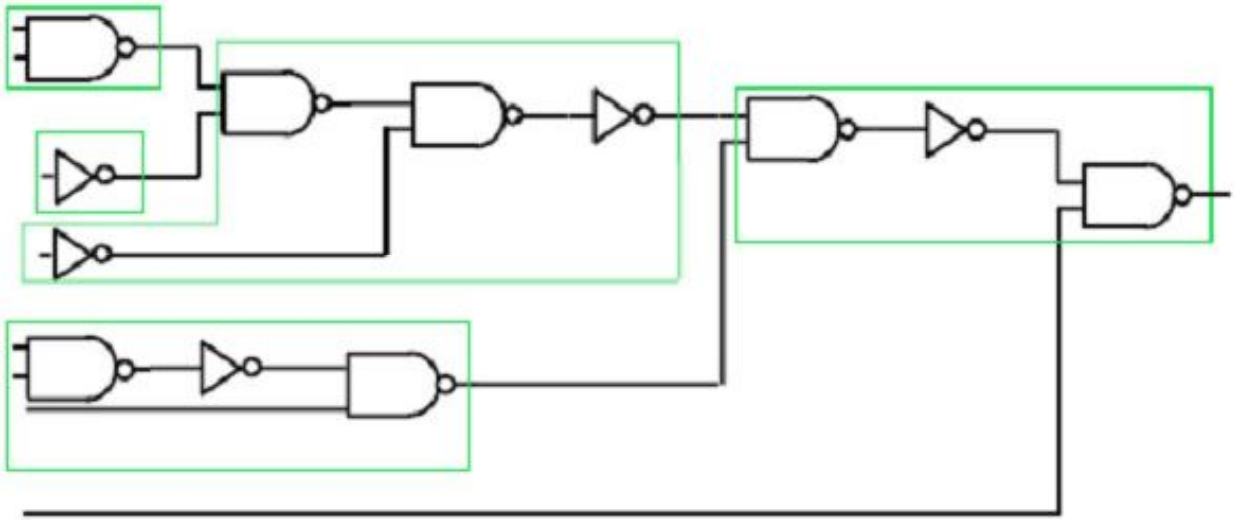
Покрытие-1



2 INV	= 4
2 NAND2	= 6
1 NAND3	= 4
1 NAND4	= 5

Area cost 19

Покрытие-2



1 INV	=	2
1 NAND2	=	3
2 NAND3	=	8
1 AOI21	=	4
	Area Cost	17

Формулировка проблемы: DAG covering

Represent input netlist in normal form
⇒ subject DAG

Represent each library gate with normal forms for the logic function
⇒ primitive DAGs

Each primitive DAG has a cost

Goal: Find a minimum cost covering of the subject DAG by the primitive DAGs

Normal form: 2-input NAND gates and inverters

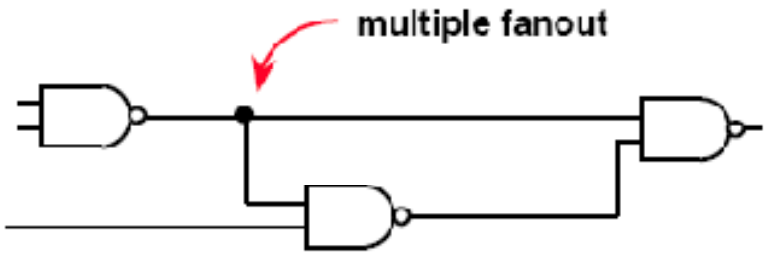
K. Keutzer, *DAGON: Technology Binding and Local Optimization by DAG Matching*, in Proceedings of the 24th Design Automation Conference, 1987 and 25 Years of Design Automation

DAG Covering

Sound Algorithmic approach

NP-hard optimization problem

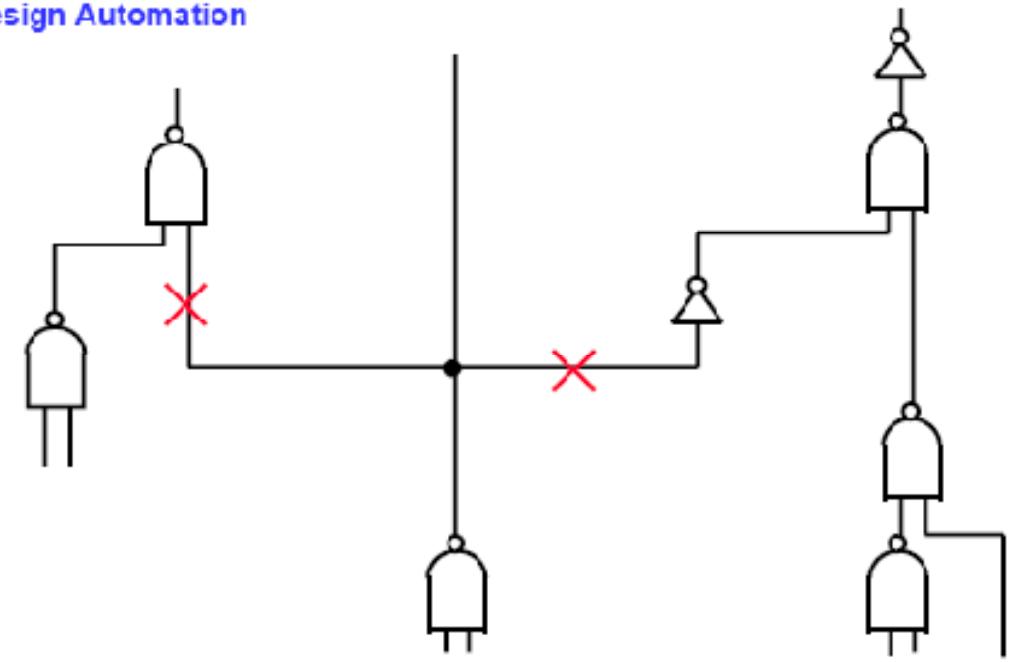
K. Keutzer, D. Richards, *Computation Complexity of Logic Synthesis and Optimization*, in Proceedings of the International Workshop on Logic Synthesis, 1989



Tree covering heuristic: If subject and primitive DAGs are trees, efficient algorithm can find optimum cover ⇒ dynamic programming formulation

Solution formulation

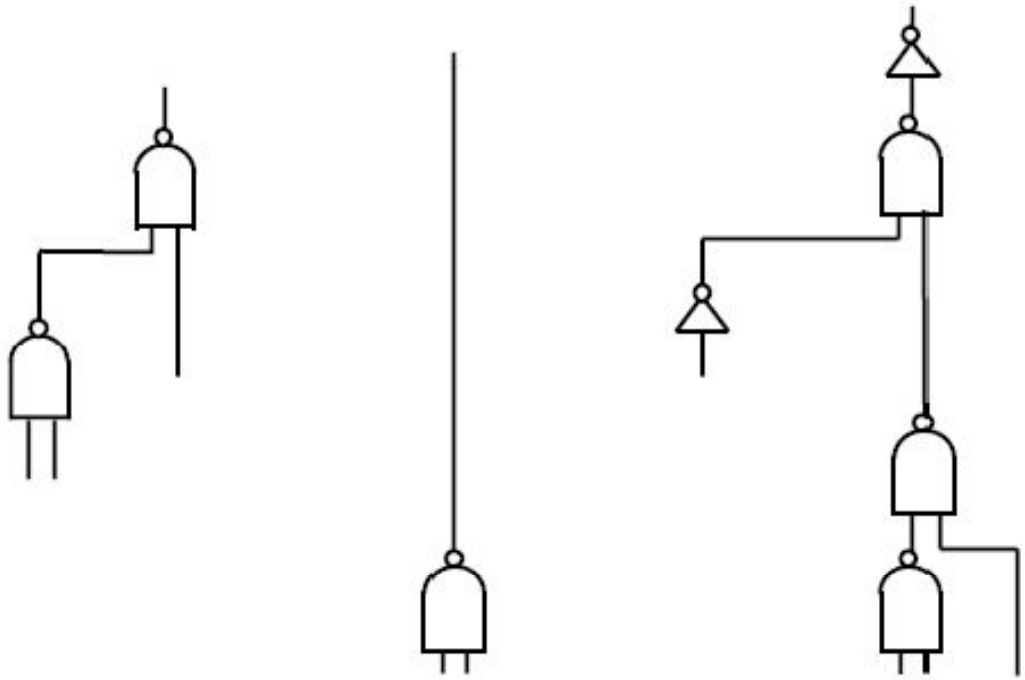
K. Keutzer, *DAGON: Technology Binding and Local Optimization by DAG Matching*, in Proceedings of the 24th Design Automation Conference, 1987 and 25 Years of Design Automation



1) Partition input netlist into forest of trees
2) Solve each tree *optimally* using tree covering
3) Stitch trees back together

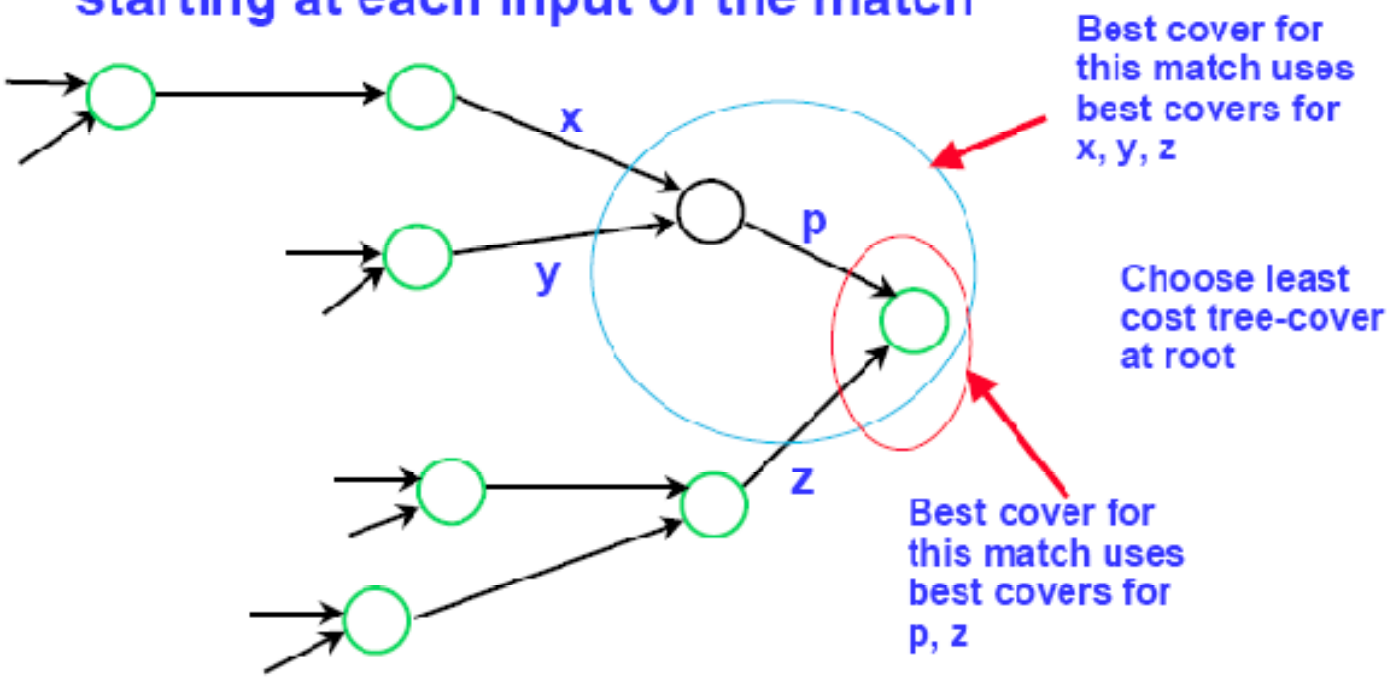
Resulting Trees

Break at multiple fanout points

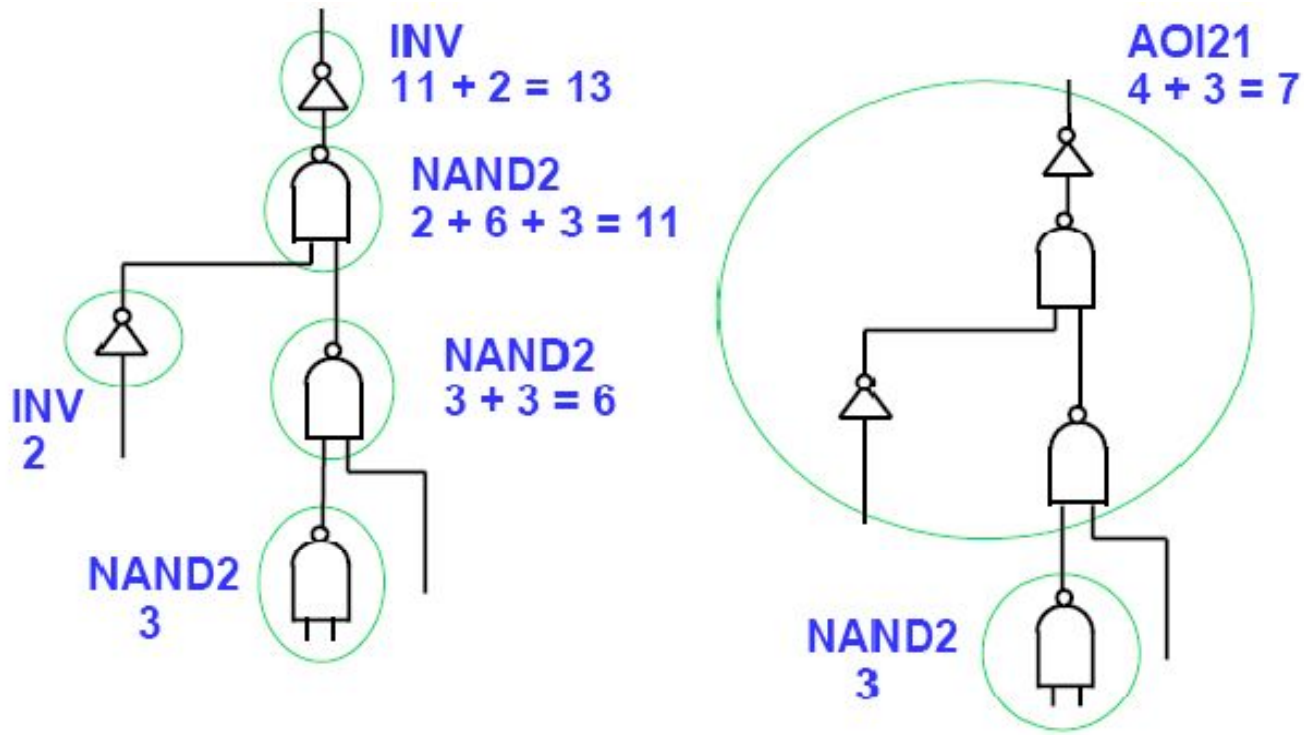


For each tree - Dynamic Programming

Principle of optimality: Optimal cover for a tree consists of a best match at the root of the tree plus the optimal cover for the sub-trees starting at each input of the match



Example of Optimal Tree Covering



DAG covering in detail

1) partition DAG into a forest of trees

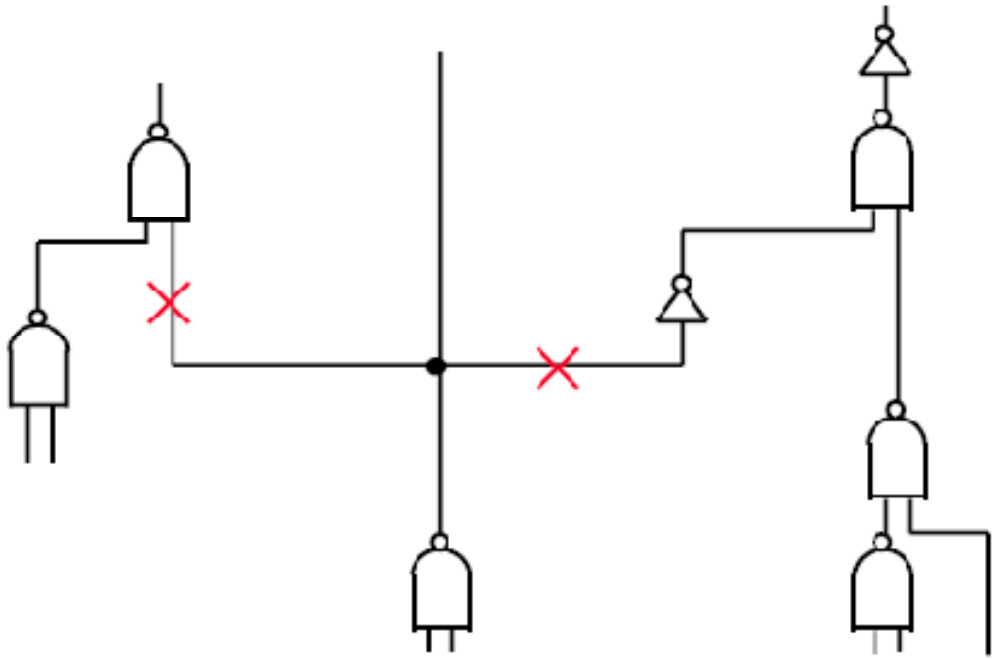
2) normalize each tree

3) optimally cover each tree

   a) generate all candidate matches

   b) find the optimal match using dynamic programming

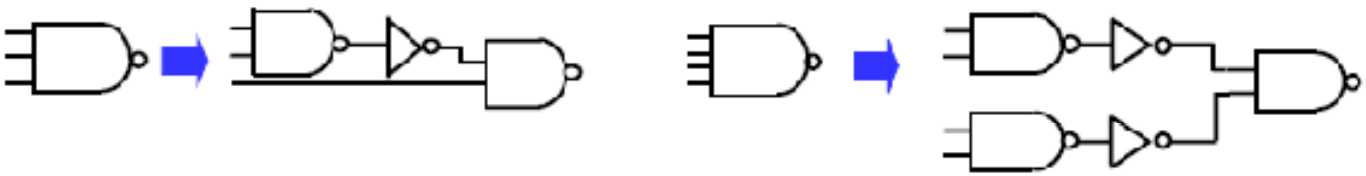
Partition DAG into Forest of trees



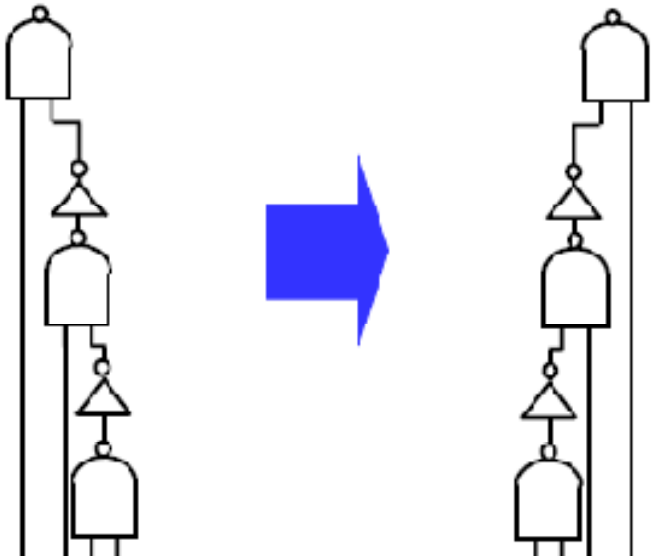
Each gate with fanout >1 becomes root of a new tree

Normalize each tree

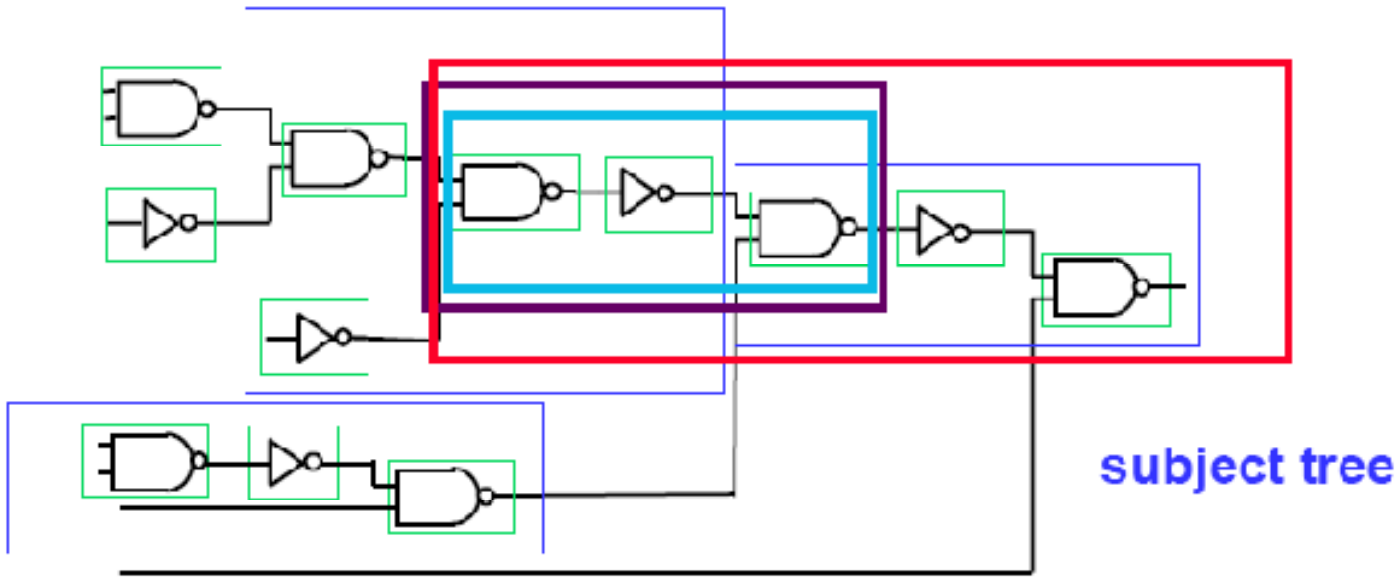
Re-express netlist into 2-input Nand gates and Inverters



Make each tree left-oriented

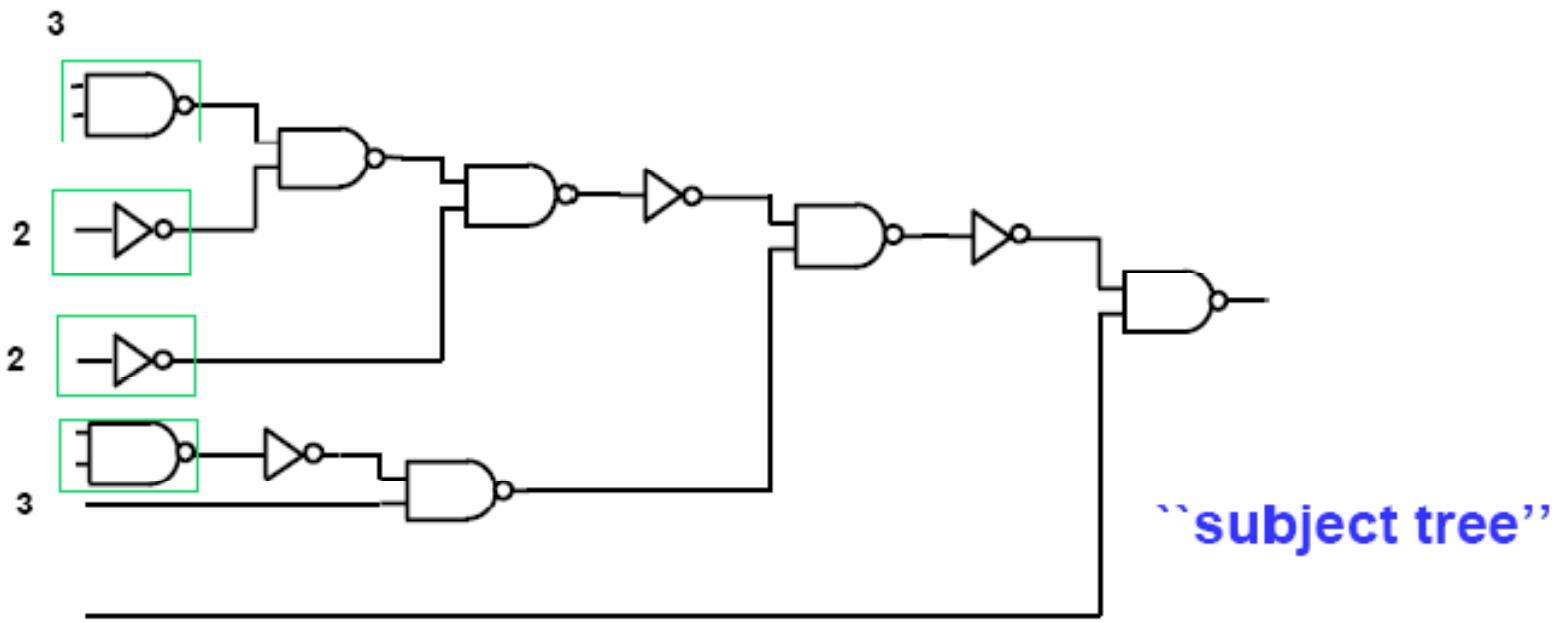


Generate candidate matches - 1

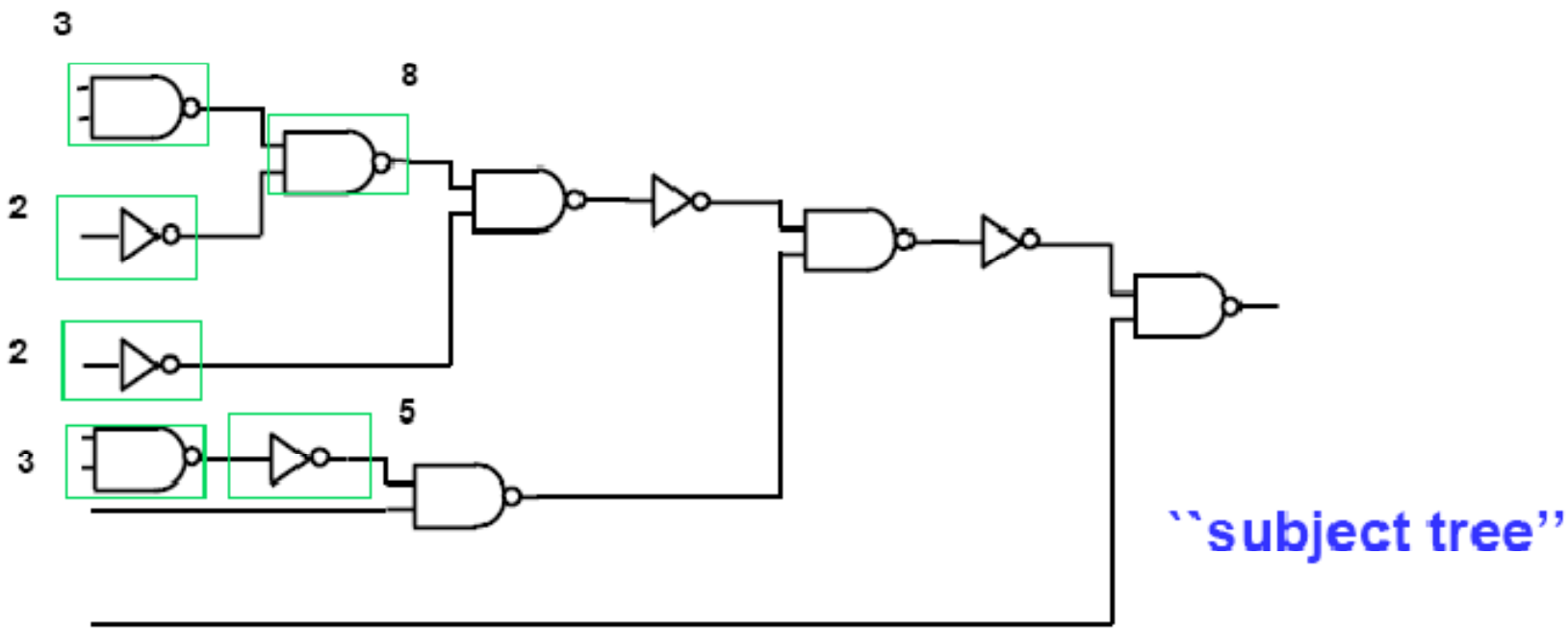


At the end of this segment each gate in the subject tree is annotated with every possible library cell that could be rooted at that gate

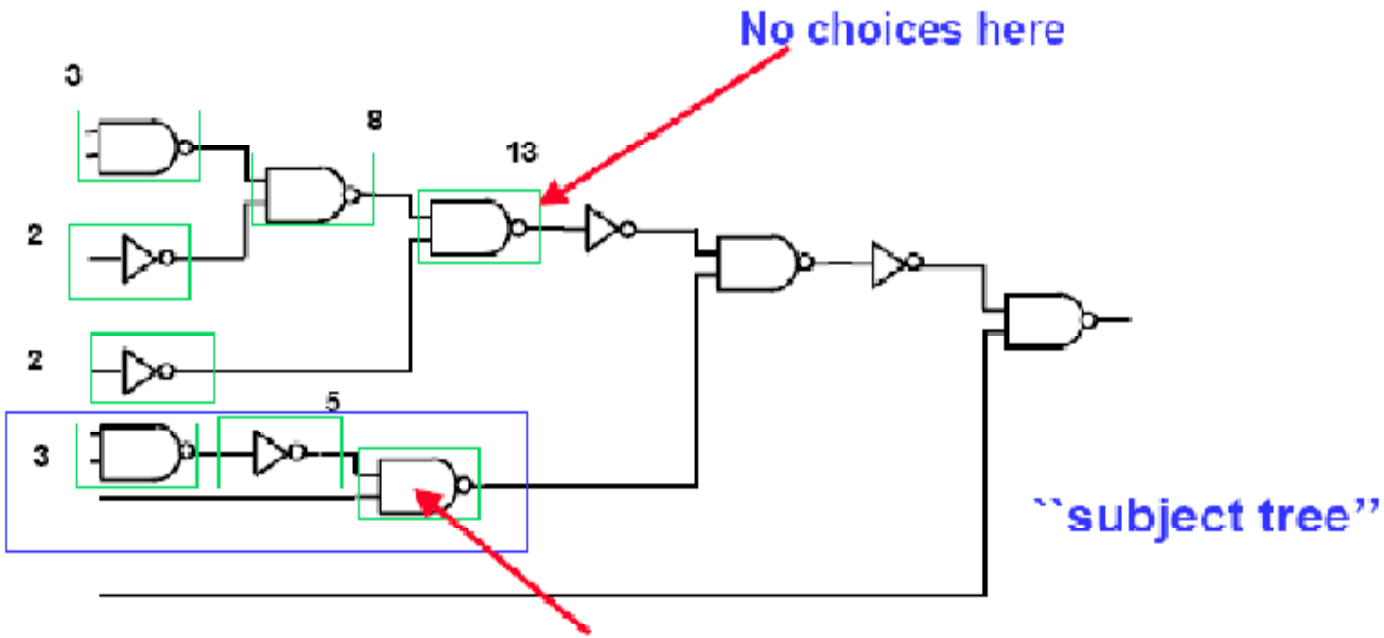
Optimal tree covering - 1



Optimal tree covering - 2



Optimal tree covering - 3

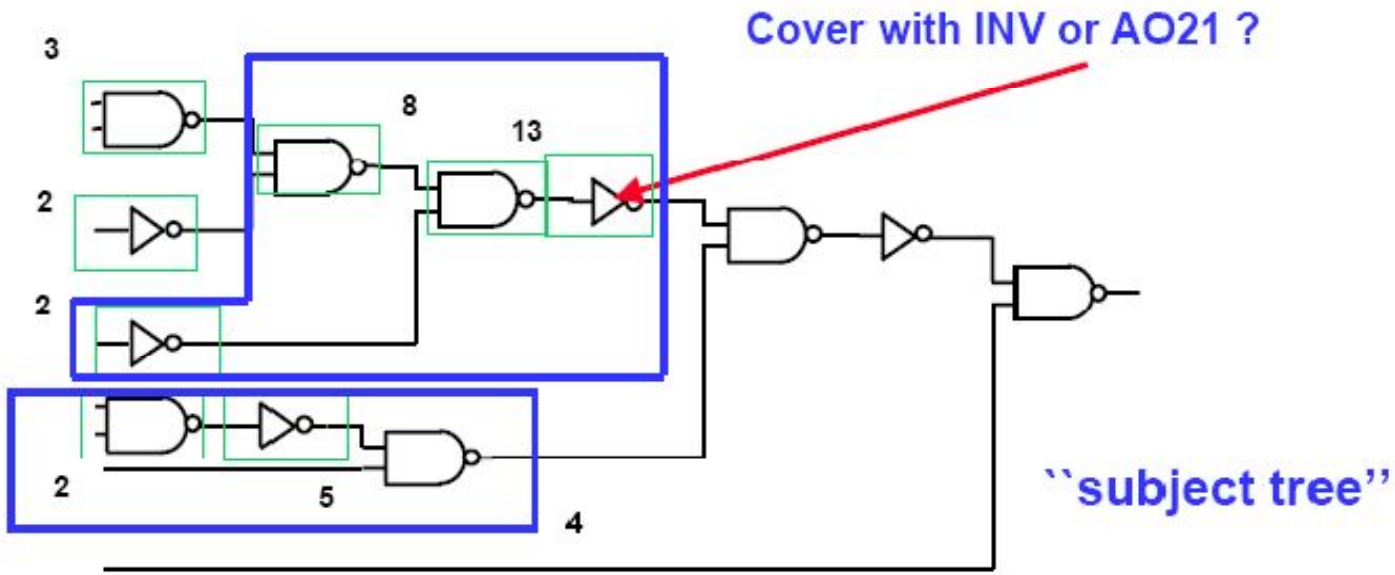


Cover with ND2 or ND3 ?

1 NAND2	3	1 NAND3	= 4
+ subtree	5		

Area cost 8

Optimal tree covering - 4



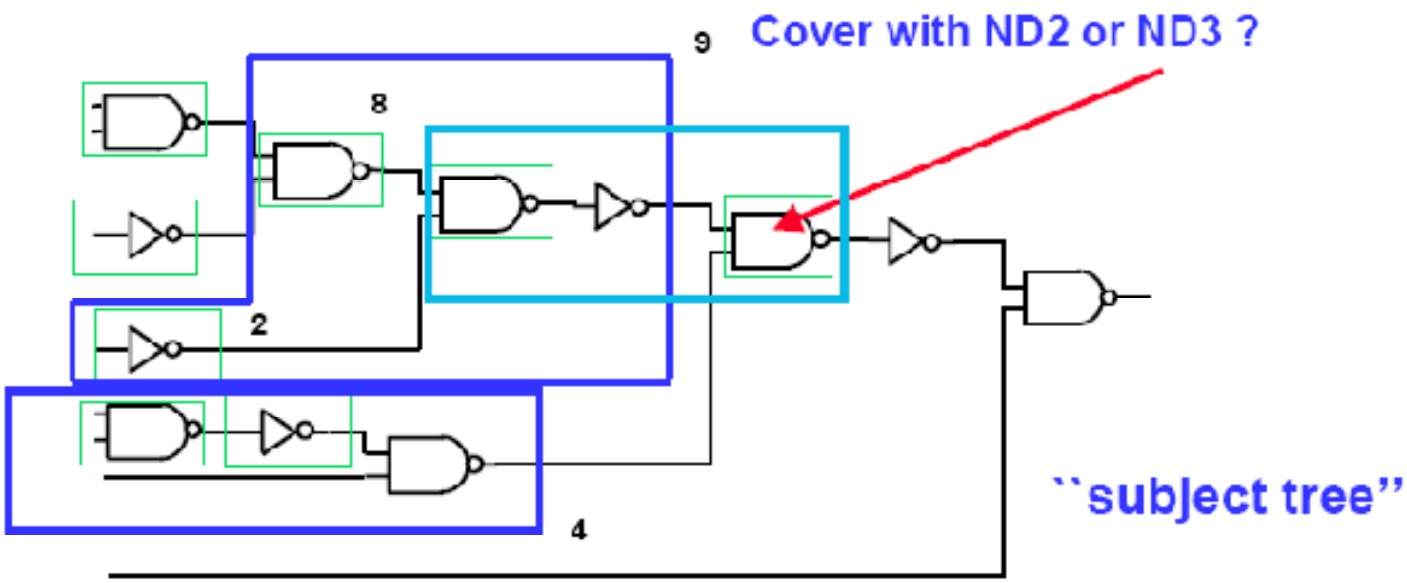
1 AO21	4
+ subtree 1	3
+ subtree 2	2

Area cost 9

1 Inverter	2
+ subtree	13

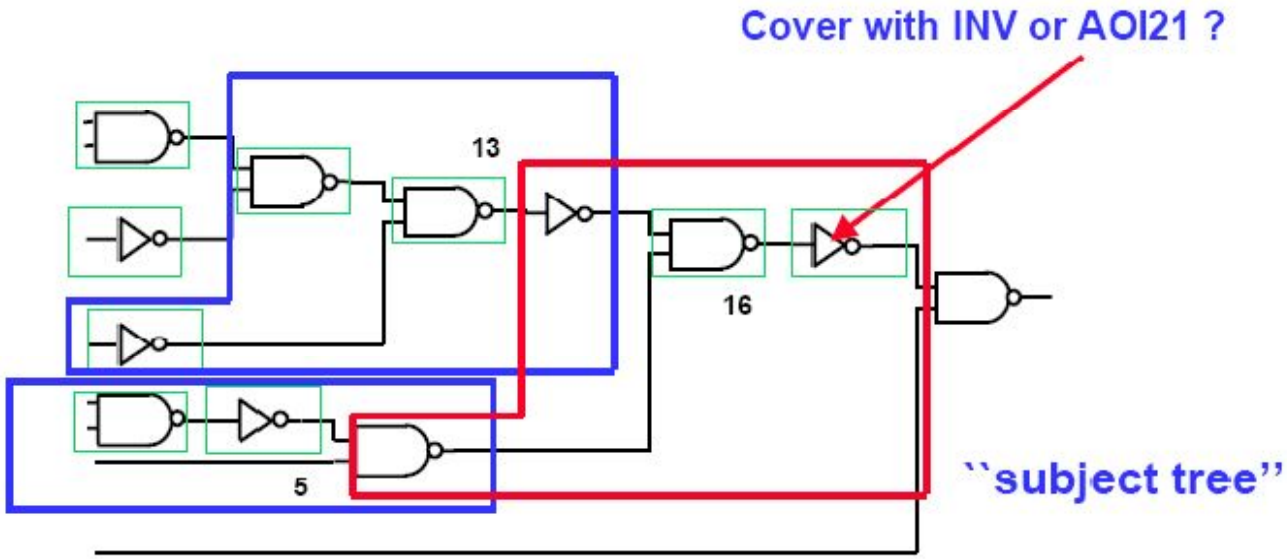
Area cost 15

Optimal tree covering - 5



NAND2			NAND3		
	subtree 1	9		subtree 1	8
	subtree 2	4		subtree 2	2
	1 NAND2	3		subtree 3	4
				1 NAND3	4
	Area cost 16			Area cost 18	

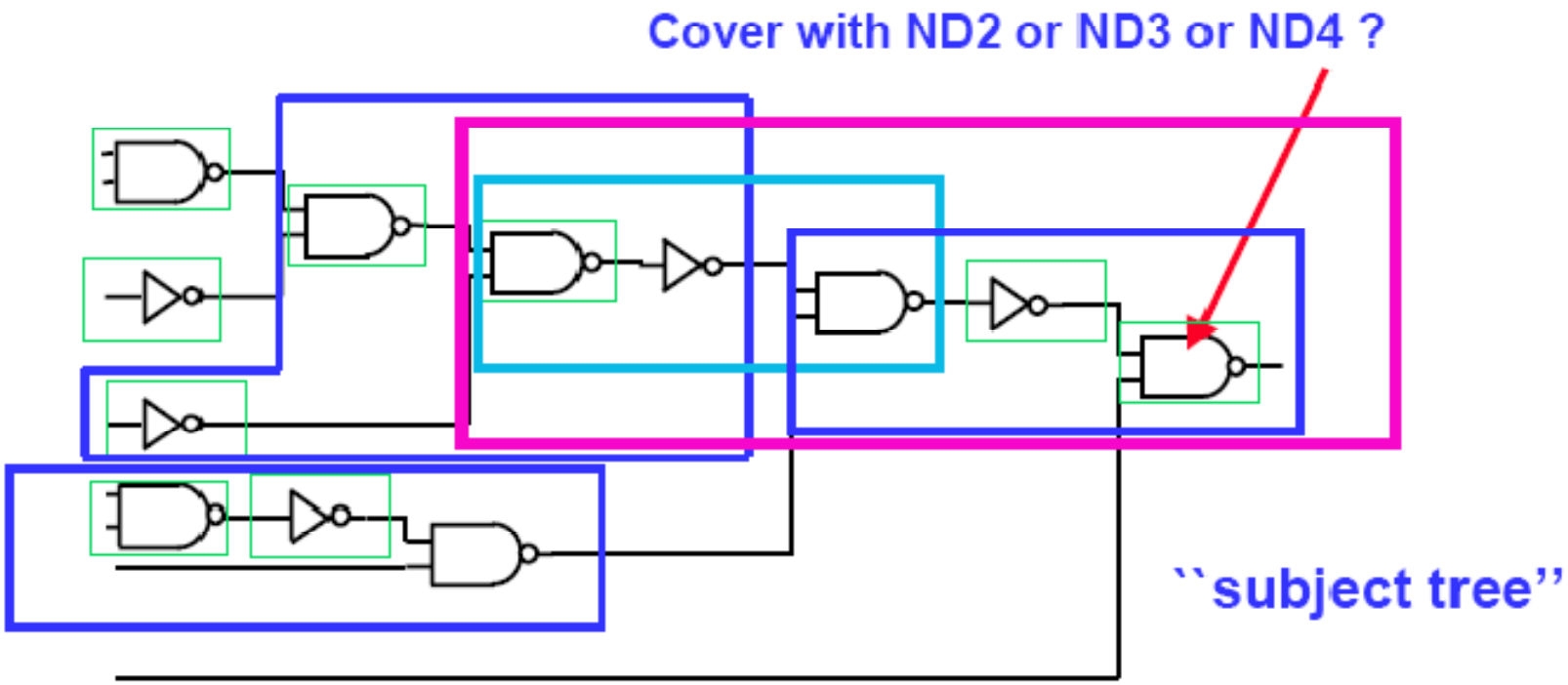
Optimal tree covering - 6



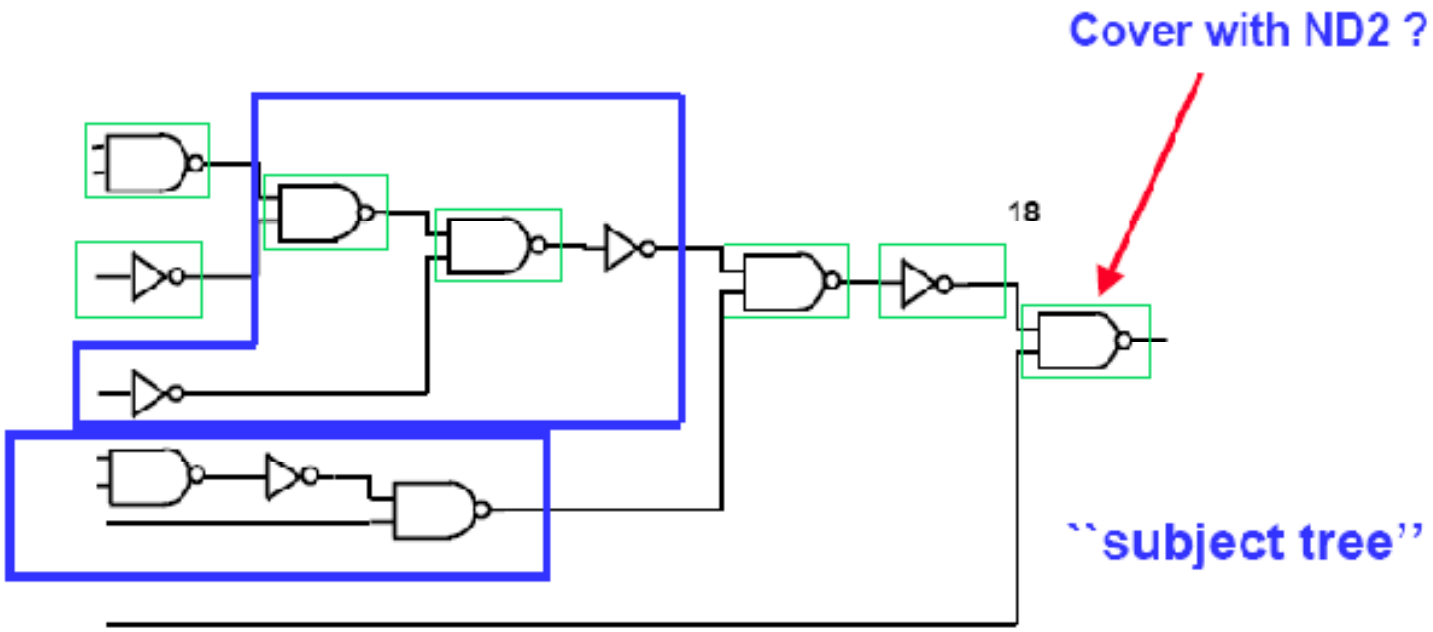
INV	subtree 1	16
	1 INV	2
	Area cost 18	

AOI21	subtree 1	13
	subtree 2	5
	1 AOI21	4
	Area cost 22	

Optimal tree covering - 7



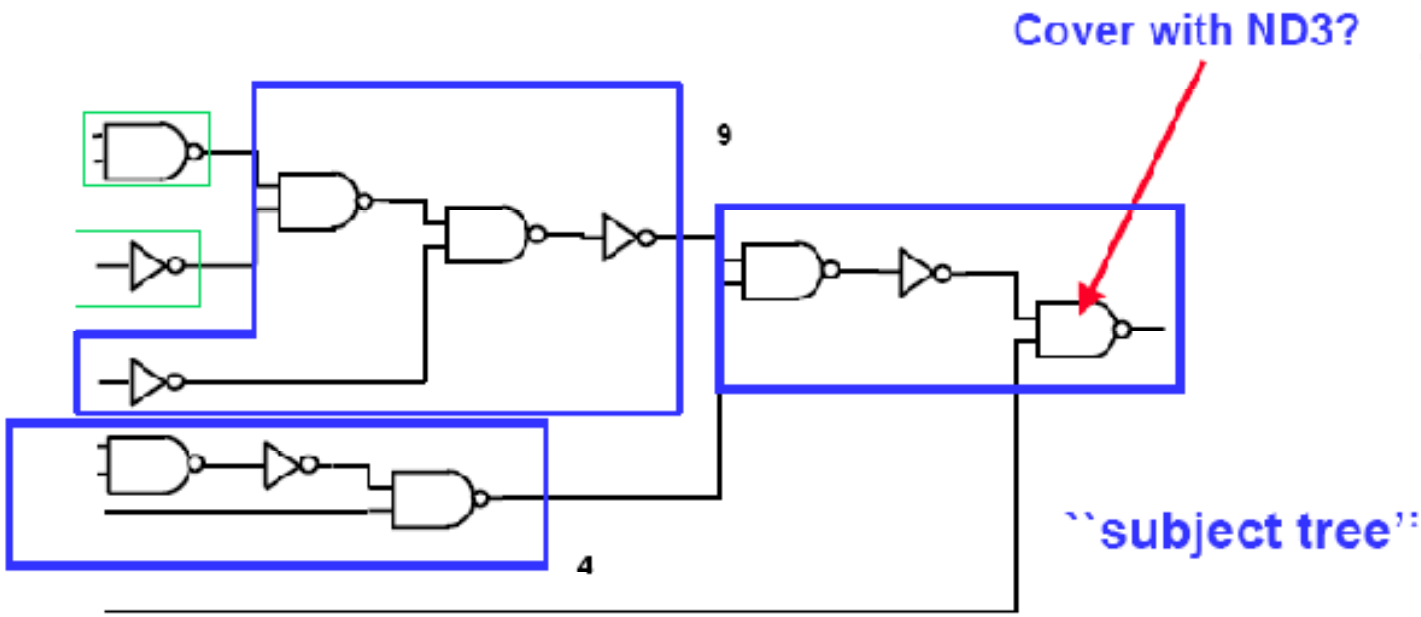
Cover 1 - NAND2



subtree 1	18
subtree 2	0
1 NAND2	3

Area cost 21

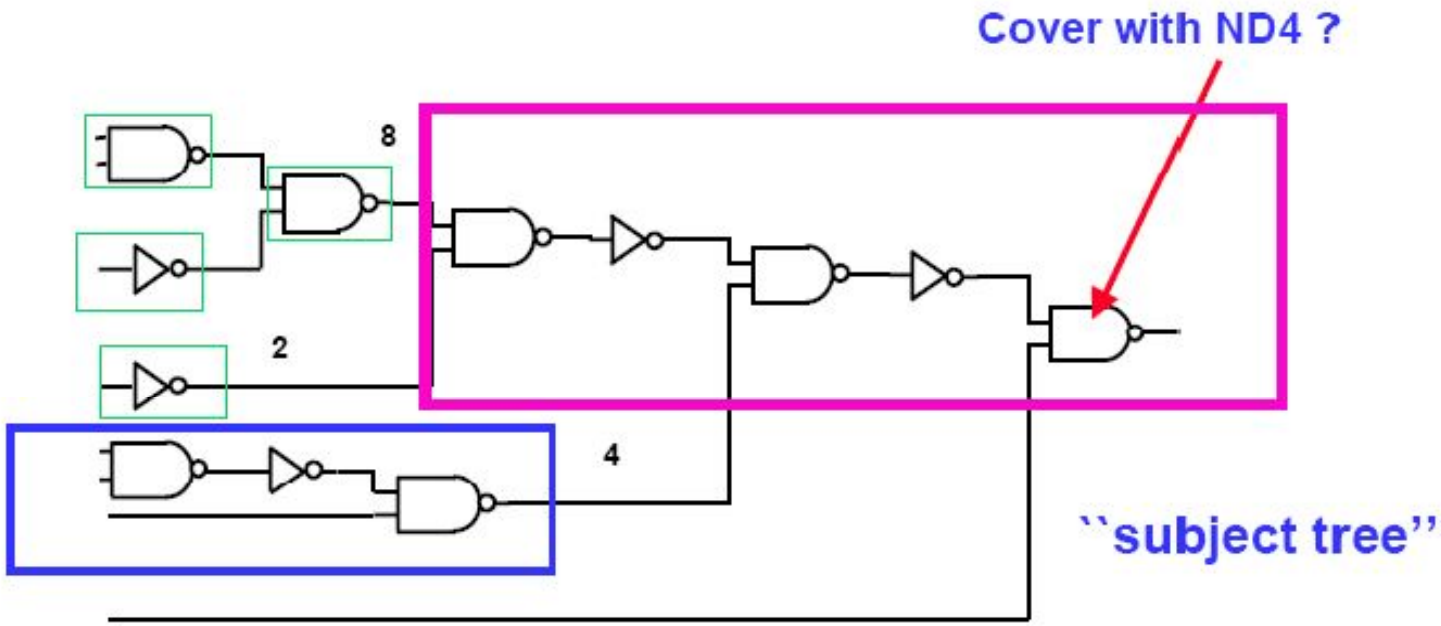
Cover 2 - NAND3



subtree 1	9
subtree 2	4
subtree 3	0
1 NAND3	4

Area cost 17

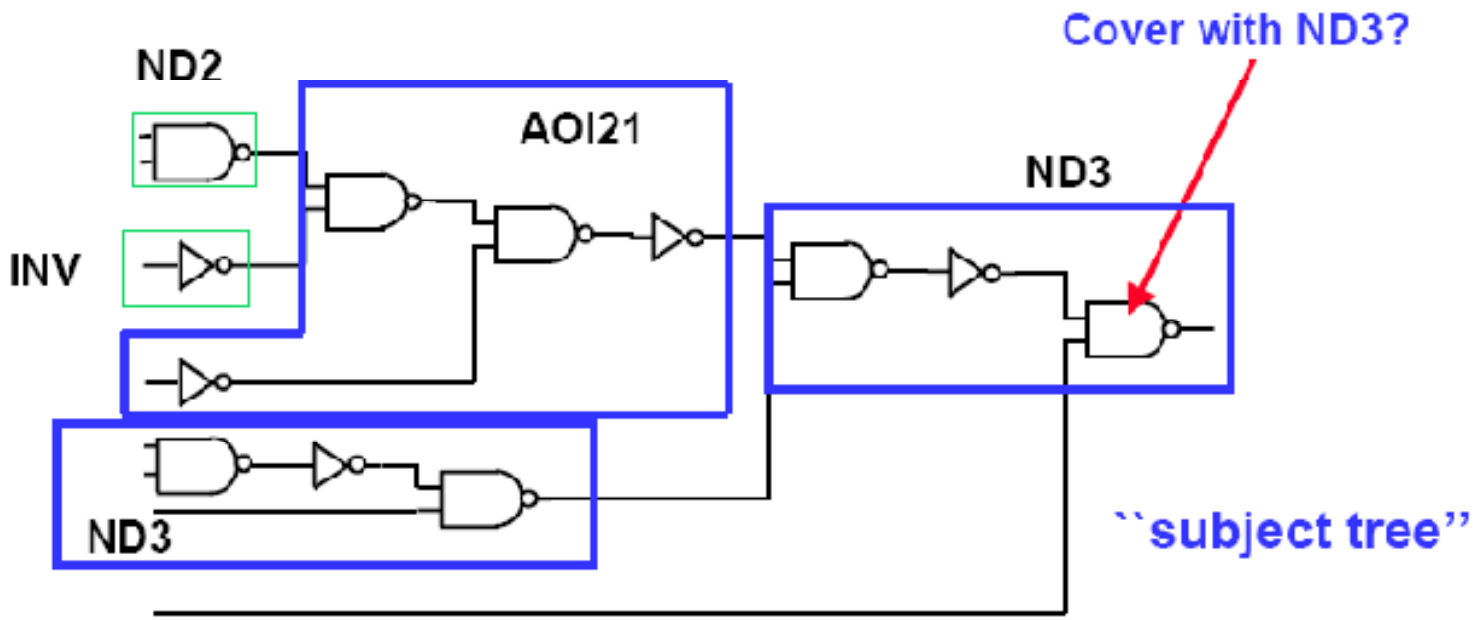
Cover - 3



subtree 1	8
subtree 2	2
subtree 3	4
subtree 4	0
1 NAND4	5

Area cost 19

Optimal Cover



What's the complexity?

INV	2
ND2	3
2 ND3	8
AOI21	4

Area cost 17