Verifying multipliers with *BMDs and a backward construction algorithm

Pierre Senellart

May 3rd 2002

Introduction

- BDDs (Bryant, 1986) very powerful tools for veifying arithmetic circuits.
- But exponential on multipliers.
- *BMDs (Bryant, 1994) give a polynomial algorithm but need high-level information.
- Backward construction algorithm (Hamaguchi et al, 1995)

Moment decomposition of a function

$$f : \{0,1\}^n \longrightarrow \mathbb{N}$$

$$\begin{array}{rlcl} f_{\bar{x}_i}: & \{0,1\}^{n-1} & \rightarrow & \mathbb{N} \\ & (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) & \mapsto & f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) \end{array}$$

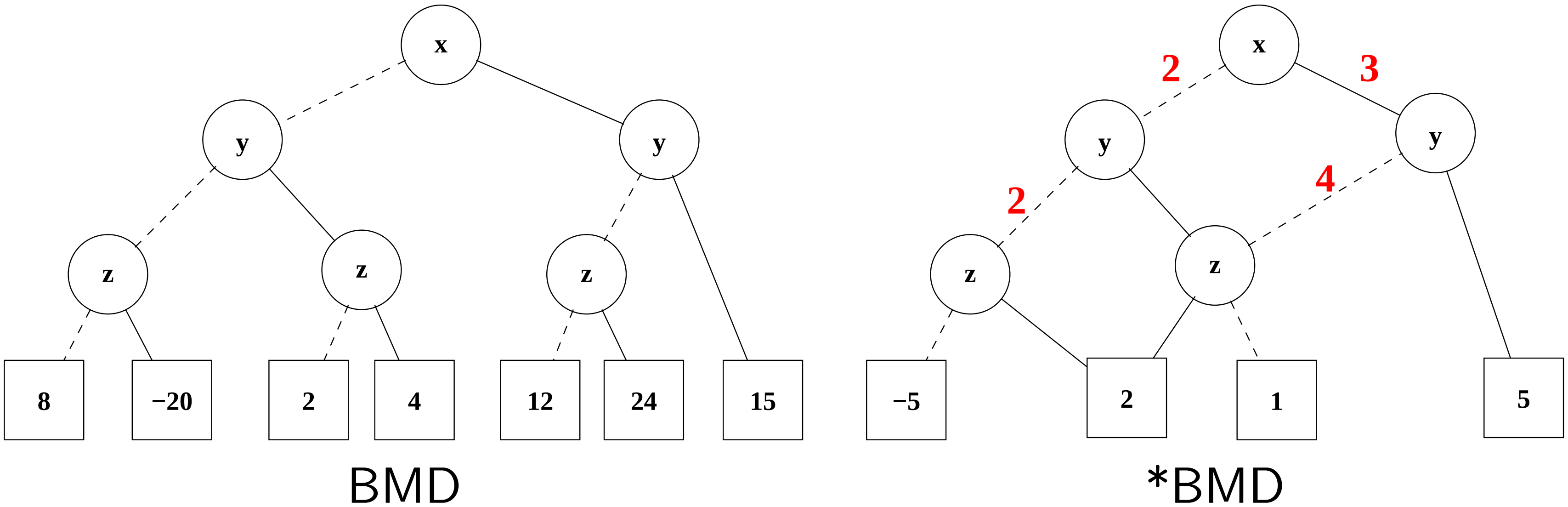
$$\begin{array}{rlcl} f_{x_i}: & \{0,1\}^{n-1} & \rightarrow & \mathbb{N} \\ & (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) & \mapsto & f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \end{array}$$

Moment decomposition of a function (continuing...)

$$f_{\dot{x}_i} = f_{x_i} - f_{\bar{x}_i}$$

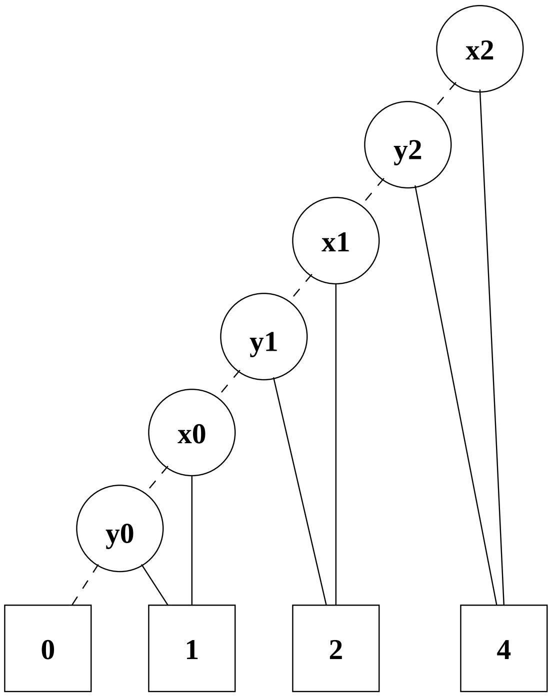
$$f(x_1, \ldots, x_n) = \underbrace{f_{\bar{x}_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}_{\text{constant moment}} + x_i \underbrace{f_{\dot{x}_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}_{\text{linear moment}}$$

BMDs and *BMDs

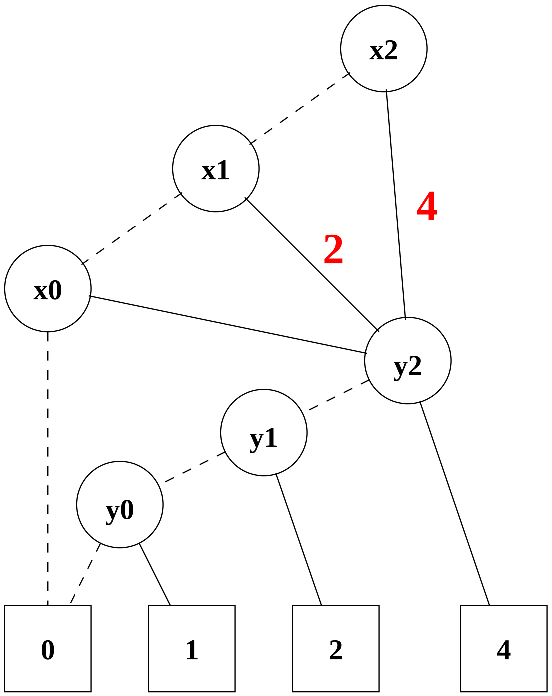


$$f(x,y,z) = 8 - 20z + 2y + 4yz + 12x + 24xz + 15xy$$

Arithmetic operations



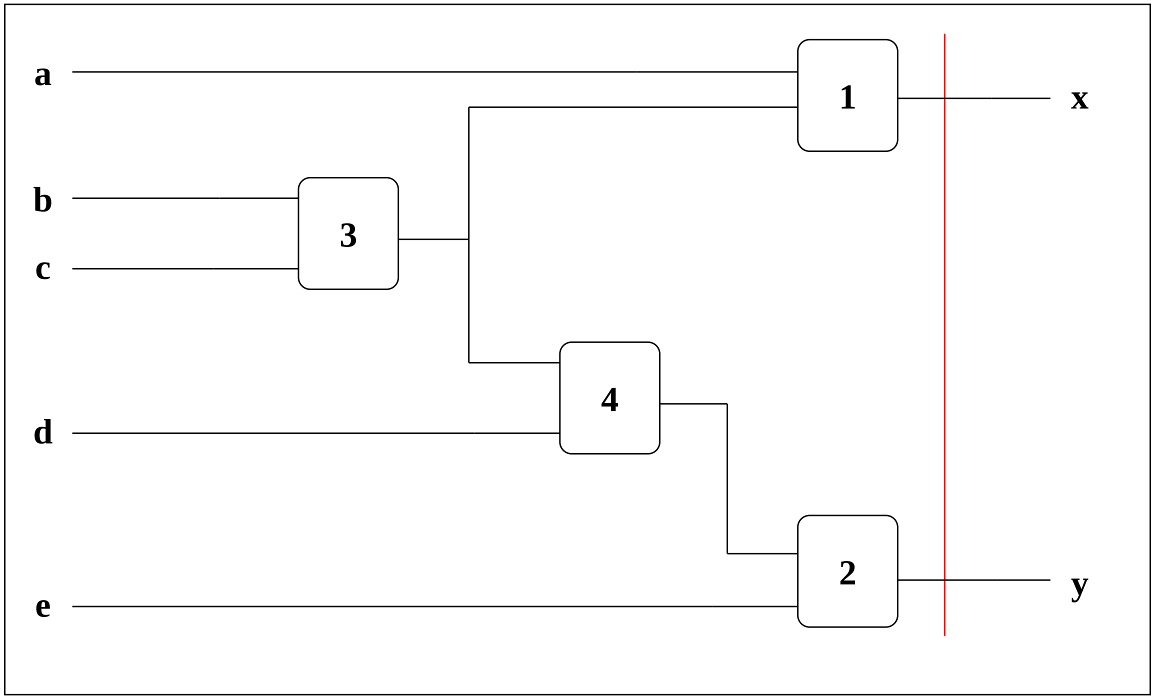
Addition



Multiplication

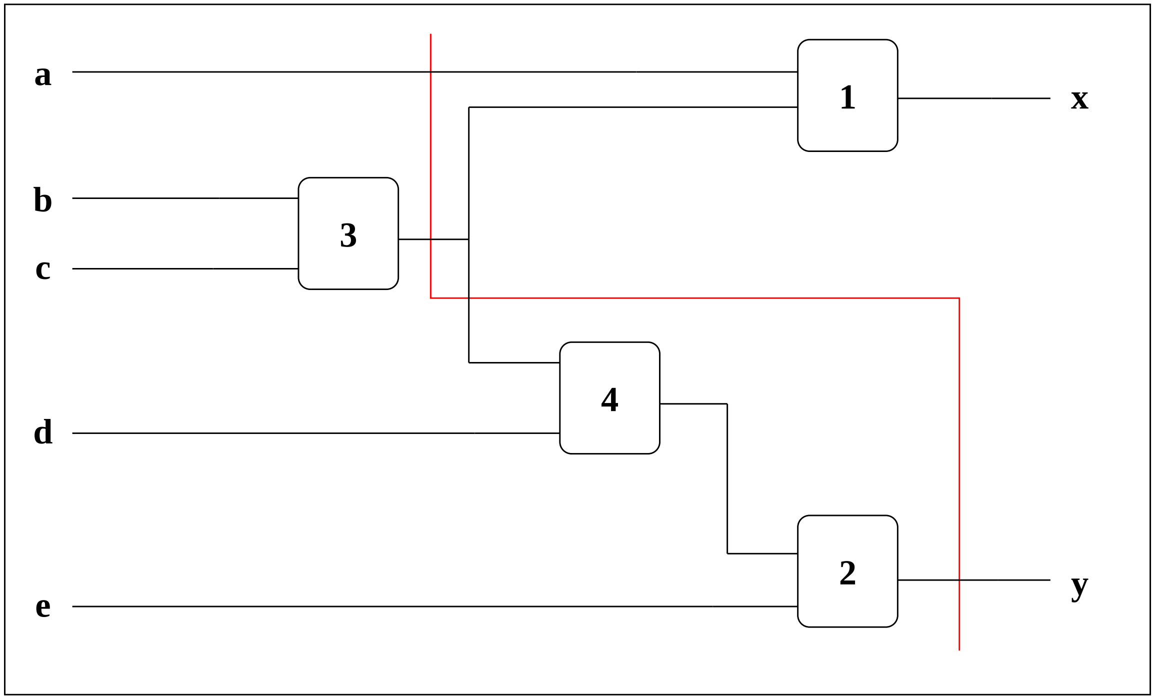
*BMDs of classical arithmetic operations are of linear size.

Backward construction algorithm - step 1



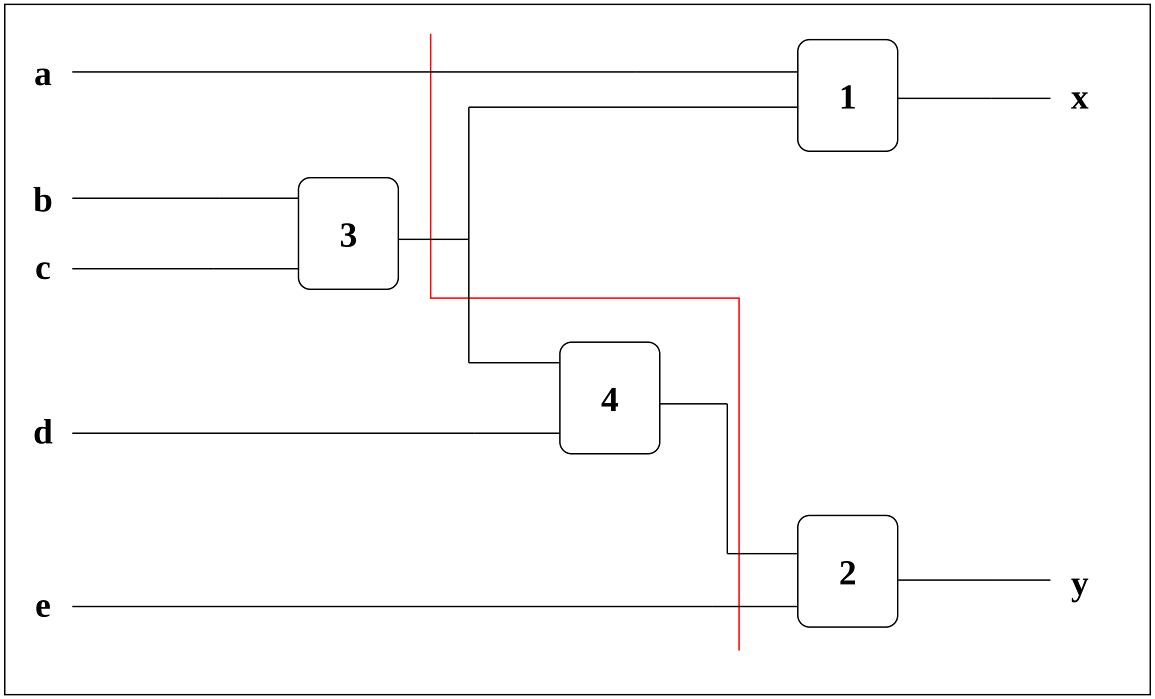
Beginning of the algorithm: the cut crosses all the primary outputs. The *BMD of the word-level interpretation of the output is constructed.

Backward construction algorithm - step 2



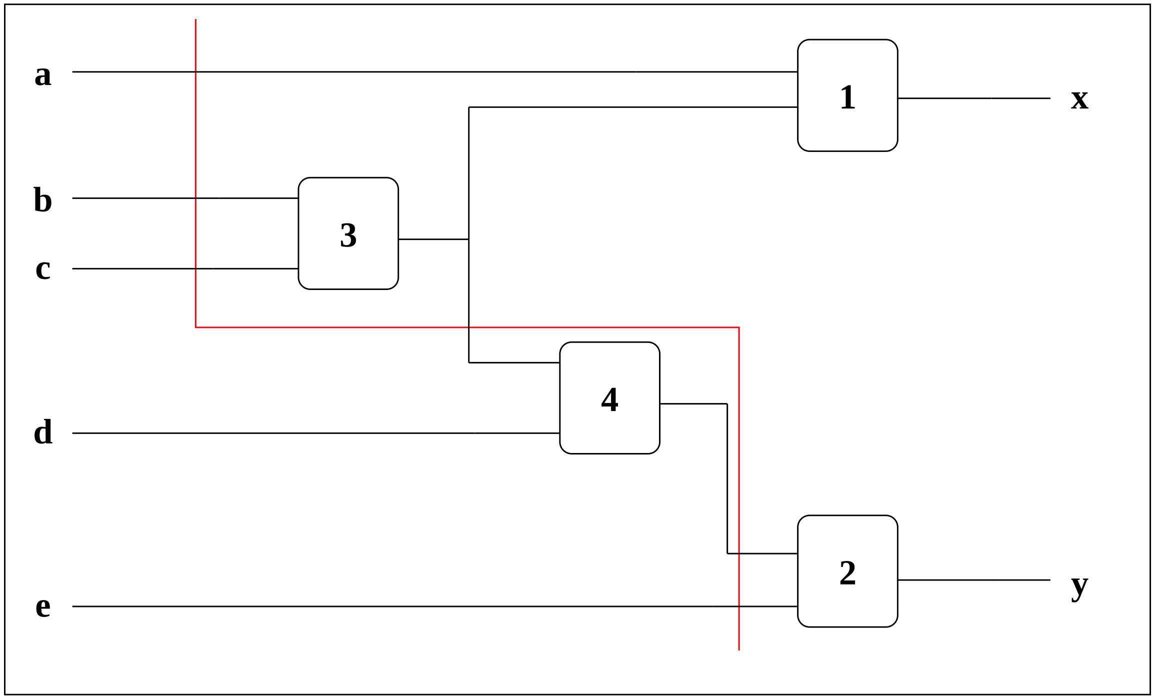
A gate just left to the cut is chosen and its output is substitued in the *BMD by the corresponding function of its inputs.

Backward construction algorithm - step 3



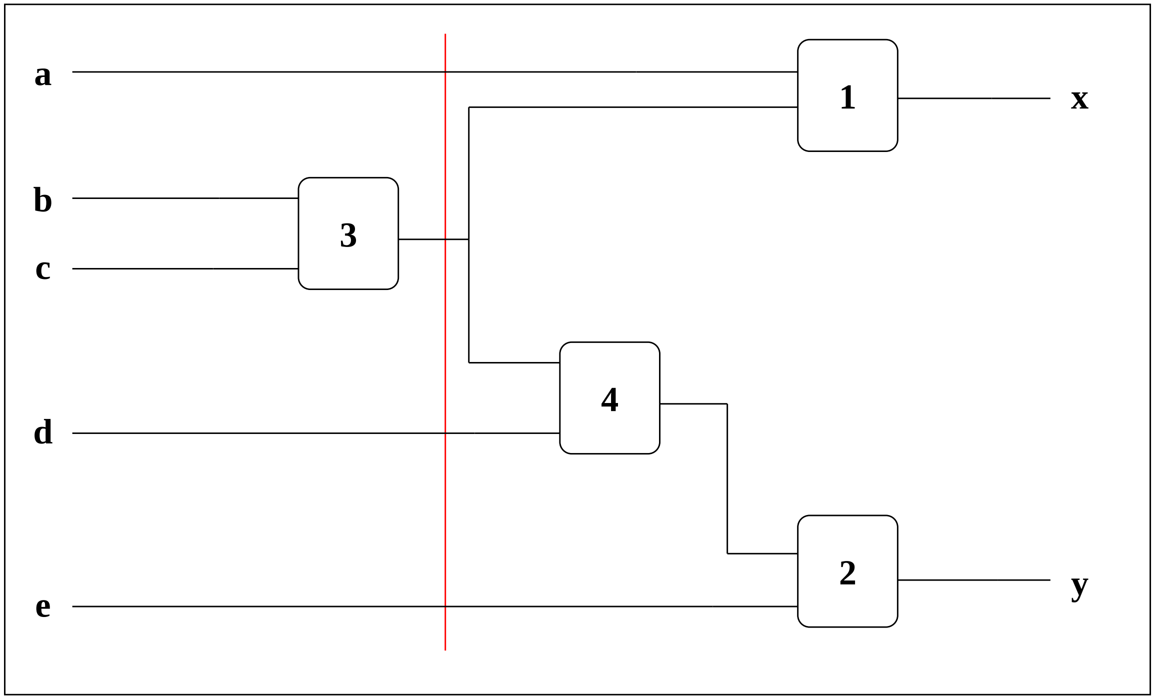
At any time, the *BMD expresses the word-level representation of the output as a function of the nets currently crossed by the cut.

Backward construction algorithm - step 4 (first try)



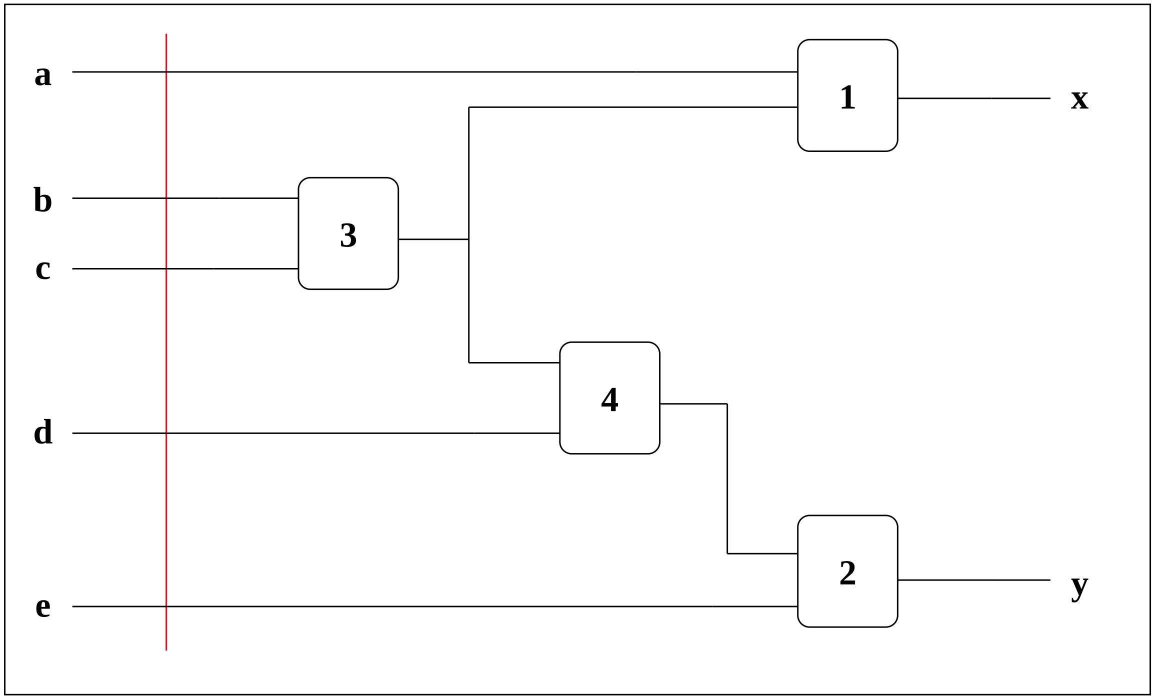
Problem: intermediary results must be kept!

Backward construction algorithm - step 4



Condition: a gate may be chosen only if its output is connected to only the input of the gates that have been already taken.

Backward construction algorithm - step 5



End of the algorithm: the cut crosses all primary outputs. The *BMD expresses the word-level representation of the output as a function of the inputs.

Add-step and carry-save multiplication

$$
\begin{array}{cccccc}
 & & & & 1 & 1 \\
\times & & & 1 & 1 & 1 \\
\hline
 & & & & 1 & 1 \\
+ & & & 1 & 1 & \cdot \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & 1 & 0 & 0 & 1 \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Add-step multiplication

$$
\begin{array}{cccccc}
 & & & & 1 & 1 \\
\times & & & 1 & 1 & 1 \\
\hline
 & & & & 1 & 1 \\
+ & & & 1 & 1 & \cdot \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & & \multicolumn{3}{r}{\{110, 11\}} \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & \multicolumn{4}{r}{\{1001, 1100\}} \\
\hline
 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Carry-save multiplication

Experimental results

Number of bits	Time Add-step (s)	Time Carry-save (s)
4	1	3
8	12	58
16	161	1115
32	2083	
	$O(n^{3.7})$	$O(n^{4.3})$

(Lava, Hotlips)

What now?

- Backward Construction Algorithm: very efficient, in comparison with former methods
- Still, need of something better: $O(n^4)$ is too much!
- Completely different direction?