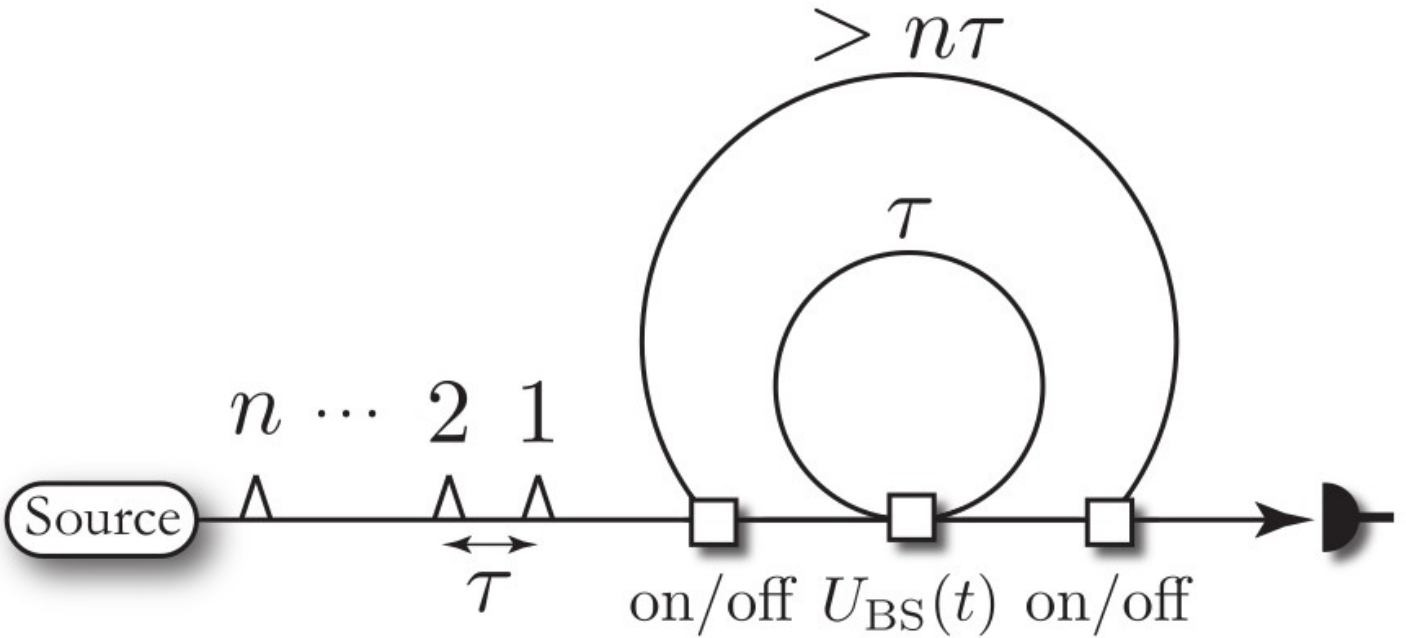
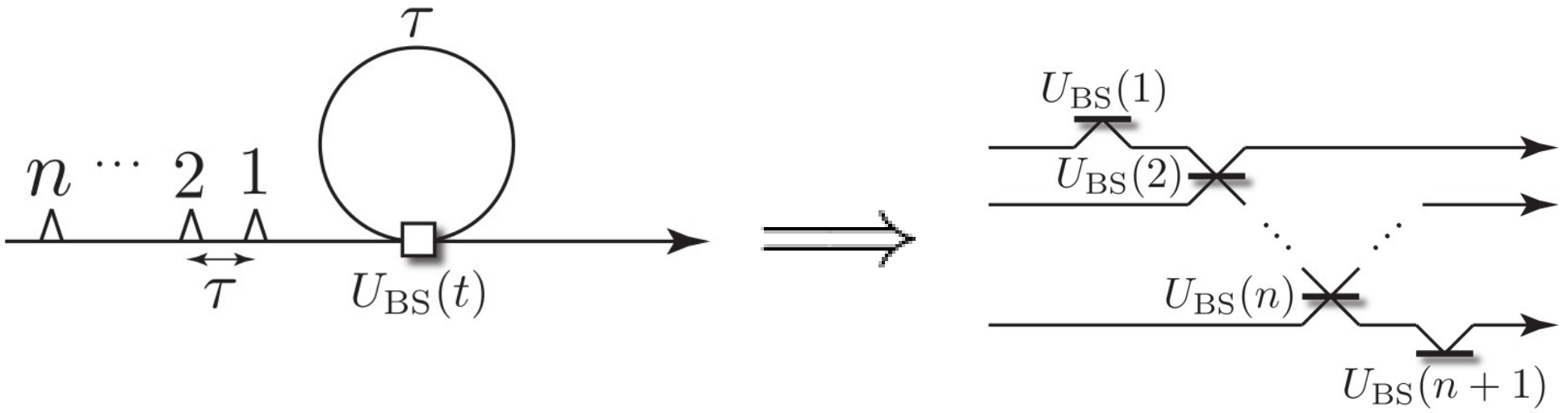
Graphs, trees and more...

in a photonic architecture!

Review: the loop architecture.

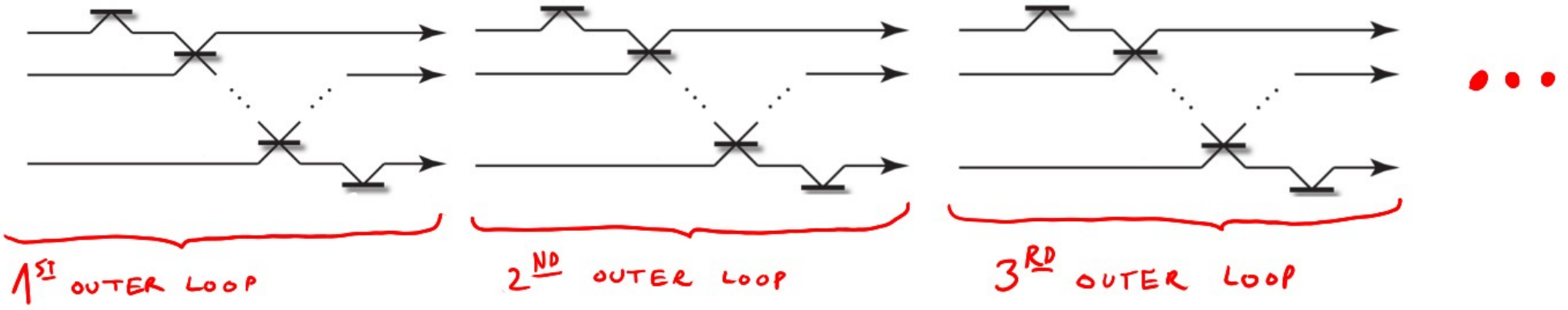


What can we do with just one loop?



Clearly, this is not universal!

Trick: add a big outer loop!



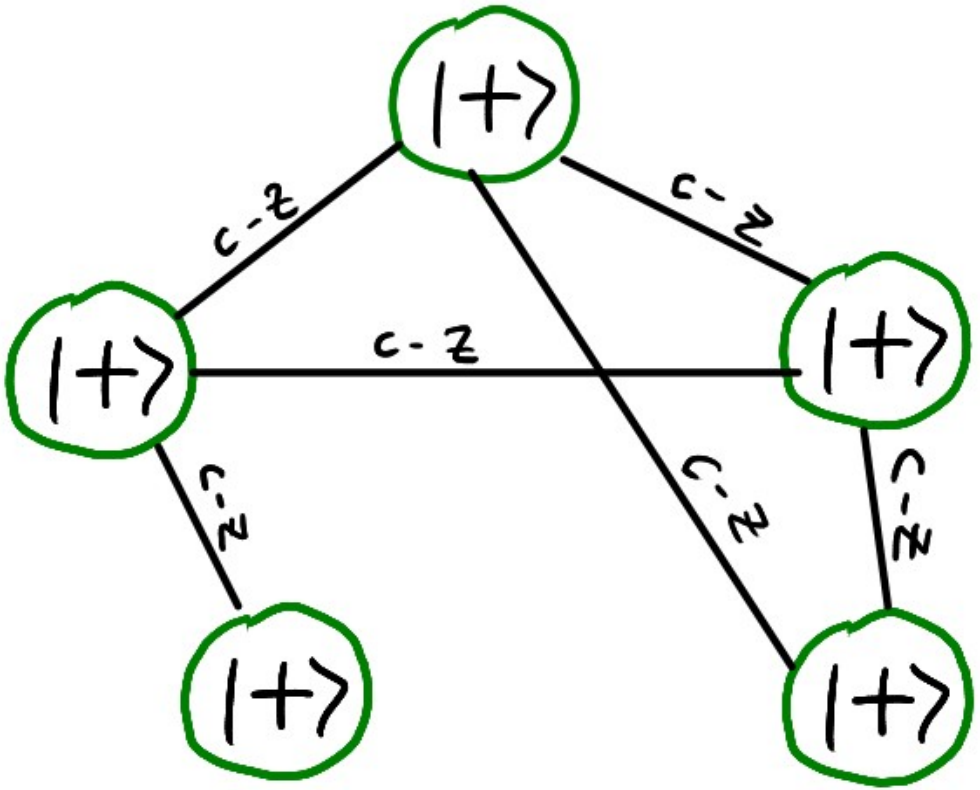
Now we are universal!

Peter Rodhe presented a constructive proof of the universality of this setup using the naive unitary decomposition presented in Reck *et al.*

This decomposition allows us to decompose a $U(d)$ unitary in d^2 beam-splitter unitaries $U(2)$, where d is the dimension of the Hilbert space. Clearly, this doesn't scale well, in fact $d = 2^n$ for a system of n qubits.

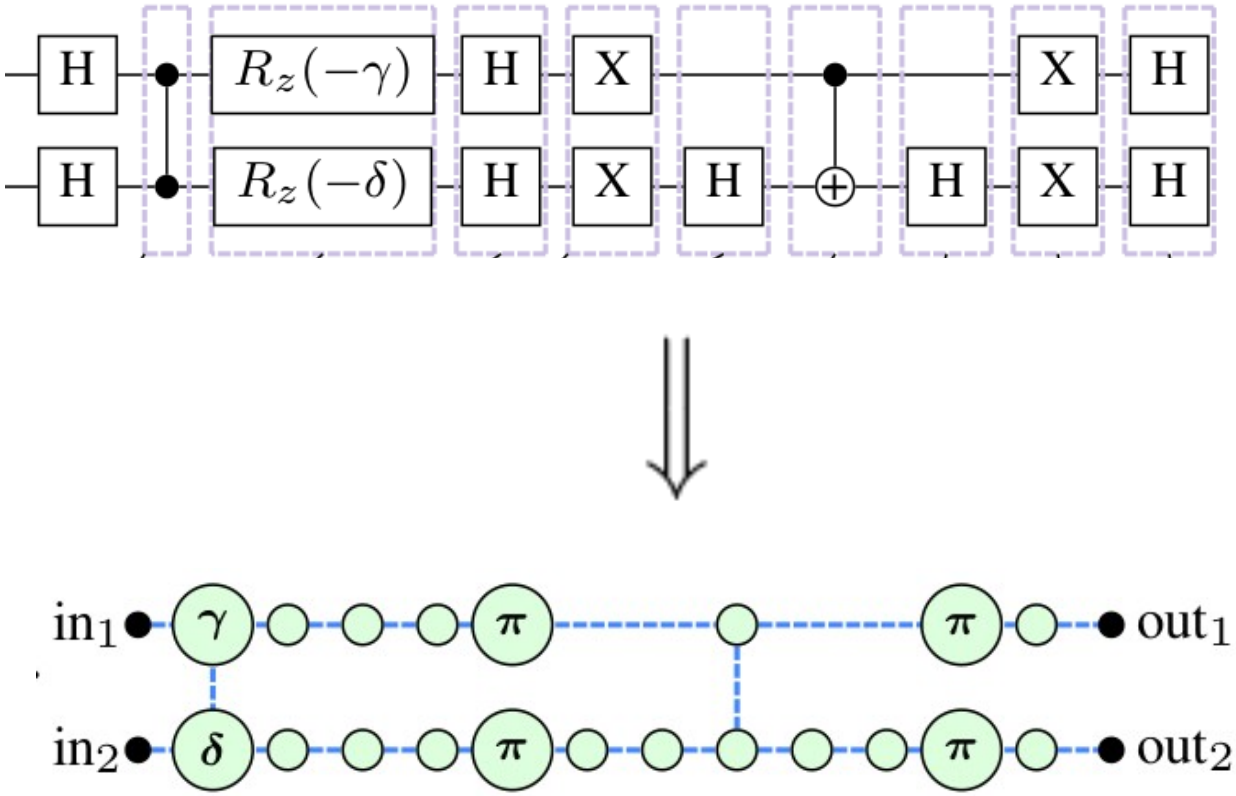
We can do much better!

Review: Graph states



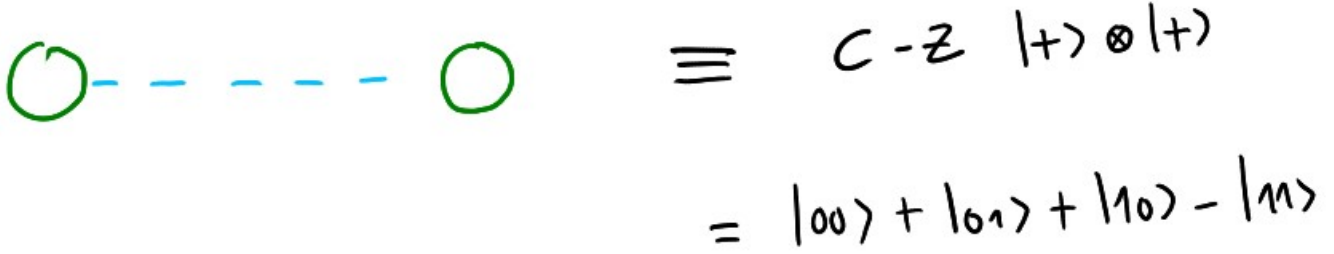
$\equiv |G\rangle$

Review: Measurement Based Quantum Computation (MBQC)



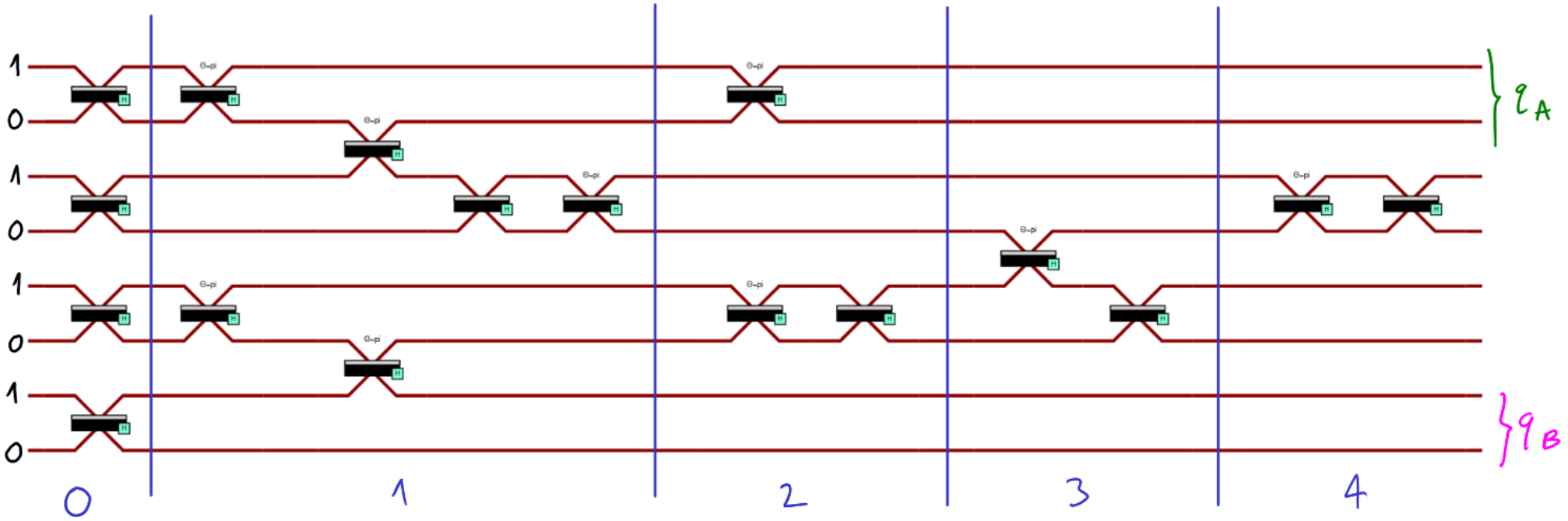
We want to build big graph states.

Two vertices graph states will be the building block of our computation:



We can generate them *in an heralded way* or *in the computational basis*.

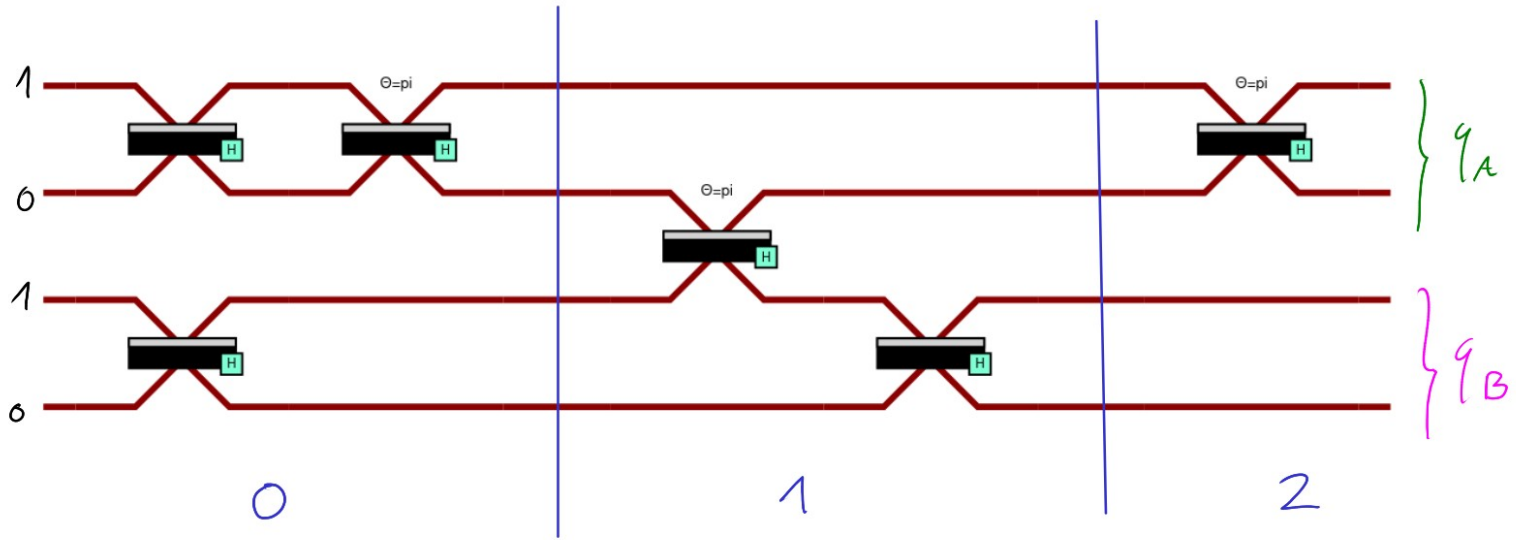
Heralded generation.



$$\text{OUTPUT} = |1010\rangle + |1001\rangle + |0110\rangle - |1010\rangle$$

$$= |00\rangle + |01\rangle + |10\rangle - |11\rangle$$

Generation in the computational basis.



$\frac{1}{2\sqrt{2}}\,|1010\rangle$

$\frac{1}{2\sqrt{2}}\,|1001\rangle$

$\frac{1}{2\sqrt{2}}\,|0110\rangle$

$-\frac{1}{2\sqrt{2}}\,|1010\rangle$

$\frac{1}{2\sqrt{2}}\,|0002\rangle$

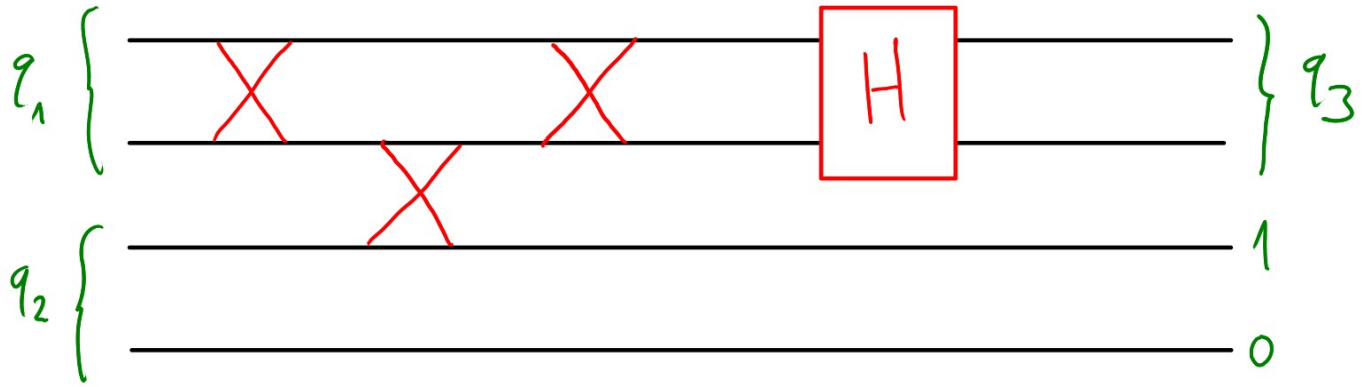
$\frac{1}{2\sqrt{2}}\,|0020\rangle$

$\frac{1}{2}\,|1100\rangle$

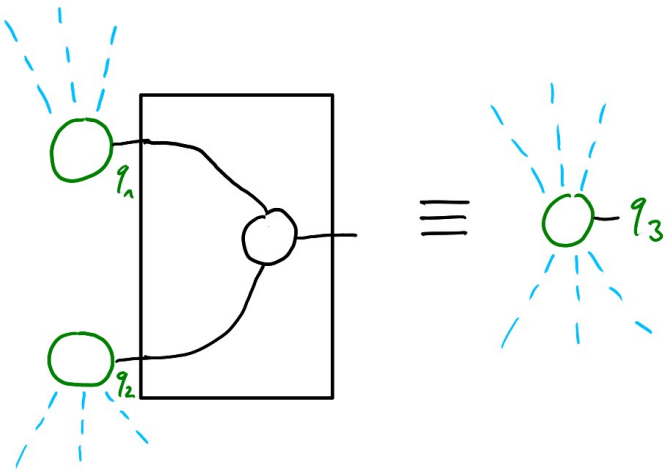
THROW AWAY

Just 4 time slots (instead of 8) and 2 outer loops (instead of 4)....

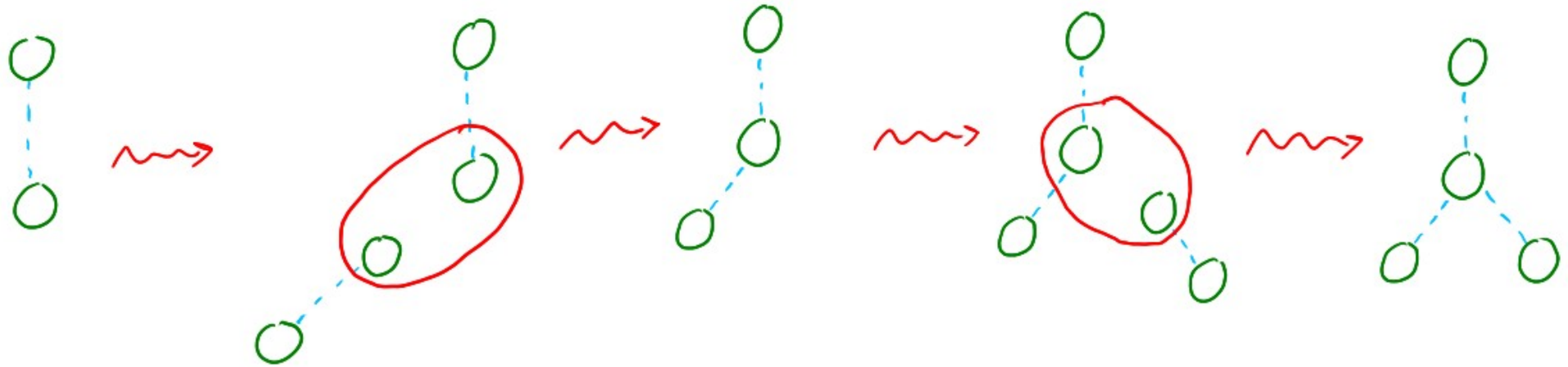
Gluing things together: Fusion



In the ZX-formalism, this is just the three legs spider node:



A simple example: three Bell pairs and two fusions.

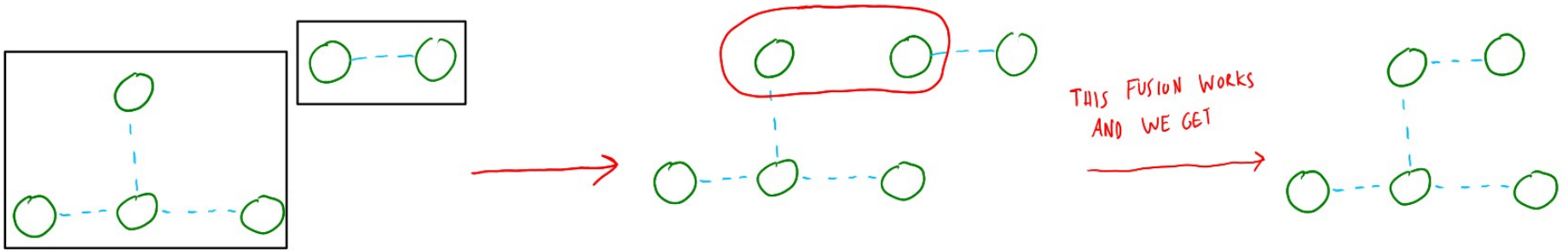


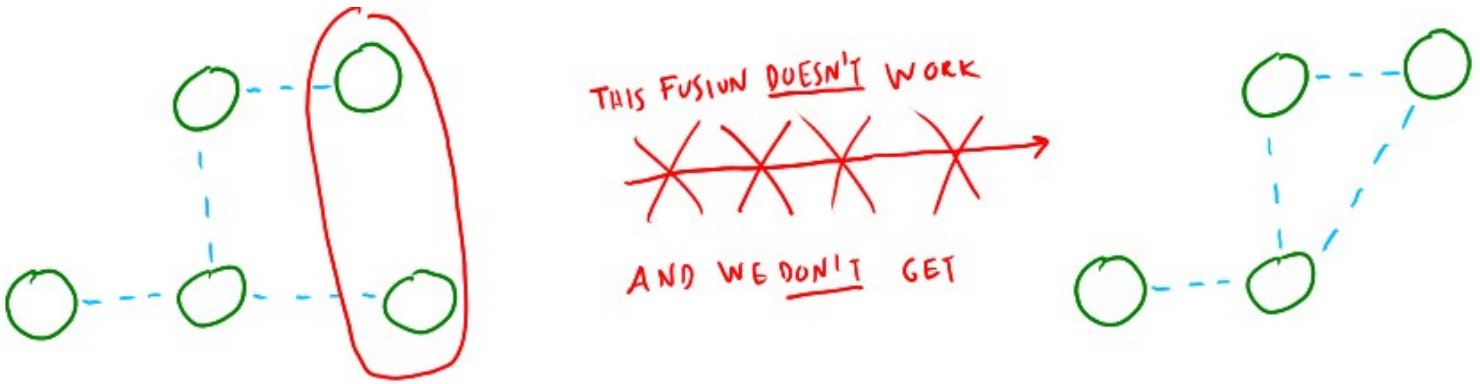
WARNING!

Fusing heralded pairs is **always** safe.
Instead, fusing pairs generated in the computational basis **doesn't always work**.

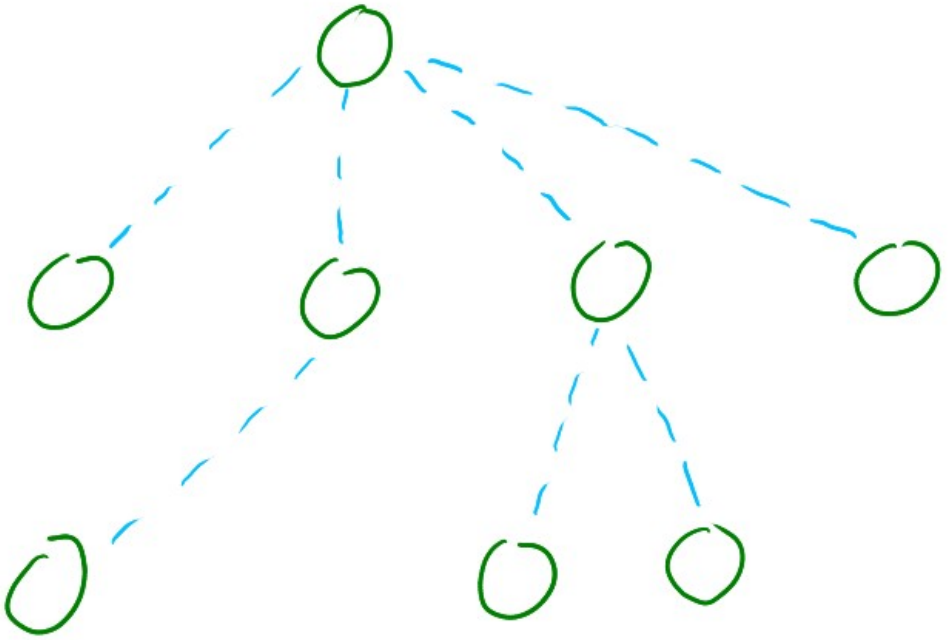
This observation motivates the first general result we proved: it's general because it works for *every* architecture that uses the computational basis, not just the loop architecture.

Theorem. Two states generated in the computational basis can always be fused together to get a new state in the computational basis.



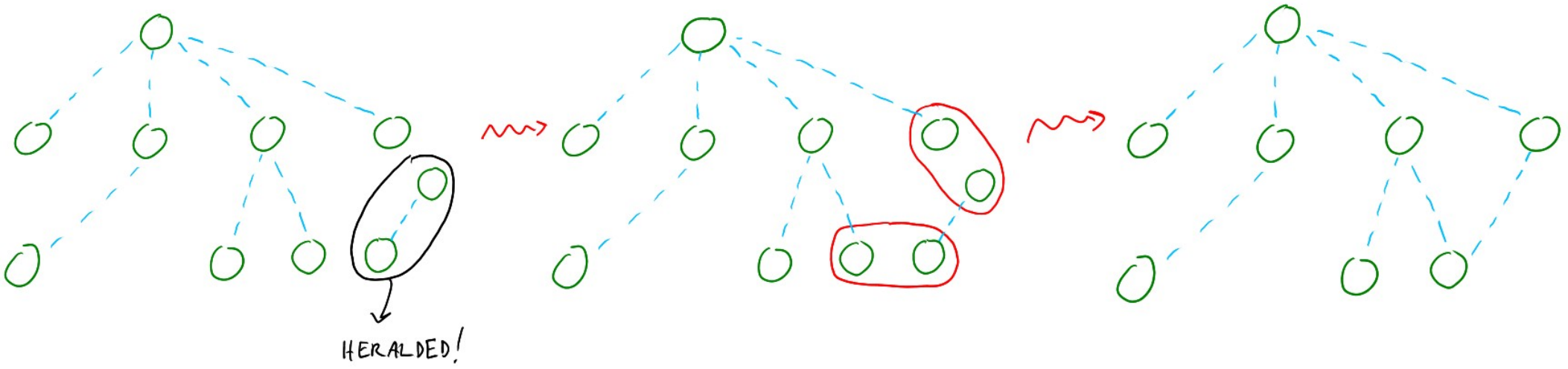


Easy corollary. We can build tree-like graph states with just Bell pairs in the computational basis and fusion.



What if we want to add cycles to our graph state?

Result. To add an edge inside a graph state in the computational basis we can use an heralded pair and the following procedure:



Interesting question. Up to local Cliffords, can we build every possible graph state (also with cycles) with just pairs in the computational basis and fusion?

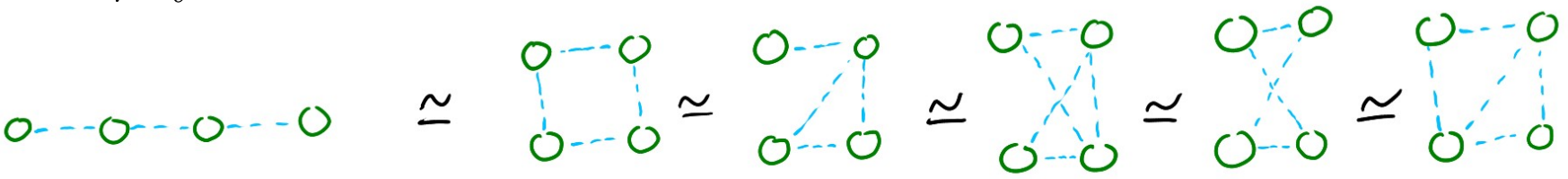
Short answer. No.

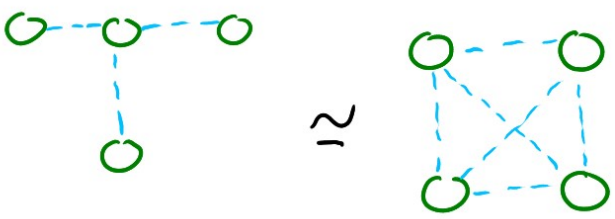
Longer answer. It depends on the number of qubits N.

- $N = 2$: yes, trivially

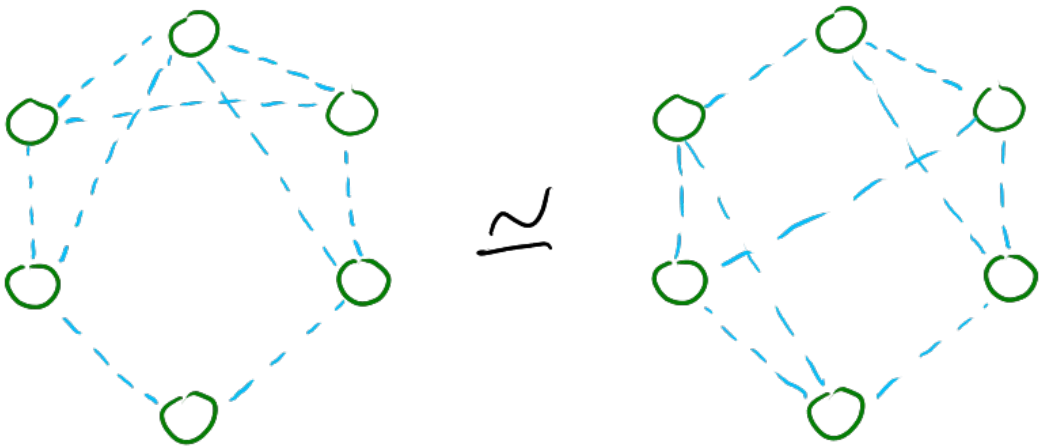
- $N = 3$: yes

- $N = 4$: yes





- $N = 5$: I'm not really sure
- $N = 6$: nope!



Another interesting question. Up to local unitaries, can we build every possible graph state (also with cycles) with just pairs in the computational basis and fusion?
Equivalently. Are tree universal for QC?

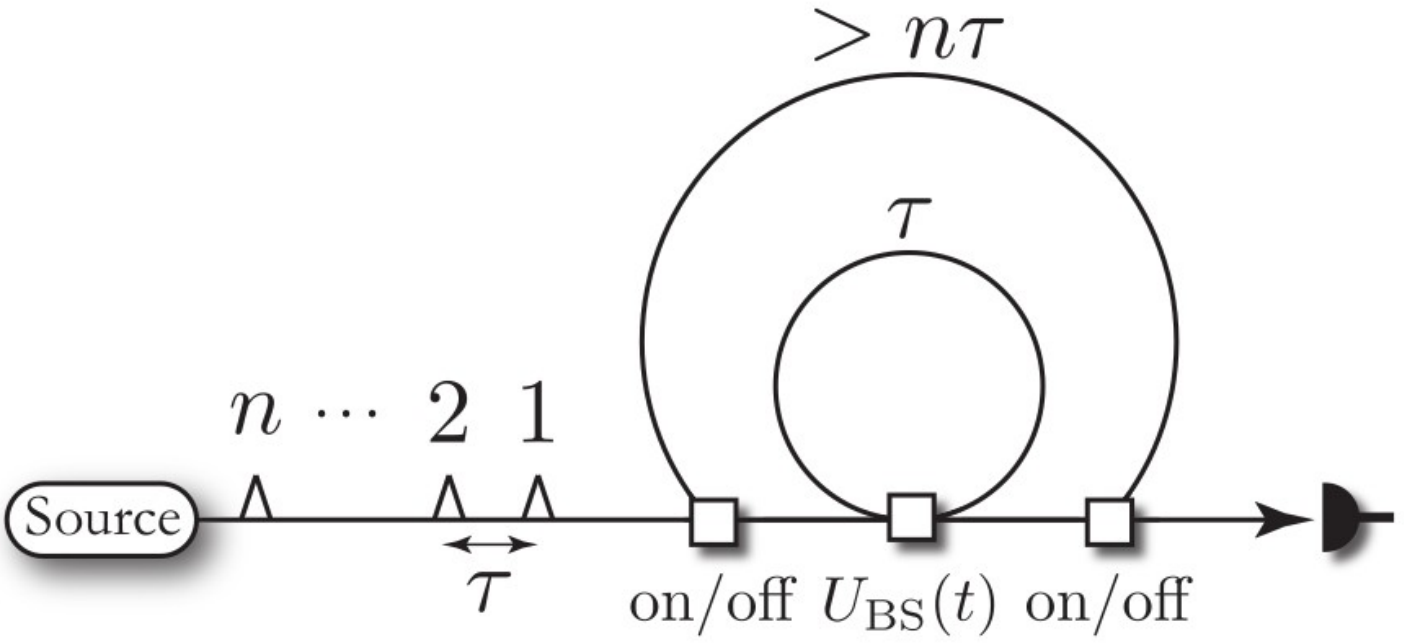
Answer. No.

Proof. Up to seven qubits, if two graph states are LC-equivalent they are also LU-equivalent.

Fact. Also stochastic local operations and classical communication (SLOCC) can't help us, because interestingly two graph states are LU-equivalent iff they are SLOCC-equivalent.

Back to the origins.

Review: the loop architecture.

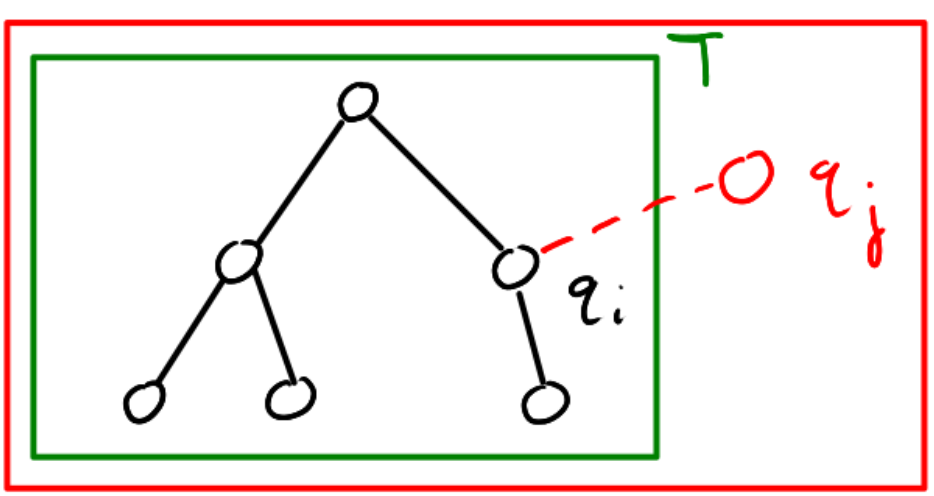


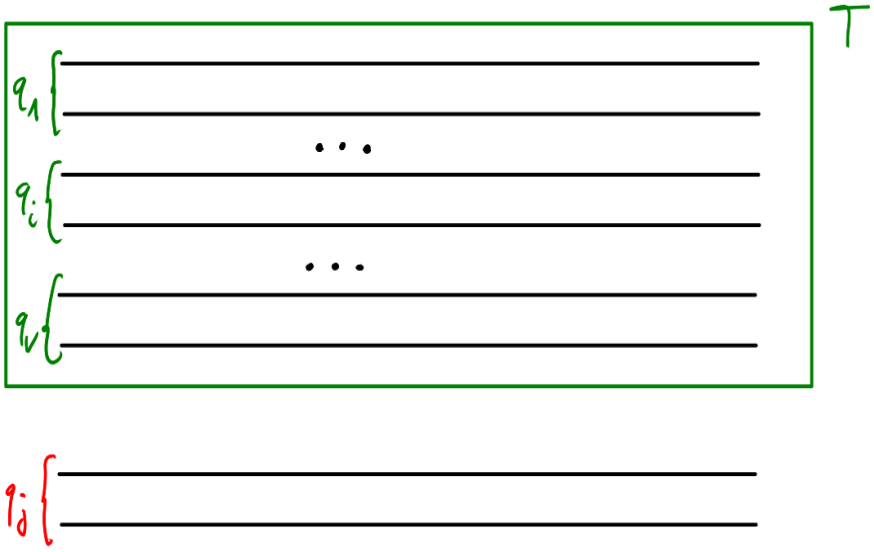
Question. *Given a tree T, how many outer loops do I need to implement it?*

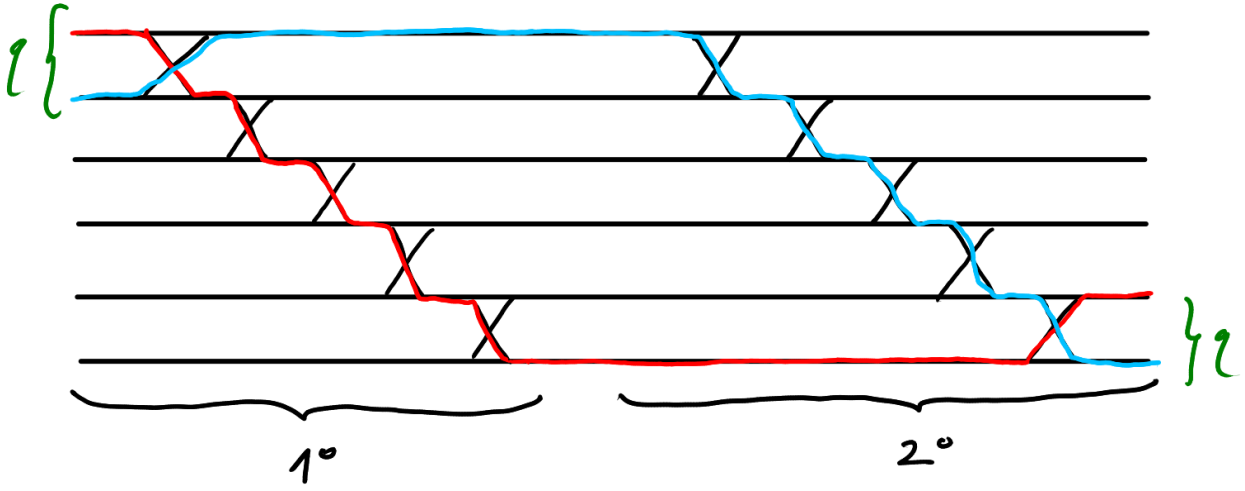
Answer. *It depends on the order O in which I add the vertices of T.*

We will study the function $C(T, O)$

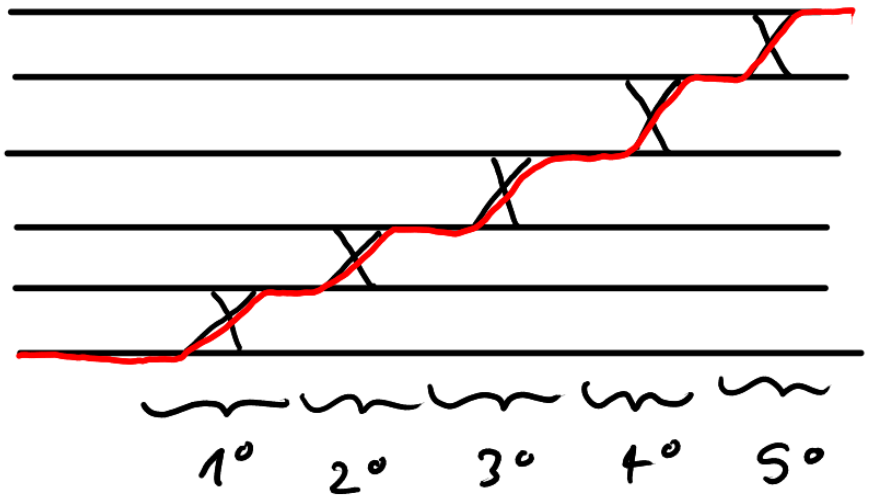
General algorithm to add a node.



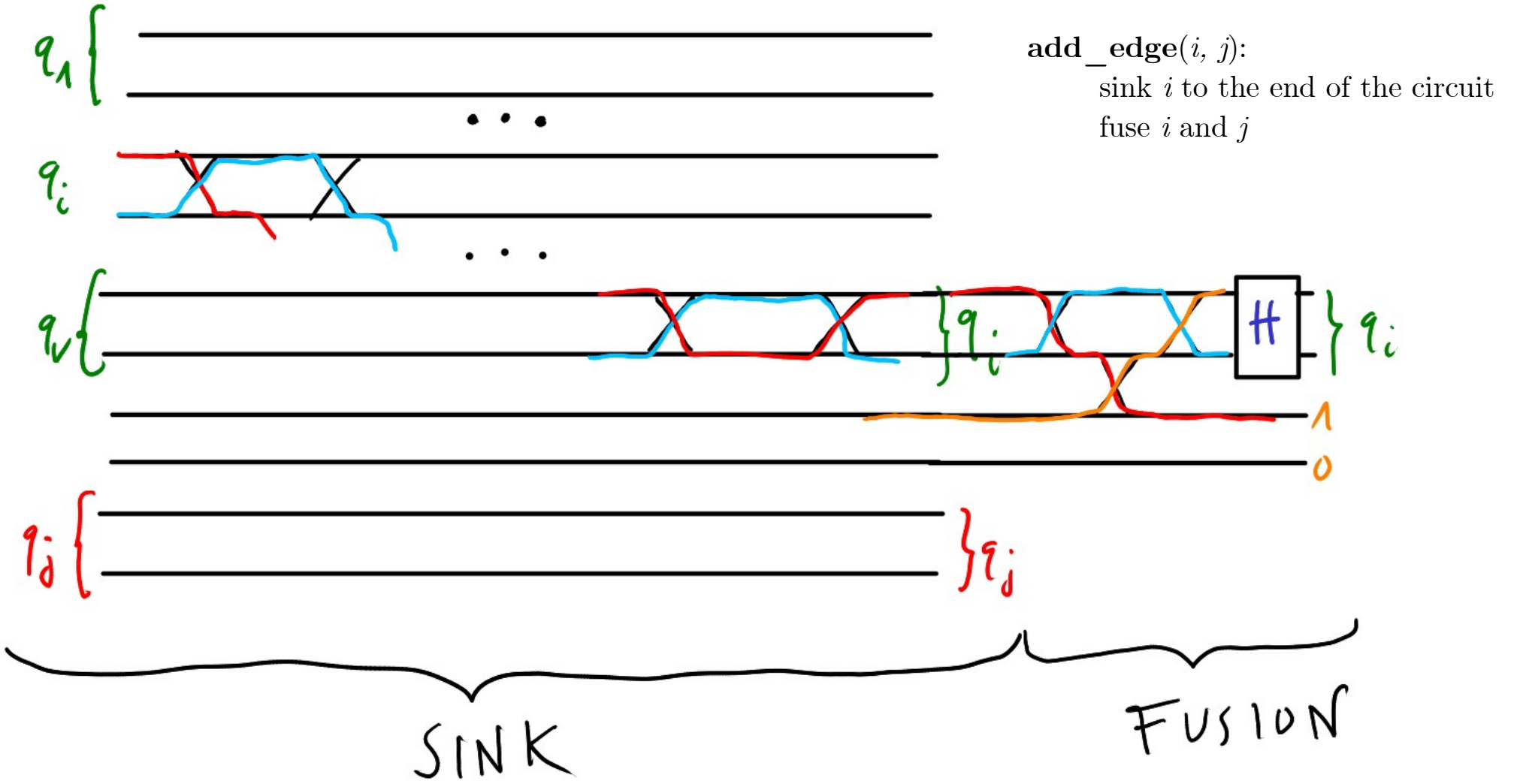




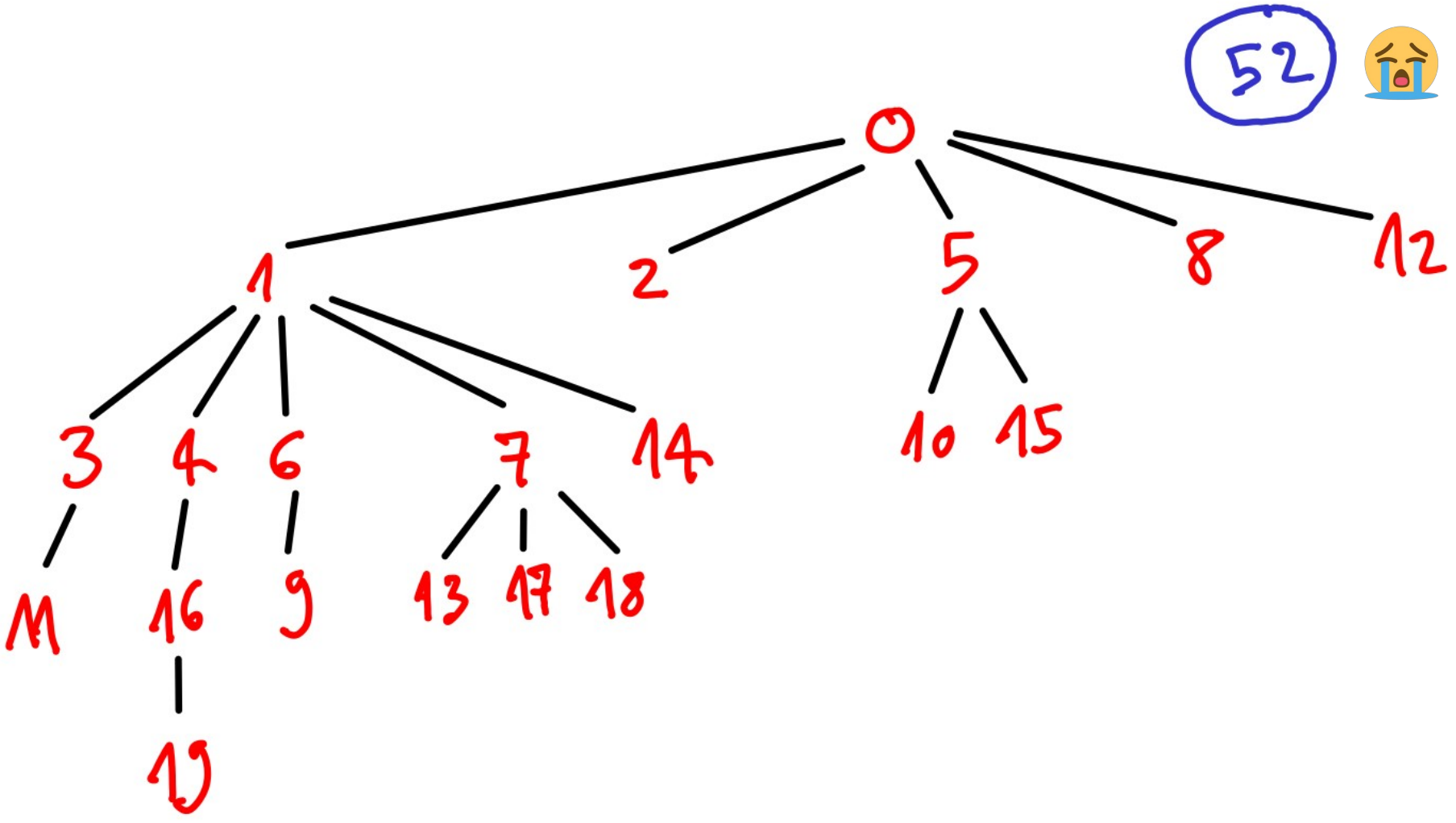
sinking :)



lifting :(



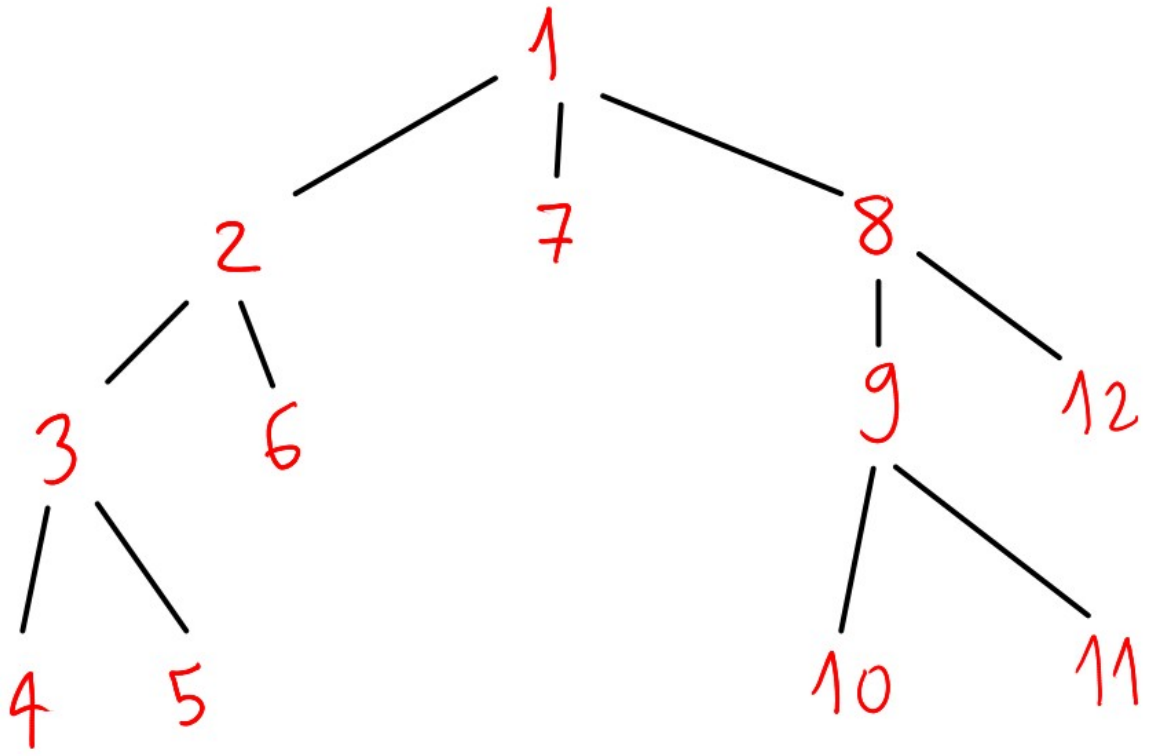
add_edge(i, j):
sink i to the end of the circuit
fuse i and j


52
0
1
2
5
8
12
3
4
6
7
14
10
15
11
16
9
13
17
18
19

A good class of orders:

DFS-orders

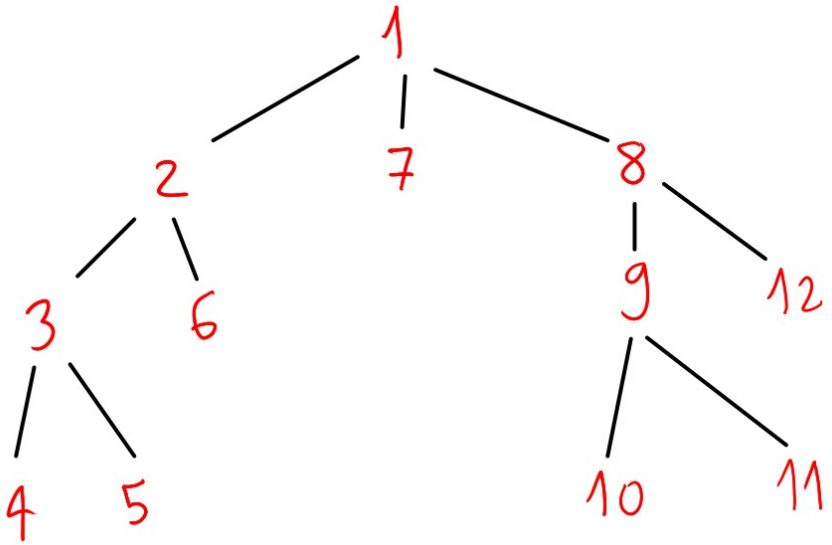
Review: the DFS algorithm



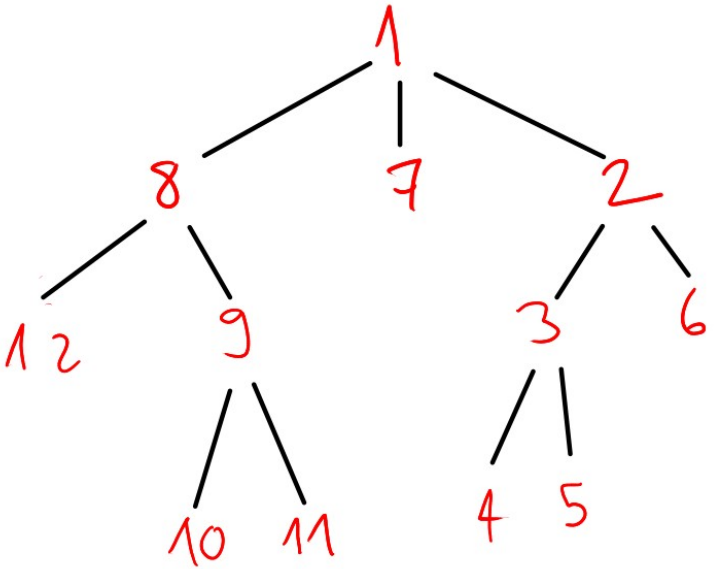
$$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12$$

The DFS algorithm gives us a possible order on the tree. But we are left with some freedom:

- we can decide in which vertex to start the DFS algorithm;
- given a vertex, we can move around it's children.



$\approx$

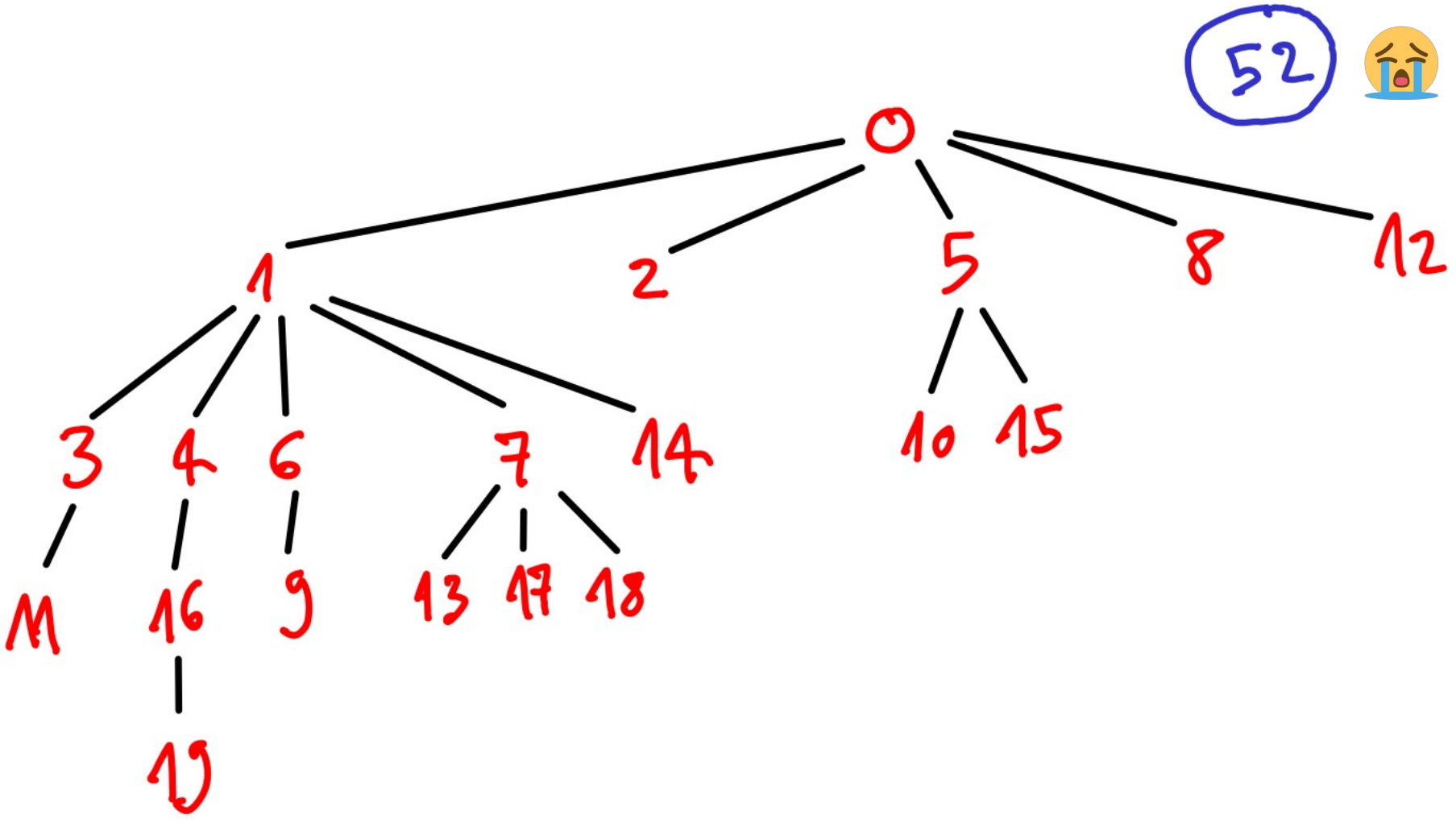


$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12$

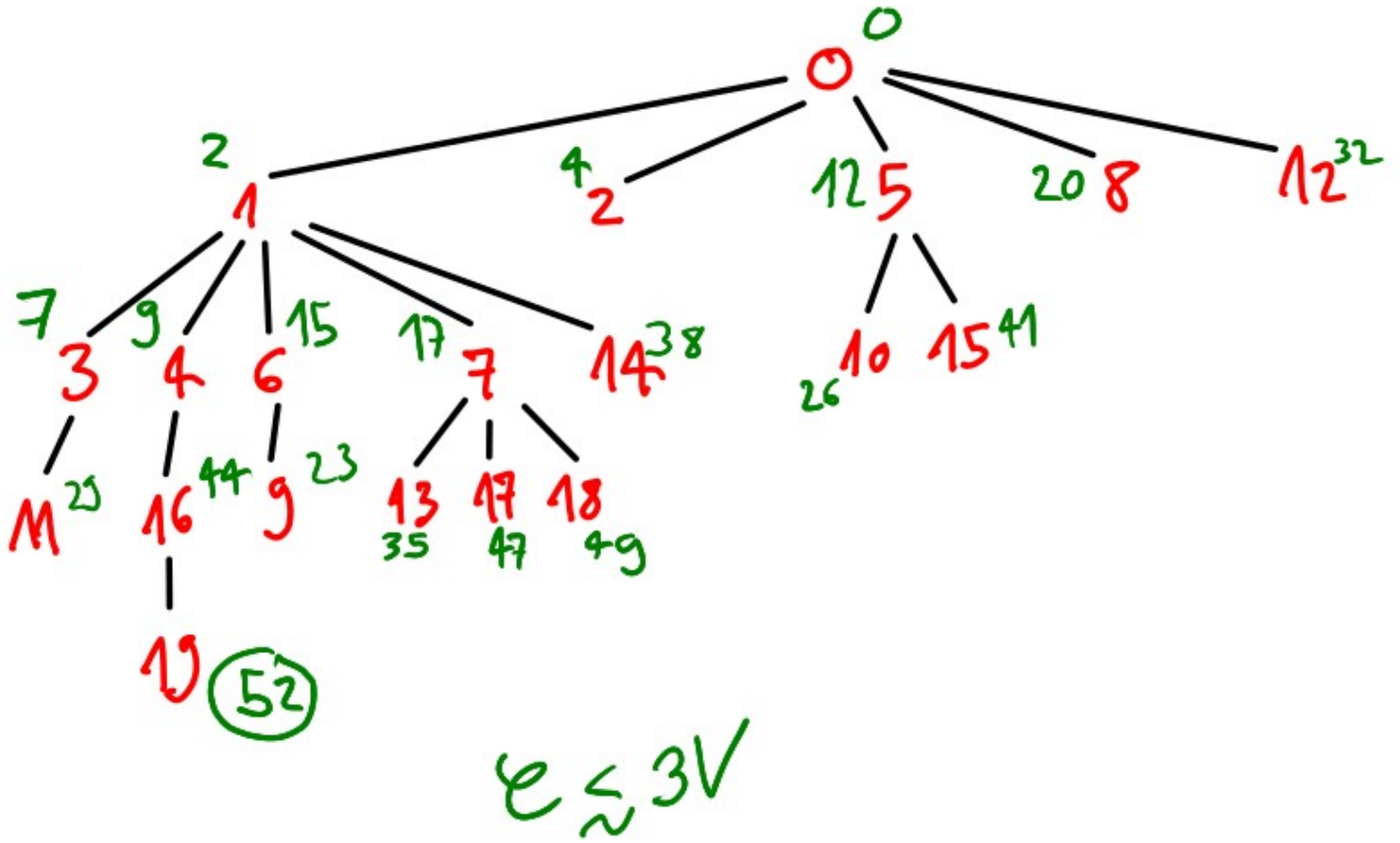
$1 \to 8 \to 12 \to 9 \to 10 \to 11 \to 7 \to 2 \to 3 \to 4 \to 5 \to 6$

Optimizing on the starting vertex is easy: it takes $O(V)$.
Optimizing on the children order is an nightmare: exploring all the possibilities would take exponential time.

Thankfully recursion and the DFS algorithm are best friends!


52
0
1
2
5
8
12
3
4
6
7
14
10
15
11
16
9
13
17
18
19

At what loop do we fuse?

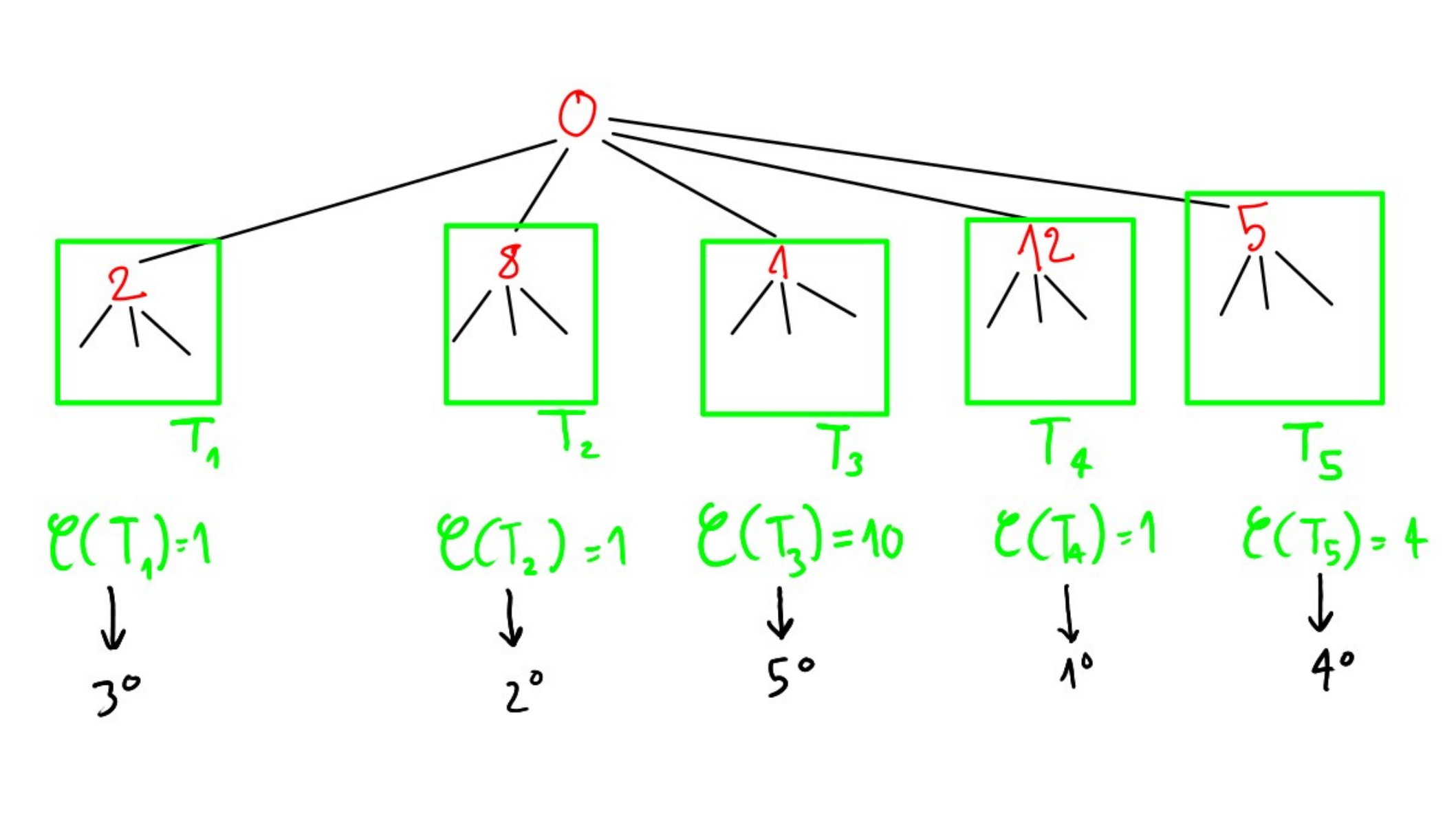


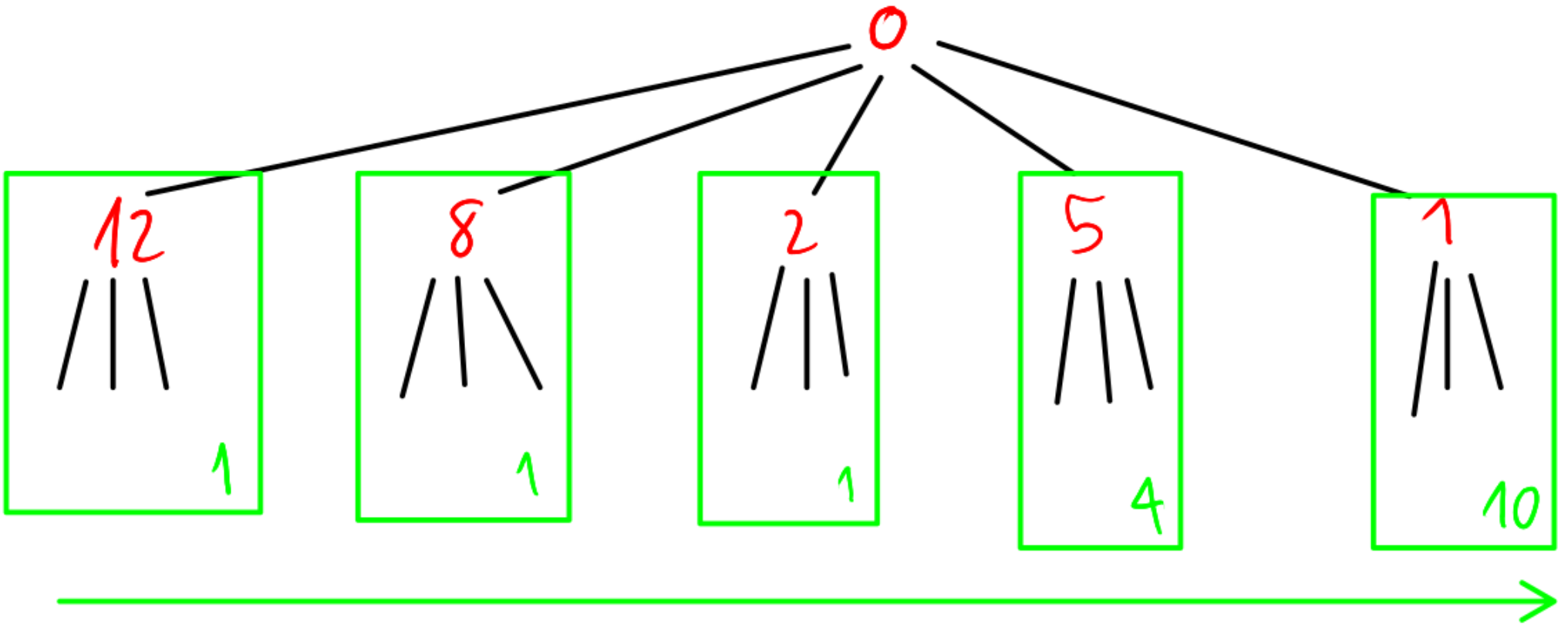
Easy bound valid for every order: every time we add a vertex we add at most 3 outer loops.

The optimal DFS-order

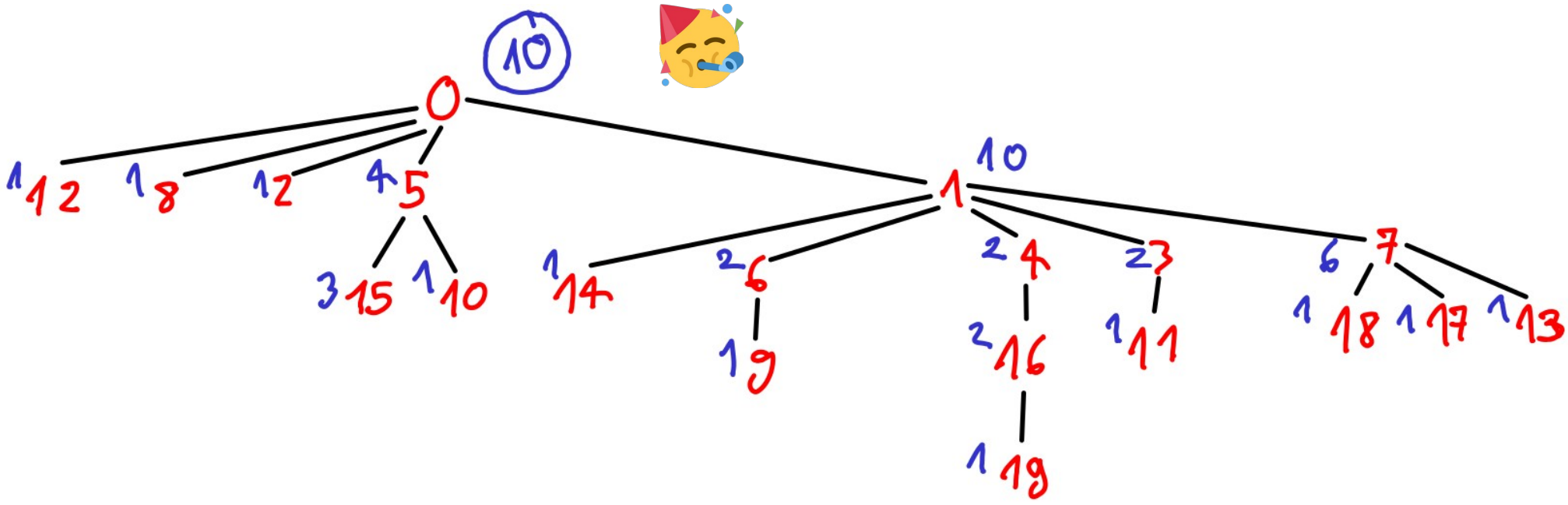
How to find it

(recursively)

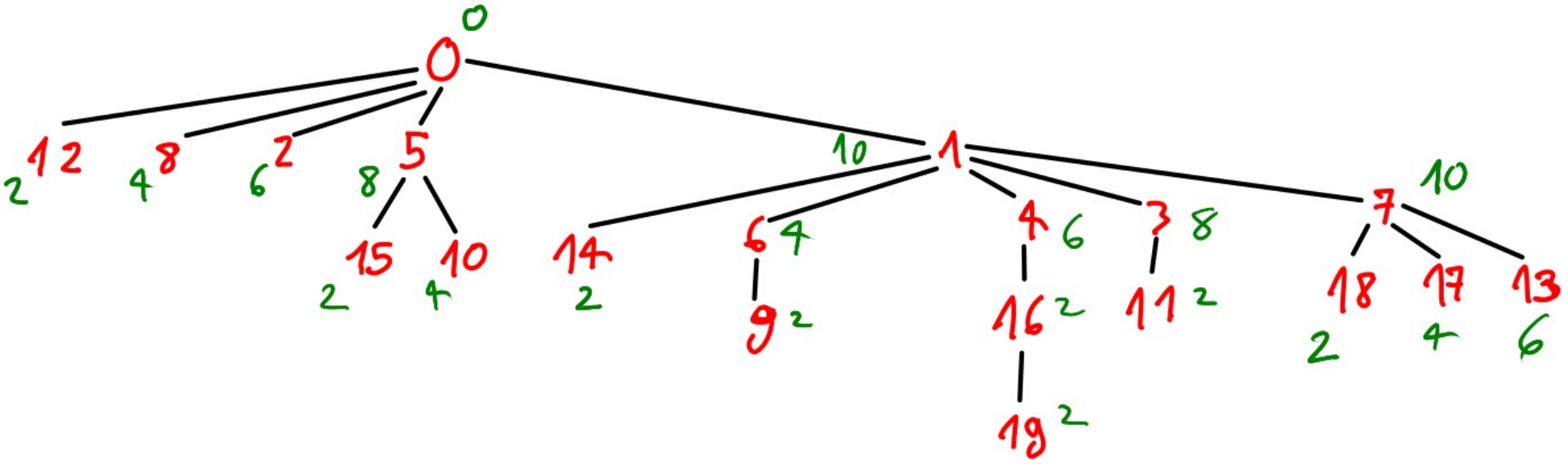

0
2
8
1
12
5
T_1
T_2
T_3
T_4
T_5
$e(T_1)=1$
$e(T_2)=1$
$e(T_3)=10$
$e(T_4)=1$
$e(T_5)=4$
3°
2°
5°
1°
4°


0
12
1
8
1
2
1
5
4
1
10

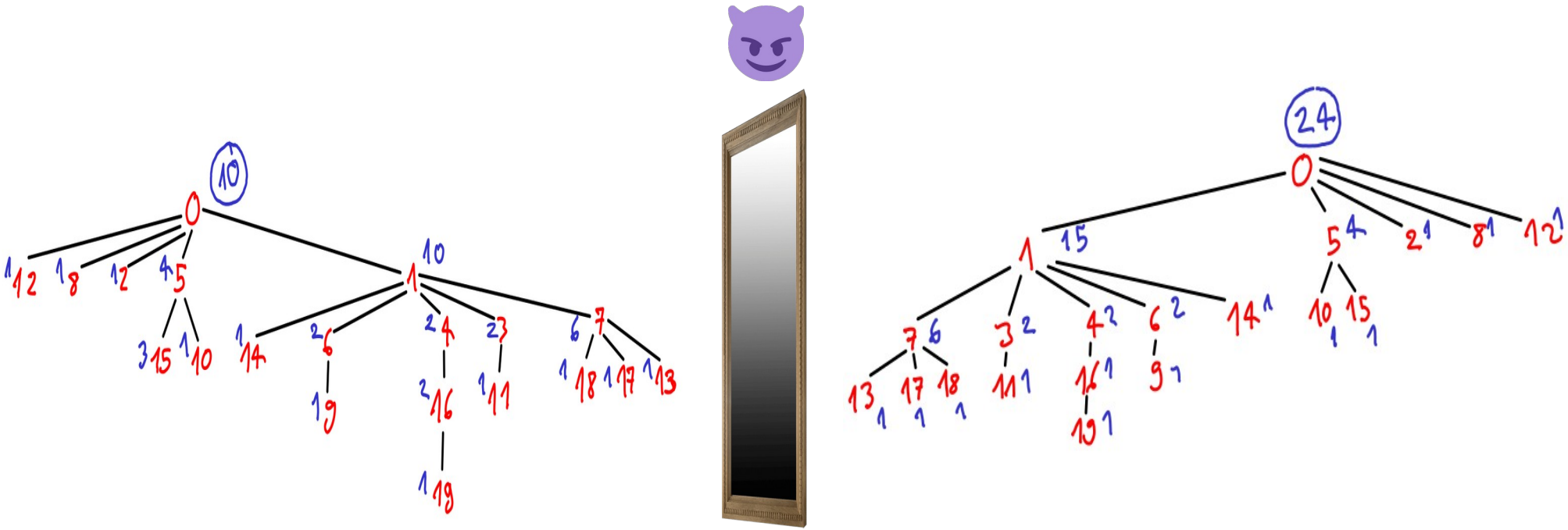
Recursive steps?



At what loop do we fuse?



What happens if a devil puts a mirror aside the DFS-optimal ordered tree?
We get the **worst** DFS-order!



In general the worst DFS-order is still better than the random order with started with!

To recap:

$$52 \gg 24 \gg 10$$

random 😭	worst DFS 😈	best DFS 🥳

We found our optimal answer. It depends on the tree T and the starting vertex v from which we start our recursion: $C(T, v)$.
Notice that this function C takes $O(V)$ time to be computed.

Recursion is a really powerful tool:

- we reduced the complexity of the optimization problem;
- it helps us also from a formal point of view, to write proofs and analytic bounds.

  And of course analytic bounds are not just some kind of overthinking that mathematicians like to do: a nice analytic bounds assure us that the algorithm we found is *reasonable* or even *optimal*!

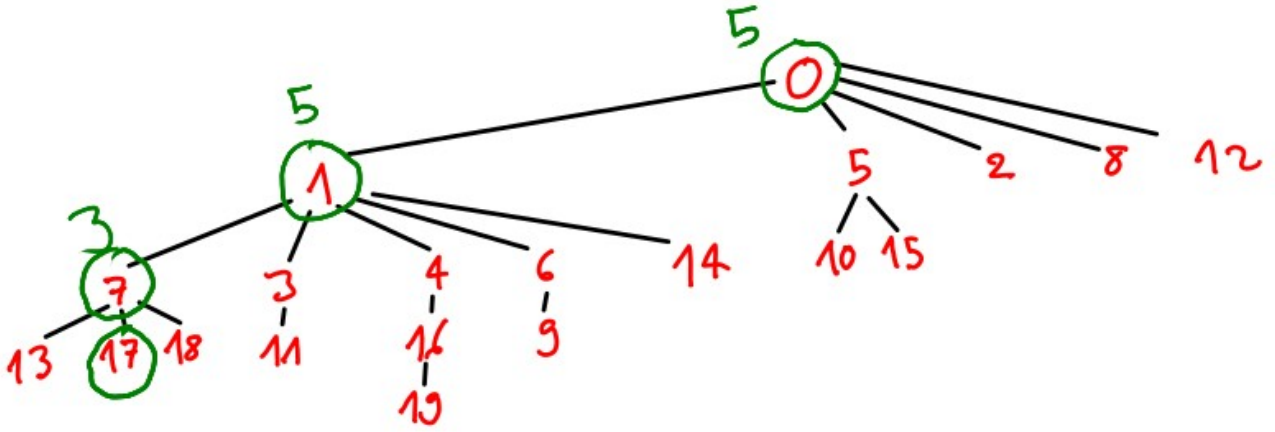
Theorem. Given a tree T and a DFS-order O on T that starts from the vertex h, we have the following bounds for the function $C(T, O)$:

$$\begin{aligned} C(T, O) &\leq \max\{b(p)\} \text{ on all the possible paths from } h \text{ to a leaf} \\ &\leq (d\text{-}1)(\max\{c(v)\} - 1) + 3 \text{ for all the vertices } v \text{ in the tree} \end{aligned}$$

where d is the tree depth, $c(v)$ is the number of children of the vertex v and $b(p)$ is the sum of $c(v) - 1$ for all the vertices v on the path.

$C(T,\ O)$ $\leq \max\{b(p)\}$ on all the possible paths from 0 to a leaf

$\leq$ (d-1)($\max\{c(v)\} - 1$)+3 for all the vertices v in the tree



$24 \leq 3((5-1)+(5-1)+(3-1))+3 = 33$

$\leq 3 \cdot 3 \cdot 4 + 3 = 39$

DEPTH

MAX # OF CHILDREN - 1

$$(\ d\text{-}1\ )(\max\{c(v)\} - 1)+3$$

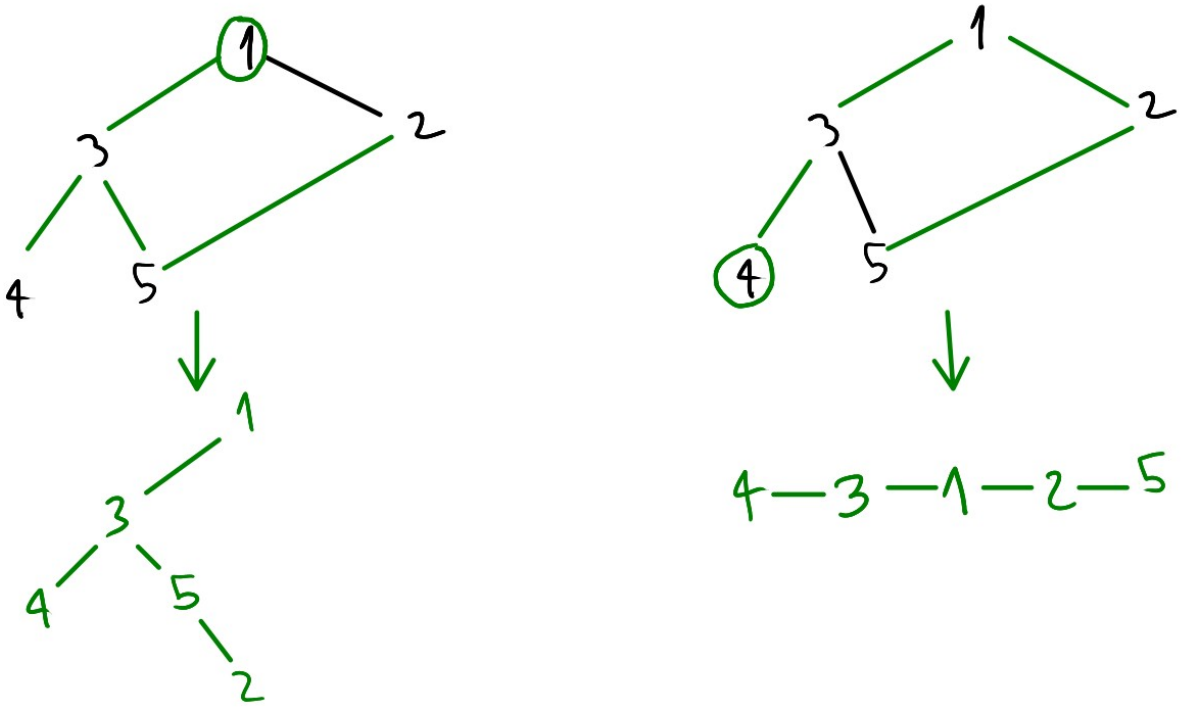
This last bound is really important, because it tells us that all the DFS-orders are in some sense *really good* orders. In fact the depth of the tree usually grows as $\ln(V)$, while the maximum number of children is a constant.

Given a tree T, we understood how to build the optimal circuit from it. Let's go back to the original problem.

Question. *Given a graph state G, how many outer loops do I need to implement it?*

Answer. *It depends on what tree we extract from the graph.*

Review: the DFS algorithm to extract a tree from a generic graph.



The topology of the extracted tree depends on the starting vertex!

We need to iterate on all the possible starting vertices to extract V possible trees ($O(V)$)

optimal_circuit(**given a graph state** G):

```
answer = ∞
for every vertex v in G:
    extract tree T given by a DFS starting from v
    for every vertex v' in T:
        answer = min(answer, C(T, v' ))
```

It takes $O(V^3)$

One last step: some ZX-calculus optimization

Now we just have to optimize on all the ZX-graphs that are ZX-equivalent to our original unitary U. For example, this allows us to play with *local complementation* (and thus *pivoting*) to get a smaller optical circuit in the end. In other words we are interested in

$$\min \mathsf{optimal\_circuit}(G) \text{ on all } G \text{ that are ZX-equivalent to } U$$

Future developments.

Introducing the concepts of **gflow** and time cones in our MBQC gives us the possibility to have a variable number of inner loop, because we can "measure on the way".
For now, the number of inner loop is fixed by the number of vertices of the graph state, so we just seek the smallest number of outer loops.
The ZX-optimization is more challenging, because we want to restrict ourself to the subset of ZX-graphs with a **gflow** and nowadays is still an active area of research to characterize the ZX-operations that preserve **gflow**.