Building the tree

Abstract

We present the precise mathematical algorithm that allows us to calculate how the complexity of a loop circuit grows when adding a specific vertex to a tree-shaped graph state. We then proceed to discuss how this complexity function grows when building the tree in a generic DFS order. We thus show that an optimal DFS order can be found. Furthermore we present a series of elementary bounds for the complexity of the circuit that apply for evey DFS order.

Contents

1 Preliminaries: characterizing the circuit

Let's start by some definitions.

Tree. A tree is a graph with no cycles. We will use a stronger definition: a tree is a graph with no cycles equipped with a special vertex called *head* of the the tree. Notice that the presence of the head induces a hierararchy: if we start building the tree from the head, when we add the vertex v, we know where to fuse it (to its parent $p(v)$).

Order on a tree. We define an order O on a tree T an array of vertices $[h, v_1, v_2, \dots]$ where h is the head of the tree. This list represents the order in which we want to add the vertices to the tree: thus the parent $p(v)$ of a vertex v must be present in the list before v.

Depth of a photonic line. Given a photonic line, there will be a last optical element on this line (remember that in our setup every optical element is a $SU(2)$ matrix that occupies two photonic lines). This optical element will belong to a specific outer loop numbered n, we define the *depth* of the photonic line as this number n.

Complexity of a circuit. The *complexity* of a circuit is the maximal depth of its photonic lines.

This series of definitions allow us to introduce our main character, the *complexity function* $C(T, O)$:

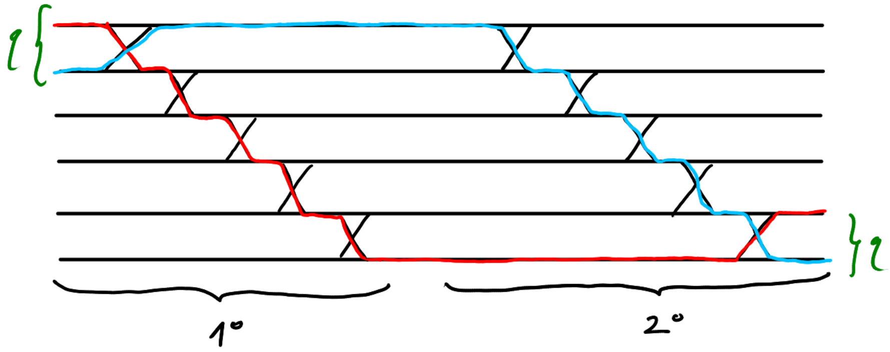
The complexity function. Given a tree and an order on this tree, this function $C(T, O)$ tells us the number of outer loops needed to implement the circuit.

2 The general algorithm

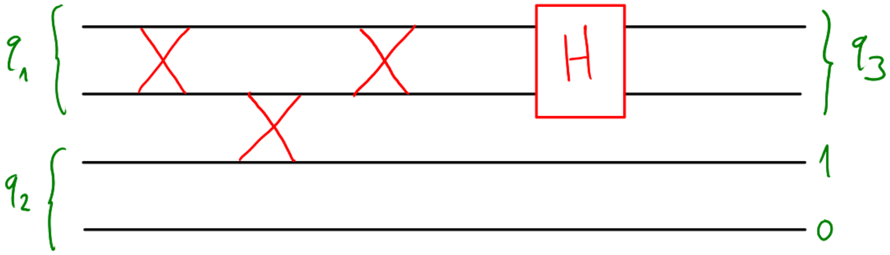
2.1 On the circuit

We now present a general algorithm to add a new vertex q_j and the edge connecting it to its parent $p(q_j) = q_i$. We need two fundamental blocks:

Sinking. Sinking a qubit q brings it at the end of the circuit.

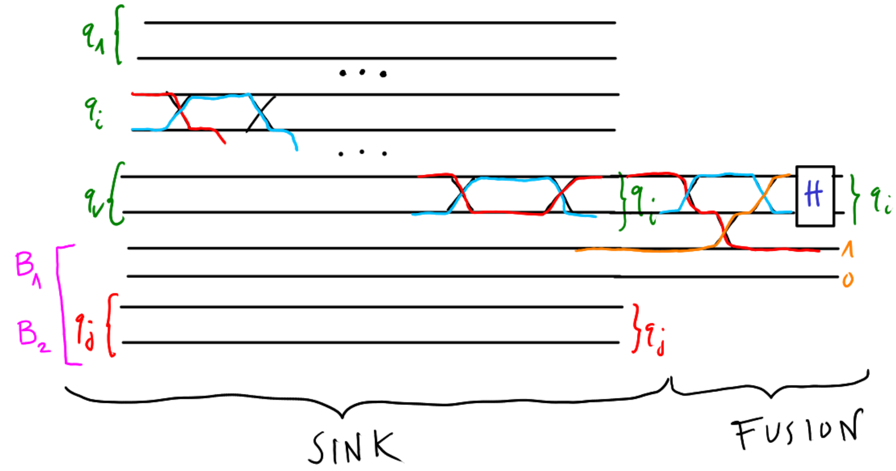


Fusing two neighbors qubits.



The general algorithm. To add a new vertex q_j and the edge connecting q_i and q_j:

- we sink q_i at the end of the circuit;
- we append a Bell pair at the end of the circuit (made of two qubits (B_1, B_2))
- we fuse the now neighbors q_i and B_1.



2.2 On paper

To get a feeling of what's happening here and how the complexity function grow when adding an edge, let's describe an algorithm to compute $C(T, O)$ for a generic order O. Let's start from a crucial observation that will simplify a lot our line of reasoning.

Observation. Every qubit is made of two photonic lines. In general the depth of these two photonic lines may be different. But if we just perform sinking and fusing operations, given a qubit q the two photonic lines that make up the qubit have always the same depth. This allow to define the *depth* of qubits and not just of photonic lines, given the fact that its photonic lines have the same depth. Just look at the previous pictures to convince yourself!

We make use of two lists:

- *order*: it's an ordered list that follows the order of the qubits in the photonic circuit. Clearly sinking the qubit q places q at the end of *order*. Fusing a at the end of the circuit appends a 1 to the list order.
- *depth*: the element $depth[i]$ gives us the depth of the qubit $order[i]$.

Now let's reason on how the two operations of *fusing* and *sinking* act on the two lists:

1. *fusing* a qubit q increments +1 the last element of *depth* and then appends a 1 to the list *depth*; it also appends the value q to the list *order*;
2. *sinking* a qubit q moves the value q at the end of the list *order*. The action on the depth list is much more complicate. There are two different

possible cases:

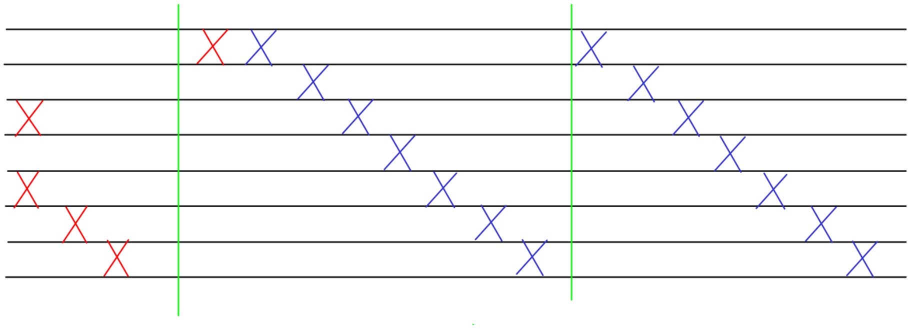


Figure 1: In the first case the qubit q I want to sink has the biggest depth. In this case I just need one more outer loop.

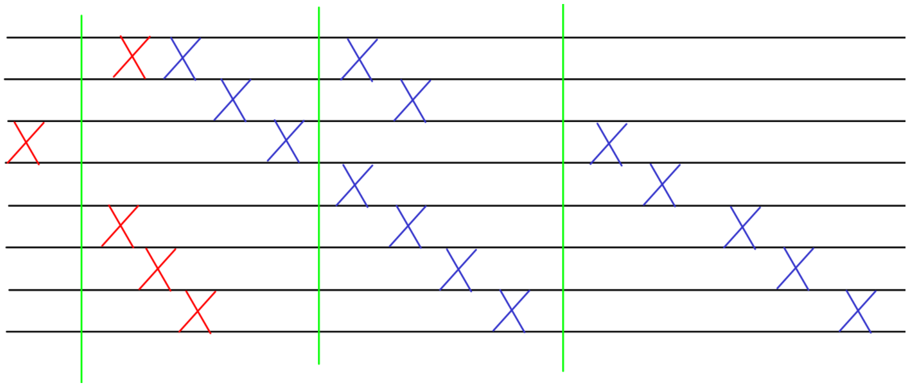


Figure 2: In the first case the qubit q I want to sink has not the biggest depth (for example there are other qubits with the same depth). In this case I will need two more outer loop.

This is summarized in the following function:

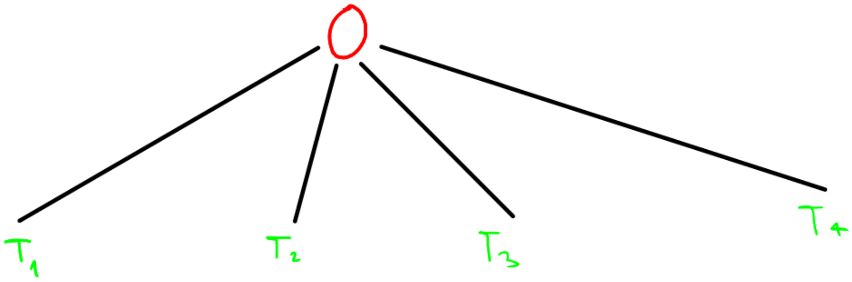
```
    #sinking...
    if order.index(parent) < len(order)-1: #check if the parent qubit isn't already at the end of the circuit
            m = depth[order.index(parent)]
            for i in range(order.index(parent)+1, len(order)):
                m = max(m, depth[i]+1)
                depth[i-1] = m+1
                order[i-1] = order[i]
            order[len(order)-1] = parent
```

```
          depth[len(order)-1] = m+1

#fusion...
      depth[len(order)-1] += 1
      order.append(leaf)
      depth.append(1)
```

3 Some bounds for the DFS-orders

3.1 The main idea



Suppose we are building a tree following this scheme and using a DFS-order. The number of outer loops will have the following upper bound:

$$C(T) \leq C(T_1) \tag{1}$$

after building and attaching the first subtree. Suppose we attach the second subtree T_2:

$$C(T) \leq \max\left(C(T_1)+3, C(T_2)\right) \tag{2}$$

this is because sinking the 0 vertex to the bottom of the circuit we use at maximum $C(T_1)+3$ outer loops. And so on:

$$C(T) \leq \max\left(\max\left(C(T_1)+3, C(T_2)\right)+3, C(T_3)\right) \tag{3}$$

$$C(T) \leq \max\left(\max\left(\max\left(C(T_1)+3, C(T_2)\right)+3, C(T_3)\right)+3, C(T_4)\right)\ldots \tag{4}$$

The expression can be rewritten as:

$$C(T) \leq \max\left(C(T_1)+3\times 4, C(T_2)+3\times 3, C(T_3)+3\times 2, C(T_4)+3\times 1, C(T_5)\right) \tag{5}$$

This last expression is extremely important and gives us the euristic for the optimal-DFS. To minimize this max function we want $C(T_1) \leq C(T_2) \leq C(T_3) \ldots$. The smallest subtree to the the left! This also explain how to build the worst-DFS.

Similarly we can build a lower bound for $C(T)$, using the fact that:

$$C(T) \geq \max\left(C(T_1)+2, C(T_2)\right) \tag{6}$$

$$\ldots \tag{7}$$

$$C(T) \geq \max\left(C(T_1)+2\times 4, C(T_2)+2\times 3, C(T_3)+2\times 2, C(T_4)+2\times 1, C(T_5)\right) \tag{8}$$

3.2 Recursive bounds

We now present a recursive upper bound that will gift us two nice analytical bounds.

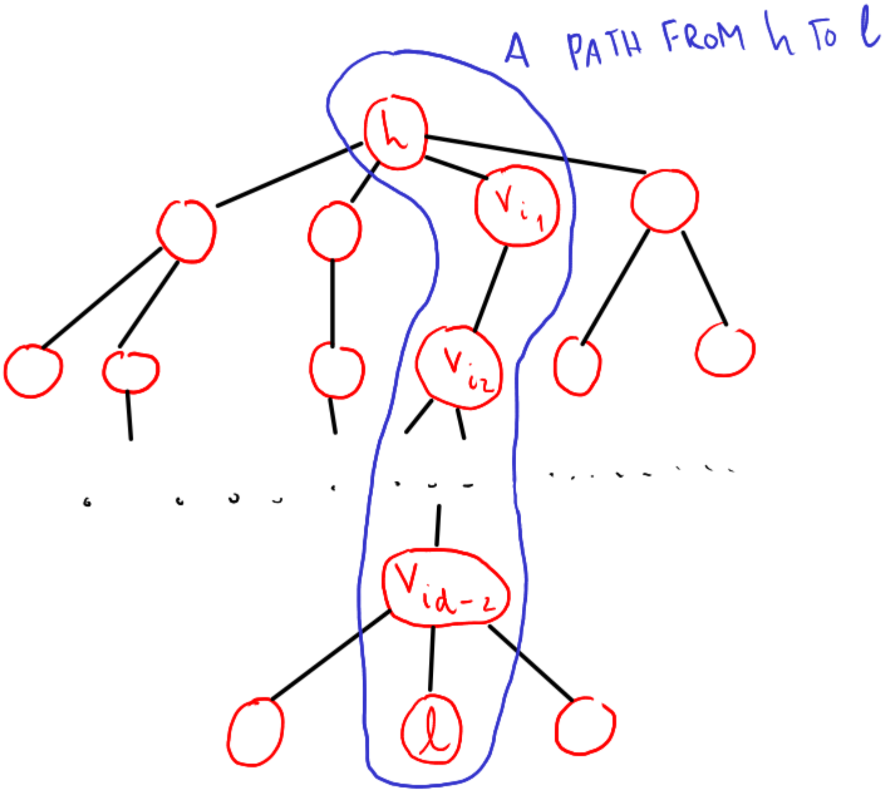
Start by assigning $B(l) = 3$ for every leaf l. Now we want to compute the value for some vertex v with children c_1, c_2, c_n ordered by left to right. We have already discussed that:

$$\begin{aligned}C(T) &\le \max(C(c_1) + 3 \times (n-1), C(c_2) + 3 \times (n-2), \ldots, C(c_n))\\ &\le \max(B(c_1) + 3 \times (n-1), B(c_2) + 3 \times (n-2), \ldots, B(c_n)) \equiv B(v)\\ &\le \max(C(c_1), \ldots, C(c_n)) + 3 \times (n-1)\end{aligned}$$

The last line tells us how to compute $B(v)$ recursively. Similarly one can construct a lower bound $b(l)$ assigning $b(l) = 0$ for every leaf and with the recursive formula:

$$b(v) \equiv \max(b(c_1) + 2 \times (n-1), b(c_2) + 2 \times (n-2), \ldots, b(c_n))$$

3.3 Analytical bounds



To find an analytical bounds suppose we have computed the values of $B(v)$ for every vertex recursively. Now, when computing $B(h)$ where h is the head of the tree there will be a children of the head $c(h)_{i_1}$ with $i_1 = 1, \ldots n(h)$ that will have given us the maximum value in the recursive formula:

$$B(h) = B(c(h)_{i_1}) + 3 \times (n(h = v_{i_0}) - i_1) \tag{9}$$

where $n(h)$ is the number of children of the head vertex and d is the depth of the tree. So we can construct this sequence of vertices

$$v_{i_0} = h, c(h)_{i_1}, v_{i_2}, \ldots, v_{i_{d-2}}, v_{i_{d-1}} = l$$

going from the head to a leaf where l. Here $i_1, i_2, \ldots$ represent the positioning of the vertex in the children list. Clearly vertex v_{i_k} is a children of $v_{i_{k-1}}$ and $i_k = 1, \ldots, n(v_{i_{k-1}})$. The upper bound for the head will thus be:

$$\begin{aligned} B(h) &= 3 + 3 \times (n(v_{i_{d-2}}) - i_{d-1}) + \cdots + 3 \times (n(v_{i_0}) - i_1) \\ &\leq 3 + 3 \left[(n(v_{i_{d-2}}) - 1) + \cdots + (n(v_{i_0}) - 1) \right] \\ &= 3 + 3 \left[n(v_{i_{d-2}}) + \cdots + n(v_{i_0}) - (d-1) \right] \end{aligned}$$

In particular, given a path starting from the head and going to a leaf:

$$p_0 = h, p_1, \ldots, p_{d-2}, p_{d-1} = l$$

we have

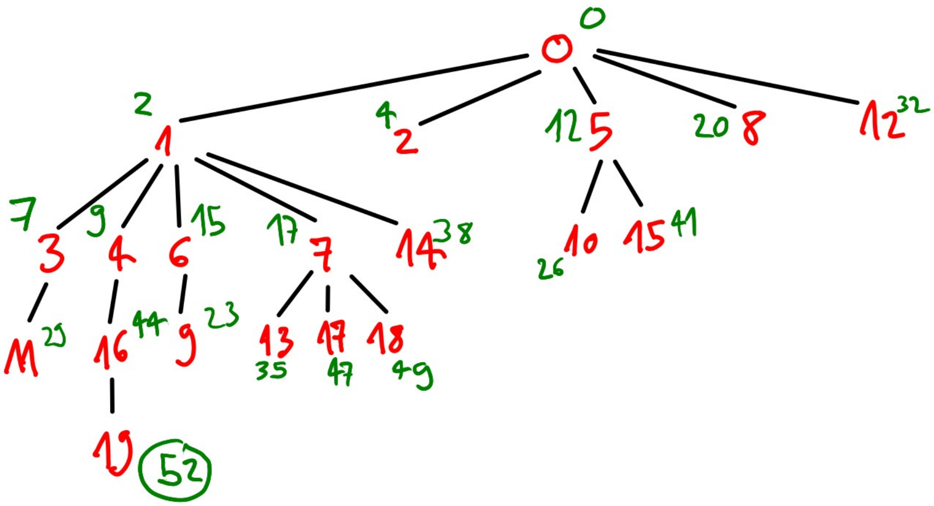
$$\begin{aligned} B(h) &\leq 3 + \max_{\text{all paths}} [n(p_{d-2}) + \cdots + n(p_0) - (d-1)] \qquad (10) \\ &\leq 3 + (N-1)(d-1) \qquad (11) \end{aligned}$$

where N is the maximum of $n(v)$ over all vertices in the tree.

One could also think if it's possible to build an analytical lower bound. Using $i_k \leq n(v_{i_{k-1}})$ one would trivially get $b(h) \geq 0$. This makes sense because a linear cluster takes only has $C(T) = 2$.

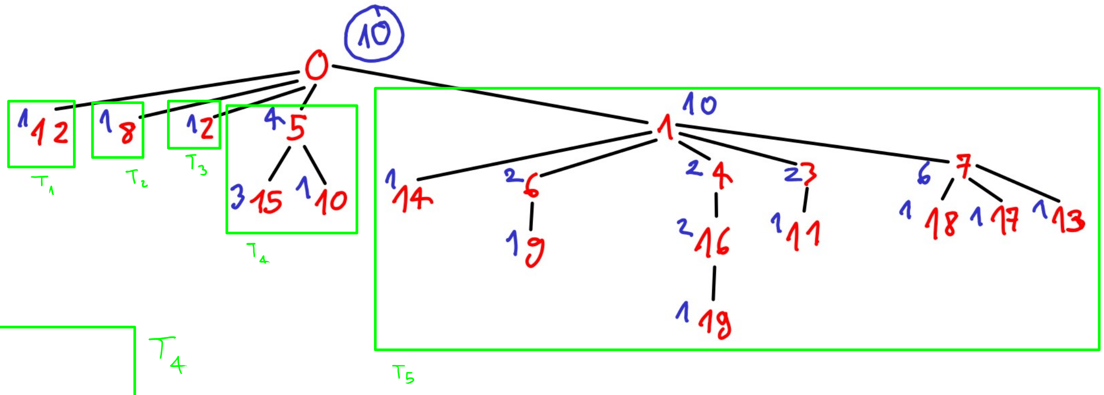
A trivial upper bound valid for every order O is $C(T) \leq 3V$.

RANDOM ORDER



```
0     1
0     1
2     1
0     2
1     0     2
3     4     1
1     3
0     2     1     3
6     6     7     1
1     4
0     2     3     1     4
6     6     8     9     1
0     5
2     3     1     4     0     5
8     10    11    11    12    1
1     6
2     3     4     0     5     1     6
8     10    13    14    14    15    1
1     7
2     3     4     0     5     6     1     7
8     10    13    14    14    16    17    1
0     8
2     3     4     5     6     1     7     0     8
8     10    13    16    18    19    19    20    1
6     9
2     3     4     5     1     7     0     8     6     9
8     10    13    16    21    21    22    22    23    1
5     10
2     3     4     1     7     0     8     6     9     5     10
8     10    13    23    23    24    24    25    25    26    1
3     11
2     4     1     7     0     8     6     9     5     10    3     11
8     15    25    25    26    26    27    27    28    28    29    1
0     12
2     4     1     7     8     6     9     5     10    3     11    0     12
8     15    25    25    28    29    29    30    30    31    31    32    1
7     13
2     4     1     8     6     9     5     10    3     11    0     12    7     13
8     15    25    30    31    31    32    32    33    33    34    34    35    1
1     14
2     4     8     6     9     5     10    3     11    0     12    7     13    1     14
8     15    32    33    33    34    34    35    35    36    36    37    37    38    1
5     15
2     4     8     6     9     10    3     11    0     12    7     13    1     14    5     15
8     15    32    33    33    36    37    37    38    38    39    39    40    40    41    1
4     16
2     8     6     9     10    3     11    0     12    7     13    1     14    5     15    4     16
8     34    35    35    38    39    39    40    40    41    41    42    42    43    43    44    1
7     17
2     8     6     9     10    3     11    0     12    13    1     14    5     15    4     16    7     17
8     34    35    35    38    39    39    40    40    43    44    44    45    45    46    46    47    1
7     18
2     8     6     9     10    3     11    0     12    13    1     14    5     15    4     16    17    7     18
8     34    35    35    38    39    39    40    40    43    44    44    45    45    46    46    48    49    1
16    19
2     8     6     9     10    3     11    0     12    13    1     14    5     15    4     17    7     18    16    19
8     34    35    35    38    39    39    40    40    43    44    44    45    45    46    50    51    51    52    1
```

BEST DFS



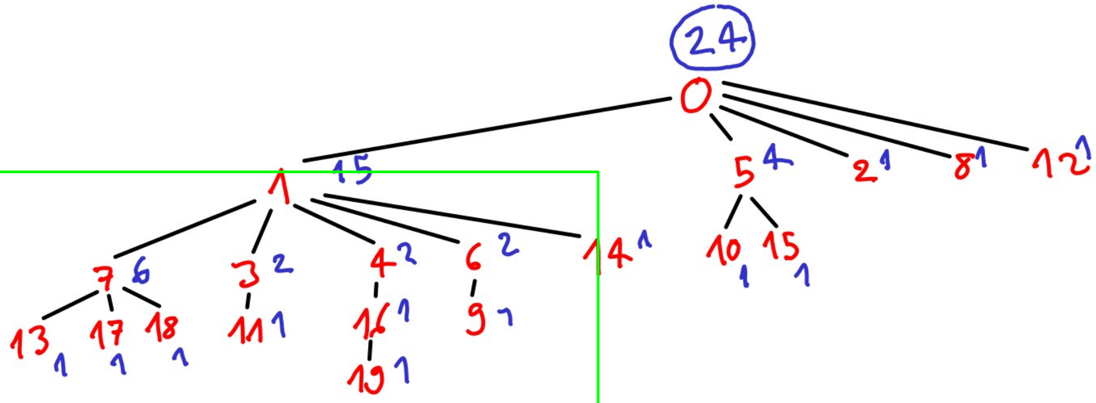
```
0     12
0      12
2      1
0     8
12     0      8
3      4      1
0     2
12     8      0      2
3      5      6      1
0     5
12     8      2      0      5
3      5      7      8      1
5     15
12     8      2      0      5      15
3      5      7      8      2      1
5     10
12     8     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

T4

max (8,4) + 3

max(10,10)

WORST DFS



```
0      1
0      1
2      1
1      7
0      1      7
2      2      1
7      13
0      1      7      13
2      2      2      1
7      17
0      1      13     7      17
2      2      3      4      1
7      18
0      1      13     17     7      18
2      2      3      5      6      1
1      3
0      13     17     7      18     1      3
2      5      7      8      8      9      1
3      11
0      13     17     7      18     1      3      11
2      5      7      8      8      9      2      1
1      4
0      13     17     7      18     3      11     1      4
2      5      7      8      8      10     10     11     1
4      16
0      13     17     7      18     3      11     1      4      16
2      5      7      8      8      10     10     11     2      1
16     19
0      13     17     7      18     3      11     1      4      16     19
2      5      7      8      8      10     10     11     2      2      1
1      6
0      13     17     7      18     3      11     4      16     19     1      6
2      5      7      8      8      10     10     12     12     12     13     1
6      9
0      13     17     7      18     3      11     4      16     19     1      6      9
2      5      7      8      8      10     10     12     12     12     13     2      1
1      14
0      13     17     7      18     3      11     4      16     19     6      9      1      14
2      5      7      8      8      10     10     12     12     12     14     14     15     1
0      5
13     17     7      18     3      11     4      16     19     6      9      1      14     0      5
7      9      10     10     12     12     14     14     14     16     16     17     17     18     1
5      10
13     17     7      18     3      11     4      16     19     6      9      1      14     0      5      10
7      9      10     10     12     12     14     14     14     16     16     17     17     18     2      1
5      15
13     17     7      18     3      11     4      16     19     6      9      1      14     0      10     5      15
7      9      10     10     12     12     14     14     14     16     16     17     17     18     3      4      1
0      2
13     17     7      18     3      11     4      16     19     6      9      1      14     10     5      15     0      2
7      9      10     10     12     12     14     14     14     16     16     17     17     19     19     19     20     1
0      8
13     17     7      18     3      11     4      16     19     6      9      1      14     10     5      15     2      0      8
7      9      10     10     12     12     14     14     14     16     16     17     17     19     19     19     21     22     1
0      12
13     17     7      18     3      11     4      16     19     6      9      1      14     10     5      15     2      8      0      12
7      9      10     10     12     12     14     14     14     16     16     17     17     19     19     19     21     23     24     1
```

$\leq \max(2,15)+3$

$\leq \max(18,4)+3$