Paper without a name

A. Auvolat A. Fromherz N. Jeannerod

June 01, 2014

Abstract

In this contribution, we investigate the problem of "SwitchBoxes". The goal is, given n wires, to generate all the permutations of these wires by using "boxes", that swap two wires. We tried to minimize the number of boxes that were necessary. We introduce a conjecture that puts our problem in relation with binary insertion sort. We successfully used a heavily optimized algorithm to prove this conjecture for small values of n.

1 Introduction

The SwitchBoxes problem is a combinatory problem. Given n wires, (coming from the top), we try to generate all the permutations of these wires by using boxes. A box takes two wires, and is allowed to swap these wires. We can concatenate these boxes to get a configuration of boxes. The configuration is said valid if, by swapping or not the wires on every box, we can obtain any permutation of the n wires.

Mathematically, we consider that the wires are numbers from 1 to n. A box is a permutation $\tau(\epsilon) = (i, j)^\epsilon$, where ϵ is equal to 0 or 1. A configuration of boxes C is the concatenation of k boxes $\tau_1, \ldots, \tau_k$, that means, $C(\epsilon_1, \ldots, \epsilon_k) = \tau_1^{\epsilon_1} \circ \ldots \circ \tau_k^{\epsilon_k}$. The configuration C is said valid if $C(\{0,1\}^k) = \mathfrak{S}_n$.

Given this problem, we tried to determine the best lower bound of the number of boxes k.

We can easily determine trivial lower and upper bounds of the number of boxes :
Given the fact that C is an application, we need to have $Card(\{0;1\}^k) \geq Card(\mathfrak{S}_n)$. We immediateley obtain that k has to be greater than $\log_2(n!)$.
We also have an upper bound : The traditionnal bubblesort is a valid configuration of the SwitchBoxes problem. We obtain that the optimal k is smaller than $\frac{(n-1)(n-2)}{2}$.

In this contribution, we first recall how the sequence of the maximal number of comparisons for sorting n elements by binary insertion is defined and develop how the optimal k seems to be linked to it. We then explain why we guess that those numbers are equal.

In a second step, we show how we proved this conjecture for n from 2 to 6, and how we demonstrated that the number of comparisons is an upper bound of the optimal number of boxes for n from 7 to 13 by optimizing the checking algorithm and by constructing a precise configuration.

In addition, we give some ideas we tried to develop in order to prove mathematically this conjecture.

In the rest of this paper, opt_n will be the optimal number of boxes for n wires.

2 The A001855 sequence

In this paragraph, we will develop on the A001855 sequence, here denoted u_n, which seems to be in relation with our problem.

u_n is defined as the maximal number of comparisons done by a binary insertion sort running on a table of n items.
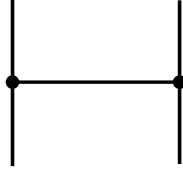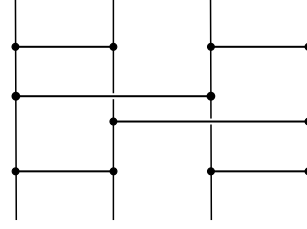
Figure 1: Trivial solution for 2 wires

Figure 2: Simple non-optimal solution for 4 wires

```
function FindPos(n, T, begin, end)    // find n in T(begin, begin+1, ..., end-1)
    assert(begin < end)
    if begin + 1 = end then
        return begin
    else
        mid = E[(begin + end) / 2]
        if T[mid] > n then
            return FindPos(n, T, begin, mid)
        else
            return FindPos(n, T, mid+1, end)

function Sort(T)
    n = T.size
    S = [T[0]]
    for i = 1 to n-1
        pos = FindPos(T[i], S, 0, S.size)
        insert T[i] in S at position pos
    return S
```

This algorithm progressively builds S, the table of the elements of T in increasing order. This is done by progressively inserting the elements of T in S at a position calculated by the $FindPos$ function. The $FindPos$ function is based on a binary search, therefore does $O(log(S.size))$ comparisons.

The u_n suite is defined as follows :

$$u_1 = 0$$

$$\forall n \in \mathbb{N}^*, u_{n+1} = u_n + \lceil log_2(n) \rceil$$

This sequence is described on the OEIS[1], and is referenced under the name A001855.

3   Our conjecture

To understand the problem, we first tried manually to get opt_n for very small values of n. For $n = 2$, the answer is quite simple : One box is enough and necessary to swap the two wires.

For $n = 3$, two boxes aren't enough. Using a permutation, we can consider that the first box swap the wires 1 and 2, and that the second box swap the wires 2 and 3. Then the permutation $(1, 3)$ can't be generated with this configuration. Hence, we have $opt_3 = 3$.

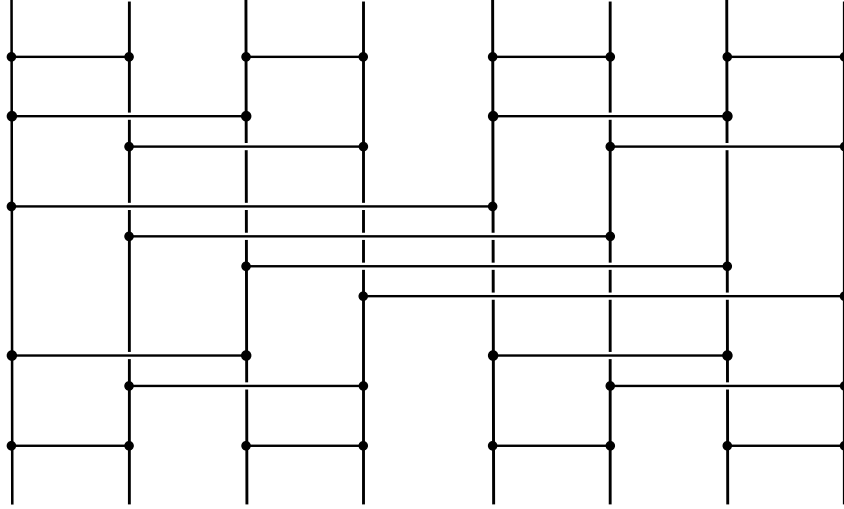[1]Online Encyclopedia of Integer Sequence

Figure 3: Simple non-optimal solution for 8 wires

Number of wires n	opt_n
2	1
3	3
4	5
5	8
6	11

Figure 4: Minimum box count for small number of wires

On these representations, a box is a horizontal line that can swap the wires the points are on.

For greater values of n, we implemented a naive and brutal algorithm that for k given, creates all the possible configurations of k boxes, and checks if one of these configurations is valid. We first try it with $opt_{n-1} + 1$. Then, we increment k and try again until we get a valid configuration. With this algorithm, we easily get opt_n.

However, this only works for small values of n. For n greater than 6, we lack memory and time to achieve the computation.

Using this algorithm, we got the results in the table of figure 4.

According to the fact that this sequence is exactly the beginning of the u_n sequence described in the previous paragraph, our conjecture is that the sequences are equal.

Conjecture. *opt_n is equal to the maximal number of comparisons for sorting n elements by binary insertion.*

In order to prove this conjecture for greater values, we tried to optimize our checking algorithm. That will be described in the next part.

3.1 First examples

$n = 2 : S_2(e_1) = (1, 2)^{e_1}$
$n = 3 : S_3(e_1, e_2, e_3) = (1, 2)^{e_1} \circ (2, 3)^{e_2} \circ (1, 2)^{e_3}$
$n = 4 : S_4(e_1, e_2, e_3, e_4, e_5) = (1, 2)^{e_1} \circ (3, 4)^{e_2} \circ (1, 3)^{e_3} \circ (2, 4)^{e_4} \circ (1, 2)^{e_5}$

3.2 Notation

We abbreviate the previous notation in the following way :

3

$$S_2 = (1,2)$$

$$S_3 = (1,2)(2,3)(1,2)$$

$$S_4 = (1,2)(3,4)(1,3)(2,4)(1,2)$$

4 Hypothesis box system

We have found a box system that seems to generate all permutations for all n. We have managed to prove that this box system is valid for small values of n ($n \leq 13$), and have found that the box systems were still valid when some boxes were removed.

4.1 The basis box configuration

We first define the q-block at position i by :

$$B(q,i) = (q, q + 2^i)(q + 1, q + 1 + 2^i) \cdots (q + 2^i - 1)(q + 2 * 2^i - 1)$$

For example:

$$B(0,i) = (i, i + 1)$$
$$B(1,i) = (i, i + 2)(i + 1, i + 3)$$
$$B(2,i) = (i, i + 4)(i + 1, i + 5)(i + 2, i + 6)(i + 3, i + 7)$$

Then, for $n = 2^p$ we define for $q < p$ the (p,q)-line by :

$$L(p,q) = B(q,0)B(q,2^{q+1}) \cdots B(q, 2^p - 2^{q+1})$$

For example:

$$L(3,0) = (0,1)(2,3)(4,5)(6,7)$$
$$L(3,1) = (0,2)(1,3)(4,6)(5,7)$$
$$L(3,2) = (0,4)(1,5)(2,6)(3,7)$$

We now conjecture that the following configuration generates all permutations for $n = 2^p$:

$$S_{2^p}^0 = L(p,0)L(p,1)L(p,2) \cdots L(p,p-1)L(p,p-2) \cdots L(p,1)L(p,0)$$

For example :

$$
\begin{aligned}
S_8^0 = \quad & L(3,0)L(3,1)L(3,2)L(3,1)L(3,0) \\
= \quad & (0,1)(2,3)(4,5)(6,7) \\
& (0,2)(1,3)(4,6)(5,7) \\
& (0,4)(1,5)(2,6)(3,7) \\
& (0,2)(1,3)(4,6)(5,7) \\
& (0,1)(2,3)(4,5)(6,7)
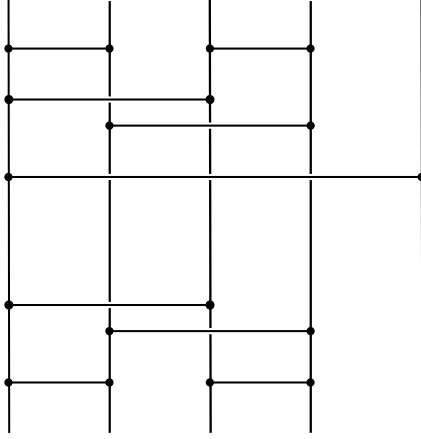\end{aligned}
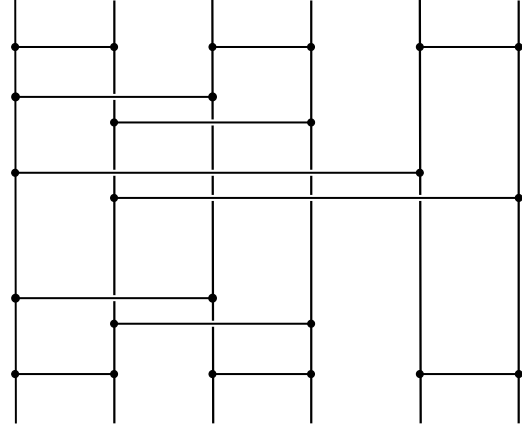$$

Figure 5: Solution for 5 wires

Figure 6: Solution for 6 wires

4.2 Truncation

For n which is not a power of two, we construct S_n^0 by taking $S_{2^{log_2(n)}}^0$ and keeping only boxes (i, j) having $i, j < n$.

For example, S_5^0 can be generated from S_8^0 and by removing quite a lot of the boxes. We obtain the following solution :

$$S_5^0 = (0, 1)(2, 3)(0, 2)(1, 3)(0, 4)(0, 2)(1, 3)(0, 1)(2, 3)$$

4.3 Testing the box system

For a box system $S = b_1 b_2 \ldots b_p$, we construct the sets :

$$Perm_0 = id$$

$$Perm_k = (b_1 b_2 \ldots b_k)(\{0, 1\}^k)$$

The permutation set $Perm_p$ therefore contains all the possible permutations we can generate with the p boxes. If $Perm_p = \mathfrak{S}_n$ then the box system is valid.

Each set $Perm_k$ is represented in memory as a bitset containing $n!$ bits, one for each permutation of $\mathfrak{S}_n$. The isomorphism between the permutations of $\mathfrak{S}_n$ and the numbers $\{0, 1, \ldots, n! - 1\}$ is constructed using the standard method described in (a citation).

The representation as a bitset is the most compact representation we have found.

4.4 Useless box elimination

We can easily detect that $Perm_{k+1} = Perm_k$ when $\#Perm_{k+1} = \#Perm_k$, because $Perm_k \subset Perm_{k+1}$. When this condition is fulfilled, then we know that the box b_{k+1} is useless because it adds no permutation to the set of generated permutations. We can therefore remove it from the box system.

By removing the boxes that this method detects as useless from the box system we constructed in the previous paragraphs, we have constructed box systems that generate all the permutations for $n \leq 13$ and that have u_n boxes (ie the conjectured optimum). We have not managed to run the test for $n \geq 14$, because the required RAM exceeds 20Go, which is not a resource we have in our possession.

5 Conclusion

All these results seem to confirm our intuition. To go further, we unsuccessfully tried to find applications from the set of trees representing a sort by binary insertion to the set of valid configurations of boxes that would be either one to one or onto, in order to find an inequality between these two sequences of integers. We also tried to put our construction of configurations in relation with binary insertion sort.