Solving recurrences

Denis TRYSTRAM

Lecture notes **Maths for Computer Science** – *MOSIG 1 – 2017*

1 Recall of the principle

Mathematical induction is used for proving that a statement *P(n)* involving integer n is true. The method is called *recurrence*.

- **Basis.** Solve the statement for the smallest values of n, usually P(0) or P(1).
- **Induction step.** Prove the statement for n assuming it is correct for $n-1$.

The previous definition is a simple induction, and can be extended to strong induction by extending the assumption for all $k < n$..

Induction is a mathematical concept which is strongly linked with *recursion* in computer science. Here, the solution for a problem of size n is obtained by calling the same problem at lower ranks (using the same method).

2 A preliminary example

Let us start by studying a puzzle whose solution can be obtained by a recursive algorithm.

Consider n bi-color tokens (one face blue and one face red) numbered from 1 to n initially placed on their blue sides (as depicted in Figure 1). They are ranked one after the other. The puzzle consists in flipping one token at a time according to one of the following constraints:

- Flip the token in position 1, or
- Flip the token which is located right after the first blue token

The two following figures 2 and 3 (for respectively even and odd n) detail the first steps of the solution of the puzzle.

The analysis of both cases lead to some evidences: First, in the even case, the process starts by reverting the second token while in the odd case, we flip the first one. Then, the first token is flipped every double steps, the second one every four steps and so on. The rules are applied alternatively.

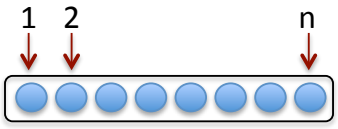

Figure 1: Initial position of the tokens (they are all on their blue side).

However, the solution is not easy to described and its cost is not easy to compute.

The solution can be easily expressed as a recursive algorithm. Indeed, for $n > 2$, if the $n-2$ first tokens are flipped, we can flit the last one, and again flip the first $n-2$, this way, we have the same position as the initial one except for the last token, which has be put in the right color. Then, the problem is obtained by simply flipping the remaining $n-1$ first tokens (as shown in Figure 4).

flipArray(n)
if $n = 1$ then flip(1)
If $n = 2$ then flip(2) and then flip(1)
if $n > 2$ then
flipArray(n-2) flip(n) flipArray(n-2)
filpArray(n-1)

2.1 Analysis

The analysis comes directly from the previous algorithm. Let call $f(i)$ the cost for flipping the whole array from position 1 to position i (the cost here refers to the number of elementary flipped token). The total cost is to flip the whole sequence, thus we have to solve:

$f(n) = f(n-1) + 2.f(n-2) + 1$ for $n > 2$
with $f(1) = 1$ and $f(2) = 2$.

If we add $f(n-1)$ in both sides of the previous expression, we obtain a much easier equation to solve (call $s(n)$ the sum $f(n)+f(n-1)$, in particular, $s(2) = 2 + 1$):

$f(n) + f(n-1) = 2.f(n-1) + 2.f(n-2) + 1$
$s(n) = 2.s(n-1) + 1 = 2.(2.s(n-2) + 1) + 1 = 2^2 s(n-2) + 2 + 1 = 2^3 s(n-3) + 2^2 + 2 + 1 = ... = 2^{n-2} s(2) + 2^{n-3} + ... + 2^2 + 2 + 1$ where $s(2) = 3$
$s(n) = 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^2 + 2 + 1$
Summing up the geometrical series leads to, $s(n) = 2^n - 1$
Coming back to the $f(n)$, we can rewrite this equation as:
$f(n) = 2^n - f(n-1) - 1$

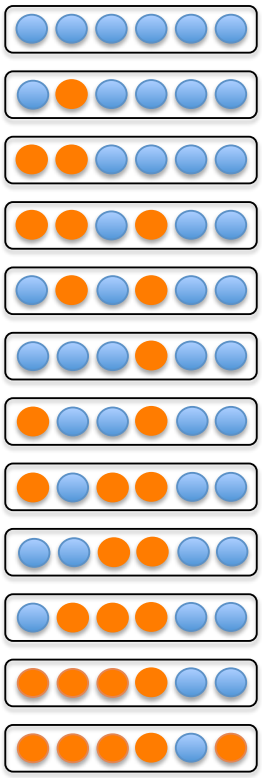

Figure 2: First steps for solving the even case ($n = 6$).

This sequence is an alternate sequence of powers of 2 where the 1 are cancelled two by two.

$f(n) = 2^n - 2^{n-1} + f(n-2) + 1 - 1 = 2^n - 2^{n-1} + 2^{n-2} - f(n-3) - 1 = ...$

We finally obtain:

$f(n) = \frac{1}{3}(2^{n+1} + 1)$ if n is even

$f(n) = \frac{1}{3}(2^{n+1} + 2)$ if n it is odd

3 Hanoi Towers

3.1 Basic version

This introductory problem comes from the french mathematician Edouard Lucas (1883) who was famous for his *récréations mathématiques*. He invented this problem, which belongs today to the folklore of computer science.

We are given a set of n disks of decreasing diameters (called a *tower*) initially stacked on one of three pegs. This is illustrated in Figure 5 where D, A and I denote respectively the Departure, Arrival and Intermediate pegs.

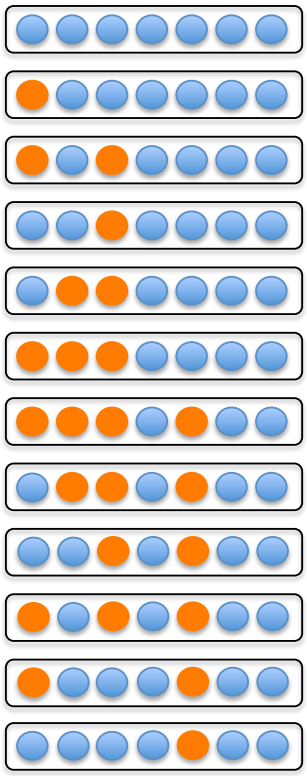

Figure 3: First steps for solving the odd case ($n = 7$).

Problem definition. The goal is to transfer the entire tower from a given peg to another fixed one, moving only one disk at a time and never moving a larger disk on top of a smaller one.

When n is odd, the arrival peg is the rightmost one, when it is even, it is the middle one.

The main question of this puzzle is to determine the *best* way to realize this operation (which means in a minimum number of moves). There exist several solutions, the most famous one is the recursive version. Let us first study a bound on the minimum number of moves.

3.2 Lower bound

Let call H_n the number of moves required to solve the puzzle with n disks. What is the minimum number of moves?

When there is only one disk, there is only one move: $H_1 = 1$, with two disks, it is easy to see that at least 3 moves are necessary: $H_2 = 3$.

Let us prove $H_n \geq 2H_{n-1} + 1$. Indeed, looking at the largest disk, the $n-1$ others must be on a single peg, which required H_{n-1} to put them here.

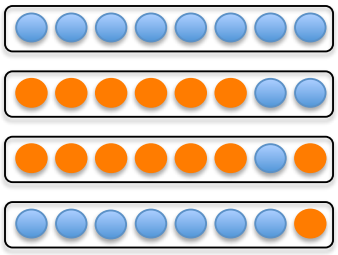

Figure 4: Principle of the recursive solution.

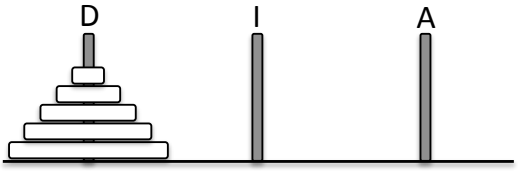

Figure 5: Initial position of the Hanoi tower composed of 5 disks.

Then, the largest disk should be moved, and again move the $n-1$ others on the target peg.

Thus, we have the following recurrence equation to solve: $H_n = 2H_{n-1}+1$ $(n \geq 2)$ with $H_1 = 1$.

There exists a closed formula, which is obtained as follows.

The first ranks give us an insight of the solution (1, 3, 7, 15, 31, ...). We guess $H_n = 2^n - 1$ for $n \geq 1$.

- **Basis** is straightforward since $H_1 = 2^1 - 1 = 1$.
- **Induction step**

  $H_n = 2H_{n-1} + 1$ where $H_{n-1} = 2^{n-1} - 1$,

  thus, $H_n = 2(2^{n-1} - 1) + 1 = 2^n - 1$ and we are done.

Notice that this expression can also be obtained directly as the sum of a geometric series:

$H_n = 2H_{n-1} + 1$
$= 2(2H_{n-2} + 1) + 1$
$= \sum_{j=0}^{n-1} 2^j$
$= \frac{1-2^n}{1-2} = 2^n - 1$

3.3 The classical (recursive) solution

The natural method is recursive. It consists in moving the $n-1$ top disks on the intermediate peg, then, put the largest one on the target peg, and moving again the $n-1$ disks on top of it. Figure 7 illustrates this principle.

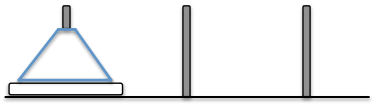

Figure 6: Principle of the recursive solution. Initial position

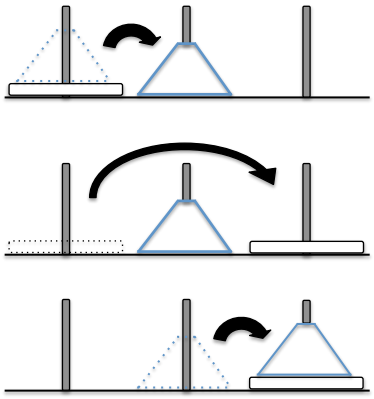

Figure 7: Principle of the recursive solution.

We give below the corresponding formal algorithm.

Hanoi
input: an integer n, three pegs indexed as D (departure), A (arrival) and I (intermediate). All the disks are stacked on peg D.
output: The disks are stacked on peg A.
If (n $\neq$ 0) then Hanoi(n-1,D,I,A)
move disk from D to A
Hanoi(n-1,I,A,D)

3.4 Analysis

It is easy to compute the number of moves, using the same recurrence equation as for the previous lower bound $H_n = 2^n - 1$. It is optimal with 3 pegs (since it corresponds to the lower bound).

3.5 Iterative solution

A simple solution can be obtained by remarking that in the recursive solution, the smallest disk moves every second step. Then, there is no choice for the other moves (according to the constraint of moving only a disk on top of a larger one, there is only one possible move). Thus, the problems are: determine on which peg to move the smallest disks and characterize the following move. A nice coding allows to write an elegant algorithm (see chapter on coding).

4 Extension to more pegs

Let us study the problem for a larger number of pegs (denoted by k). Let Hanoi(n,k) denotes this problem. The problem studied in the previous section corresponds to Hanoi(n,3).

Now we are interested in the problem where the number of pegs is not fixed, in particular, we are interested in determining what is the minimum number of moves that can be achieved if this number is as large as needed.

4.1 Linear time with a linear number of pegs

It is clear that for $k = n + 1$, the number of move is linear in n (just move each disk on a different peg (which requires $n - 1$ moves), move the largest one (1 move) and put them one after the other on top of the largest one ($n - 1$ moves), thus, a total of $2n - 1$ moves.

Actually, it is the absolute lower bound since each disk (except the largest one) should be moved at least twice.

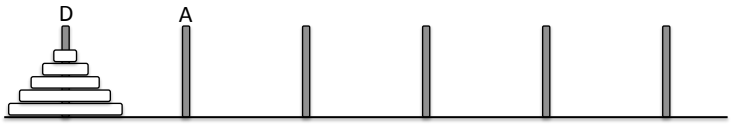

Figure 8: Improved solution in optimal number of moves with $n + 1$ pegs (n=5 and k=6). Initial position.

4.2 An improved solution

An interesting question is if the previous bound can be achieved for less pegs than $n + 1$. The answer is "no" but there is asymptotically a solution with less pegs that keeps a linear number of moves (in n). More precisely, we present as follows a solution with $\mathcal{O}(\sqrt{n})$ pegs with $\mathcal{O}(n)$ number of moves.

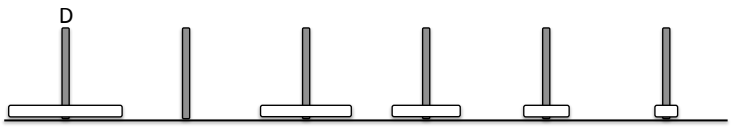

Figure 9: First $n-1$ moves with a linear distribution.

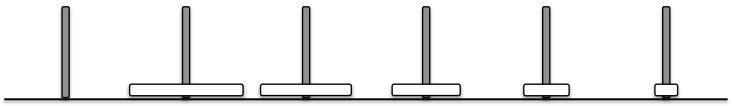

Figure 10: Move the largest disk.

This is obtained by dividing the n disks into groups of size $\sqrt{n}$ and consider $k = 2\sqrt{n}$ pegs (for the sake of clarity, we assume here that n is a perfect square).

The analysis follows immediately: There are $\sqrt{n}$ blocks composed each of $\sqrt{n}$ disks. These groups are moved using the linear pattern presented in the previous section ($2\sqrt{n}-1$ moves of disks). As the group of the largest disks is on place, it remains to move the $\sqrt{n}-1$ ones using again the linear pattern in $2\sqrt{n}-1$ moves. Thus, a total of $(2\sqrt{n}-1)(2\sqrt{n}-1) = 4n-4\sqrt{n}+1$. The multiplicative constant becomes 4 instead of 2 but there is a second order term in $\sqrt{n}$.

Notice to conclude this section that the pegs are not used very efficiently, this suggest further improvements. In particular, the solution of the puzzle for $n = 9$ and $k = 5$ is still linear using a decomposition in three groups of sizes 4, 3 and 2. This leads to a number of moves equal to: $7+5+3+5+7 = 27$, which is to be compared to $4 \times 9 - 4 \times 3 + 1 = 25$ with 6 pegs.

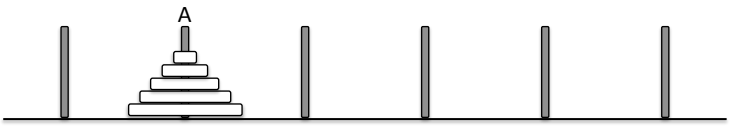

Figure 11: $n-1$ last moves (linear).

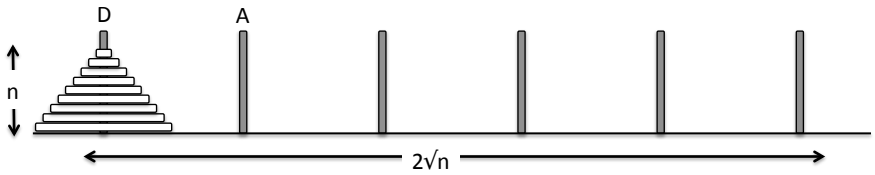

Figure 12: Improved solution for a tower of 9 disks in a linear number of moves with $2\sqrt{n} = 6$ pegs. Initial position.

5 Josephus' problem

The problem comes from an old story reported by Flavius Josephus during the Jewish-Roman war in the first century. The legend reports that Flavius was among a band of 41 rebels trapped in a cave by the roman army. Preferring suicide to capture, the rebels decided to form a circle and proceeding around to kill every third remaining person until no one was left. As Josephus had some skills in Maths and wanted none of this suicide non-sense, he quickly calculated where he should stand in the circle in order to stay alive at the end of the process.

Definition. Given n successive numbers in a circle. The problem is to determine the *survival number* (denoted by $J(n)$) in the process of removing every second remaining number starting from 1 (see figure 19).

In particular, we are going to determine if there exists a closed formula. Guessing the answer sounds not obvious. We need to better understand the progression.

Property 1. $J(n)$ is odd
Proof. This is straightforward since the first tour removes all even numbers! See figure 20.

We called *round*, the set of steps to come back at a given position in the circle. Starting at 1, the first round is completed after $\lceil \frac{n}{2} \rceil$ steps. Then, again half of the of the remaining numbers are removed in the second round and so on.

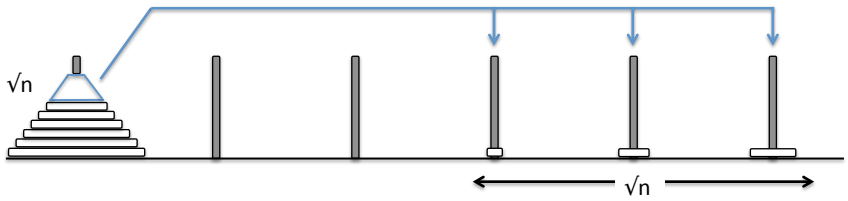

Figure 13: Move of the first block composed of 3 disks (linear).

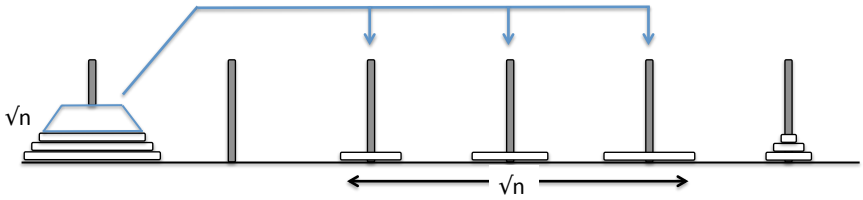

Figure 14: Move of the second block.

How many rounds do we have for determining $J(n)$? If N denotes this number, it verifies $\Sigma_{i=1}^{i=N} \frac{1}{2^i} = 1$.

Property 2. (even numbers) $J(2n) = 2J(n) - 1$
Proof. This is a simple generalization of the previous property. If n is even, the first round corresponds simply to come back to the original circle where one half of the points have been removed.

From this, we deduce $J(2^m) = 1$ for all m.
Let us turn to odd numbers.

Property 3. (odd numbers) $J(2n + 1) = 2J(n) + 1$

We can compute easily the first ranks. It turns out that the progression is composed of grouped terms starting at each power of 2. Let $n = 2^m + k$, the rule within each group m is to start at 1 and increase by 2 the successive numbers ($0 \leq k < 2^m$). Let prove it by recurrence on n.

Property 4. $J(2^m + k) = 2k + 1$
Proof.

- **Basis.** $n = 1$, thus $m = 0$, $k = 0$ and $J(1) = 2^0 + 0 = 1$
- **Induction step.** Suppose the formula holds for any integer lower than $n = 2^m + k$. Since there are two expressions for $J(.)$, we distinguish the cases whether k is even and k is odd:

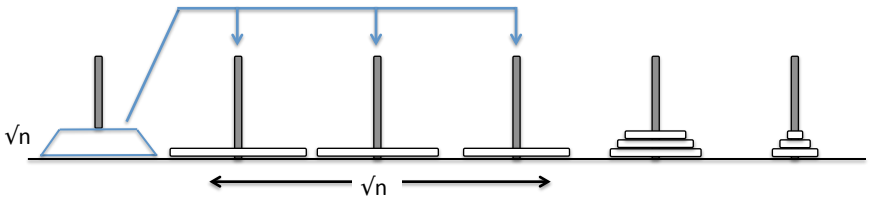

Figure 15: Move of the last block.

Figure 16: The 3 distributed blocks.

- If k is even, then, $2^m + k$ is even, and we can write:
  $J(2^m + k) = 2J(2^{m-1} + \frac{k}{2}) - 1$
  by induction hypothesis, $J(2^{m-1} + \frac{k}{2}) = 2\frac{k}{2} + 1 = k + 1$
  Thus, $J(2^m + k) = 2(k + 1) - 1 = 2k + 1$.
- If k is odd, the proof is similar:
  $J(2^m + k) = 2J(2^{m-1} + \lfloor \frac{k}{2} \rfloor) + 1 = 2\lfloor \frac{k}{2} \rfloor + 1 = 2k + 1$.

Figure 17: Final move of the second block.

Figure 18: Final move (last block).

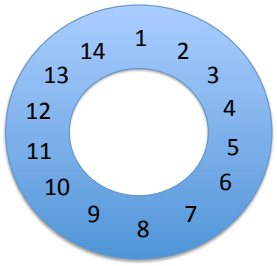

Figure 19: Initial situation for the Josephus process.

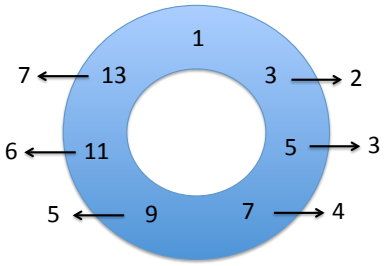

Figure 20: First step of the process (n is even).