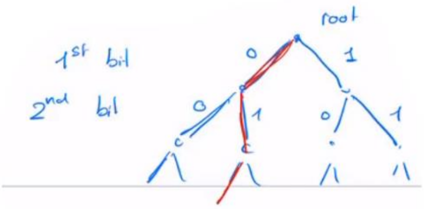
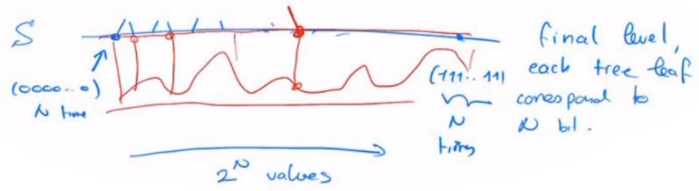
15- Ant algorithms: the simple version in {0,1}^N, and discussion of performance

What we are doing here is a benchmark of this algorithm!

- We will consider a simpler version of the ant algorithm
- We want to find a bit string which maximizes a given fitness function: S={0,1}^N
- This makes it so the search space is a tree structure, at each step (node) we have two options for the next bit, 0 or 1. This tree has length of N

Visually it looks like this:

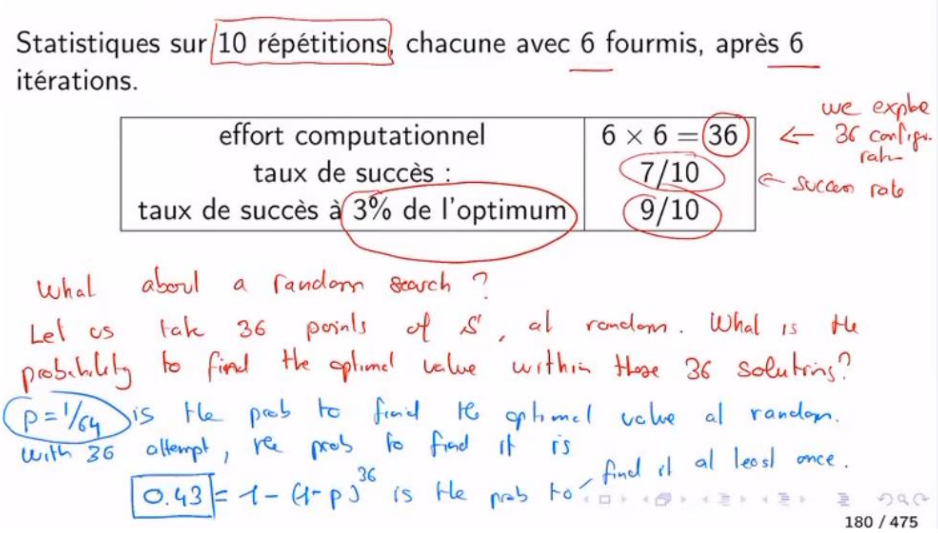




For every value of the search space -> 2^n, there is a fitness function that evaluates the path, then the deposited pheromone on this path is a function of the fitness function result

➔ At each bifurcation an ant will more likely choose the path that has the most pheromone, each path has the following probability of being chosen:

$$P_{left} = \frac{\tau_{left}}{(\tau_{left} + \tau_{right})} \qquad P_{right} = 1 - P_{left}.$$



Statistiques sur 10 répétitions, chacune avec 6 fourmis, après 6 itérations.

effort computationnel	$6 \times 6 = 36$	← we explore 36 configuration
taux de succès :	7/10	← success rate
taux de succès à 3% de l'optimum	9/10	

What about a random search?
Let us take 36 points of S, at random. What is the probability to find the optimal value within those 36 solutions?
p = 1/64 is the prob to find the optimal value at random.
With 36 attempt, the prob to find it is
0.43 = 1 - (1-p)^36 is the prob to find it at least once.

180 / 475