Complexité CM6

Antonio E. Porreca
aeporreca.org

Diapositives basées sur le livre Complexité algorithmique par Sylvain Perifel

Algorithmes non déterministes

Un exemple : cycle hamiltonien

fonction hamiltonien(V, E)

$n := |V|$

$perm$:= tableau(n)

pour $i := 0$ **à** $n - 1$ **faire**

$perm[i]$:= devine($0, \ldots, n - 1$)

pour chaque $v \in V$ **faire**

si $perm$ ne contient pas v exactement 1 fois **alors**

rejeter

pour $i := 0$ **à** $n - 1$ **faire**

si $(perm[i], perm[(i + 1) \bmod n]) \notin E$ **alors**

rejeter

accepter

fin

Un exemple : cycle hamiltonien

fonction hamiltonien(V, E)

$n := |V|$

$perm :=$ tableau(n)

pour $i := 0$ **à** $n - 1$ **faire**

$perm[i] :=$ devine($0, \ldots, n - 1$)

pour chaque $v \in V$ **faire**

si $perm$ ne contient pas v exactement 1 fois **alors**

rejeter

pour $i := 0$ **à** $n - 1$ **faire**

si $(perm[i], perm[(i + 1) \bmod n]) \notin E$ **alors**

rejeter

accepter

fin

$perm$ **est-elle une permutation ?**

Un exemple : cycle hamiltonien

fonction hamiltonien(V, E)
    $n := |V|$
    $perm :=$ tableau(n)
    pour $i := 0$ **à** $n - 1$ **faire**
        $perm[i] :=$ devine($0, \ldots, n - 1$)
    pour chaque $v \in V$ **faire**
        si $perm$ ne contient pas v exactement 1 fois **alors**
            rejeter
    pour $i := 0$ **à** $n - 1$ **faire**
        si $(perm[i], perm[(i + 1) \bmod n]) \notin E$ **alors**
            rejeter
    accepter
fin

***perm* est-elle une permutation ?**

***perm* est-il un cycle dans le graphe ?**

Un exemple : cycle hamiltonien

fonction hamiltonien(V, E)
$n := |V|$
$perm :=$ tableau(n)
pour $i := 0$ **à** $n - 1$ **faire**
$perm[i] :=$ devine($0, \ldots, n-1$)
pour chaque $v \in V$ **faire**
si $perm$ ne contient pas v exactement 1 fois **alors**
rejeter
pour $i := 0$ **à** $n - 1$ **faire**
si $(perm[i], perm[(i+1) \bmod n]) \notin E$ **alors**
rejeter
accepter
fin

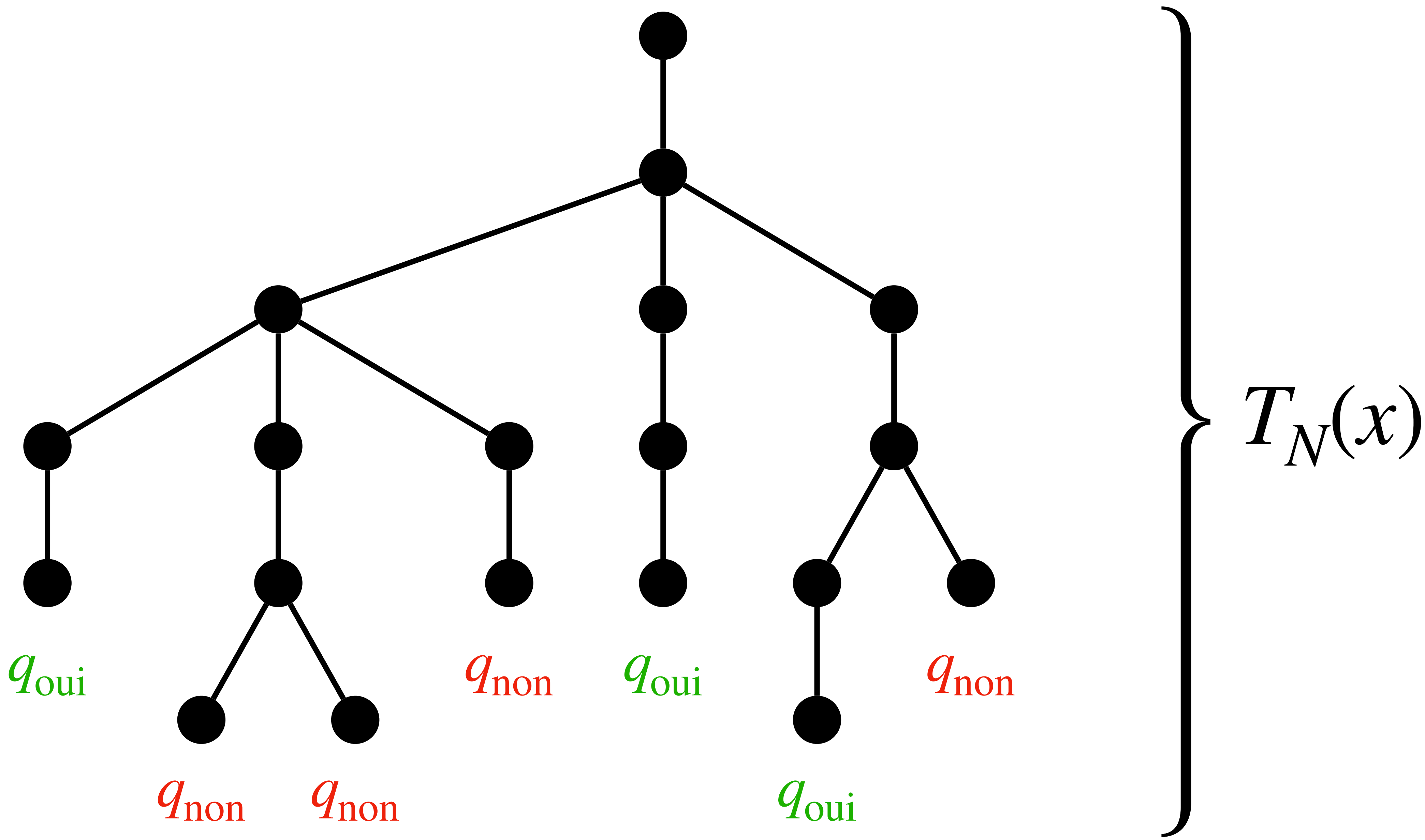
$O(n \log n)$ **bits devinés**

$perm$ **est-elle une permutation ?**

$perm$ **est-il un cycle dans le graphe ?**

Mesure du temps de calcul non déterministe

Temps de calcul d'une machine non déterministe N sur l'entrée x



Temps de calcul d'une machine non déterministe N

- Comme dans le cas déterministe, on prend le max des temps de calcul des entrées de taille n :

$$T_N(n) = \max\{T_N(x) : x \in \Sigma^\star \text{ et } |x| = n\}$$

- Donc la longueur du chemin de calcul le plus long des arbres de calcul des entrées de taille n

- Le temps de calcul est polynomial si $T_N(n) \in O(p(n))$ pour un polynôme p

La classe de complexité $\mathbf{NP}$

- C'est la classe de langages reconnus par des machines de Turing non déterministes en temps polynomial

$$\mathbf{NP} = \left\{ \begin{array}{c} L : \text{il existe une machine de Turing} \\ \text{non déterministe } N \text{ qui fonctionne en temps} \\ \text{polynomial telle que } L = L(N) \end{array} \right\}$$

- De façon équivalente, c'est aussi la classe de problèmes π sous le codage S tels que $L(\pi, S) \in \mathbf{NP}$

$\mathbf{P}$ vs $\mathbf{NP}$

- Comme on peut voir chaque machine déterministe comme machine non déterministe, on a automatiquement $\mathbf{P} \subseteq \mathbf{NP}$

- On sait qu'on peut simuler de façon déterministe chaque machine non déterministe, mais on ne sais pas si on peut le faire efficacement (en temps polynomial)

- Donc on ne sais pas si $\mathbf{NP} \subseteq \mathbf{P}$ et donc si $\mathbf{P} = \mathbf{NP}$

- C'est l'un des Millennium Prize Problems du Clay Mathematics Institute, et il y a un prix de 1000000 $ pour celle ou celui qui trouve la réponse !

La classe de complexité $\mathbf{coNP}$

- C'est la classe de langages dont le complément appartient à $\mathbf{NP}$:

$$\mathbf{coNP} = \{L : \text{co-}L \in \mathbf{NP}\}$$

- ⚠️ $\mathbf{coNP}$ n'est pas le complément de $\mathbf{NP}$, c'est-à-dire,
ce n'est pas la classe des problèmes qu'on ne peut pas résoudre en temps polynomial de façon non déterministe

- Notamment, on a $\mathbf{P} = \mathbf{coP}$, donc $\mathbf{P} \subseteq \mathbf{coNP}$, ce qui implique
$\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$: $\mathbf{NP}$ et $\mathbf{coNP}$ ne sont pas disjointes

- On ne sait pas si $\mathbf{NP} = \mathbf{coNP}$ non plus !

$\mathbf{NP} \neq \mathbf{coNP}$ vaut 1000000 \$

- Si $\mathbf{NP} \neq \mathbf{coNP}$ alors $\mathbf{P} \neq \mathbf{NP}$!

- On a vu que $\mathbf{P} = \mathbf{coP}$, donc $\mathbf{P} = \mathbf{NP}$ impliquerait $\mathbf{coNP} = \mathbf{coP} = \mathbf{P}$ et donc $\mathbf{NP} = \mathbf{coNP}$

- La proposition contraposée de $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{NP} = \mathbf{coNP}$ est $\mathbf{NP} \neq \mathbf{coNP} \Rightarrow \mathbf{P} \neq \mathbf{NP}$

- $\mathbf{NP}$ vs $\mathbf{coNP}$ ne semble pas du tout plus simple à résoudre que $\mathbf{P}$ vs $\mathbf{NP}$…

Divinations 🔮 et vérifications 🔍, ou le non déterminisme dans le monde réel

Proposition 2-AO (p. 56)

Caractérisation existentielle de $\mathbf{NP}$

Les deux conditions suivantes sont équivalentes :

- Le langage L est reconnu en temps polynomiale par une machine de Turing non déterministe N

- Il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que

$$x \in L \text{ ssi } \exists y \in \{0,1\}^{p(n)}\ M(x,y) \text{ accepte}$$

Le mot y qui justifient l'appartenance de x à L est appelé preuve ou certificat

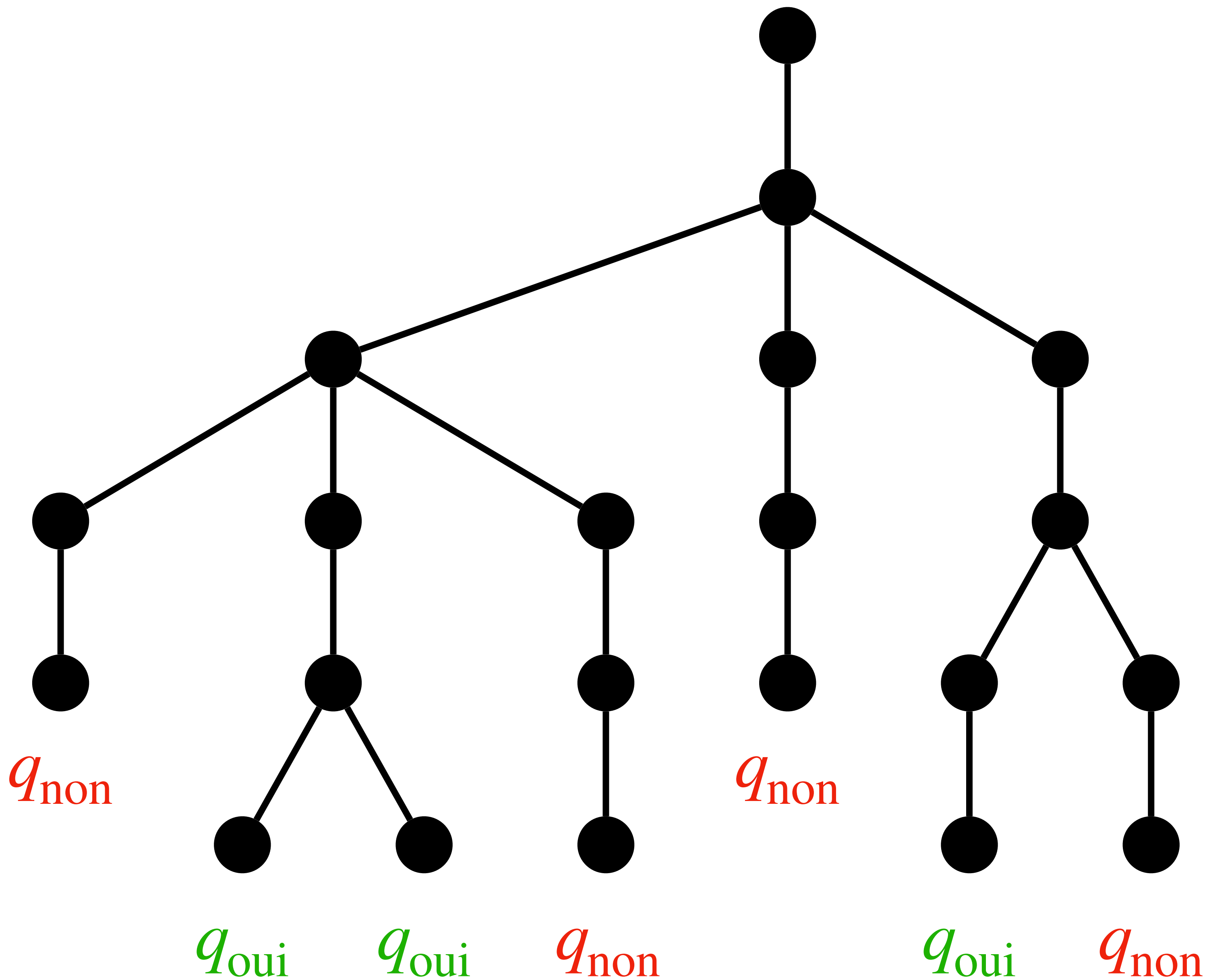
Idée de la démonstration

Le certificat y correspond au chemin acceptant
pour l'entrée x dans la machine non déterministe N
qui reconnait le langage L

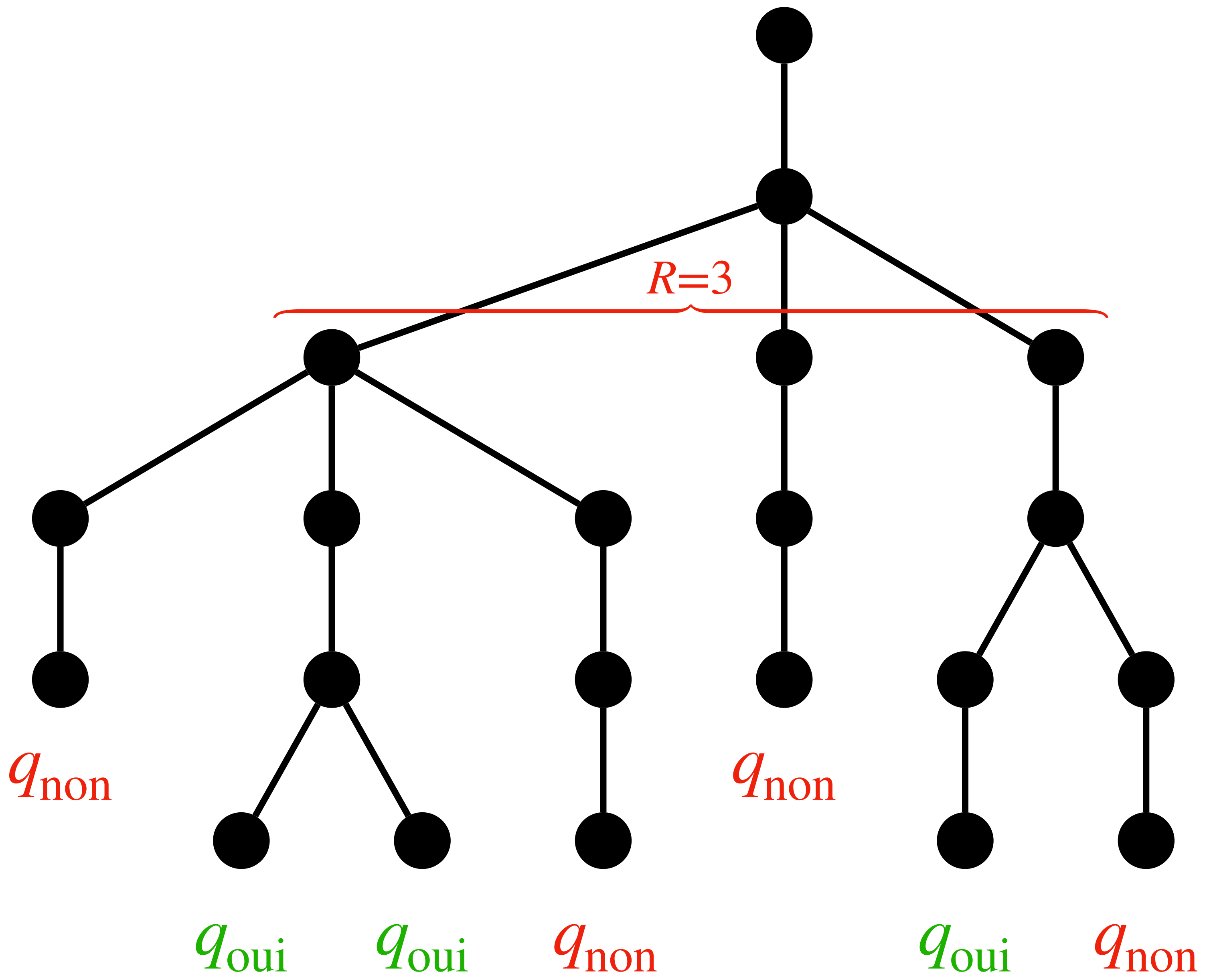
Demonstration

- Supposons que N reconnaît L en temps polynomial $q(n)$
- À chaque étape, N choisit entre un nombre $\leq R$ de transitions possibles (R est constant, il ne dépend pas de l'entrée x)
- Chaque choix peut être décrite avec $\lceil \log R \rceil$ bits

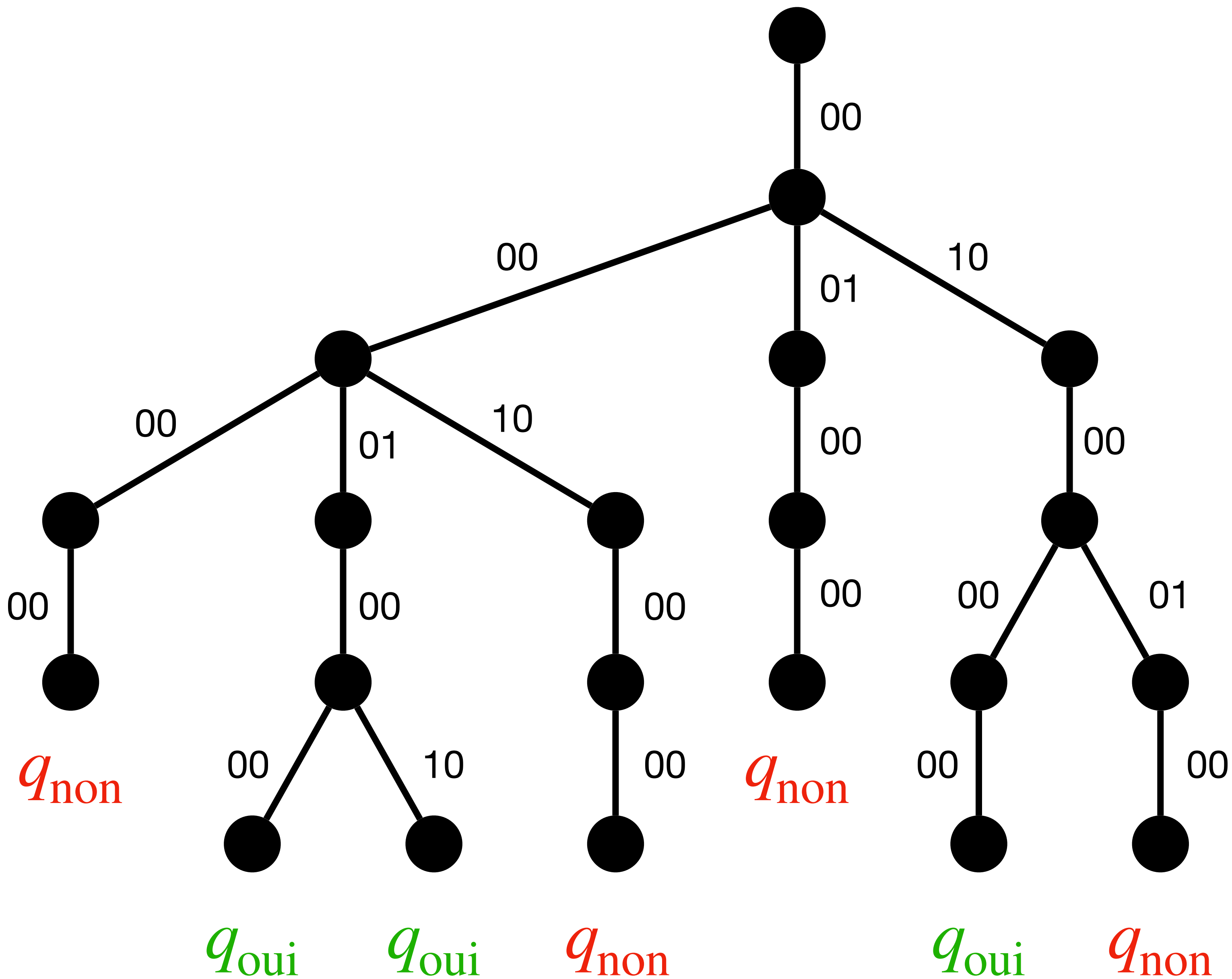
Arbre de calcul de N sur x



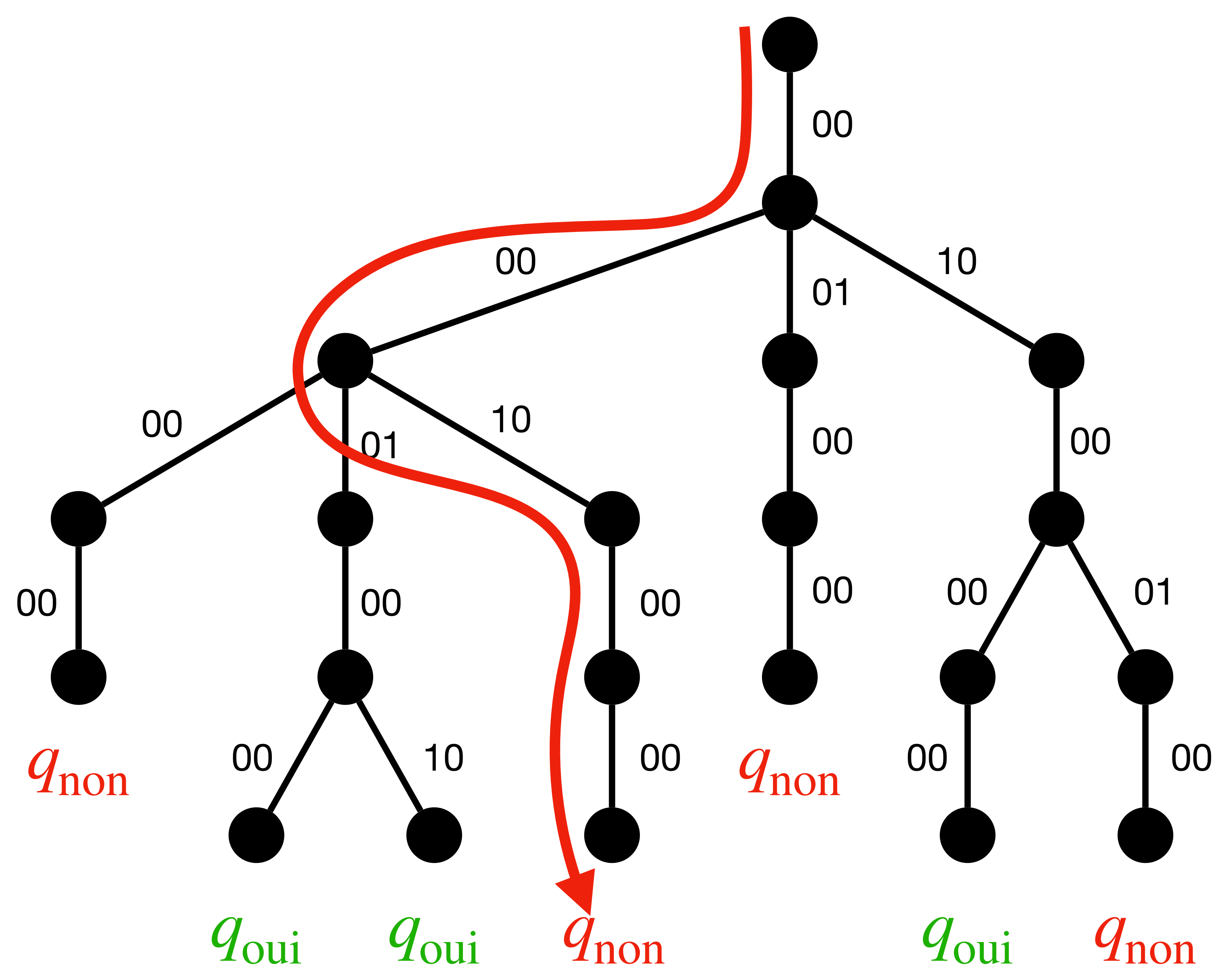
Arbre de calcul de N sur x



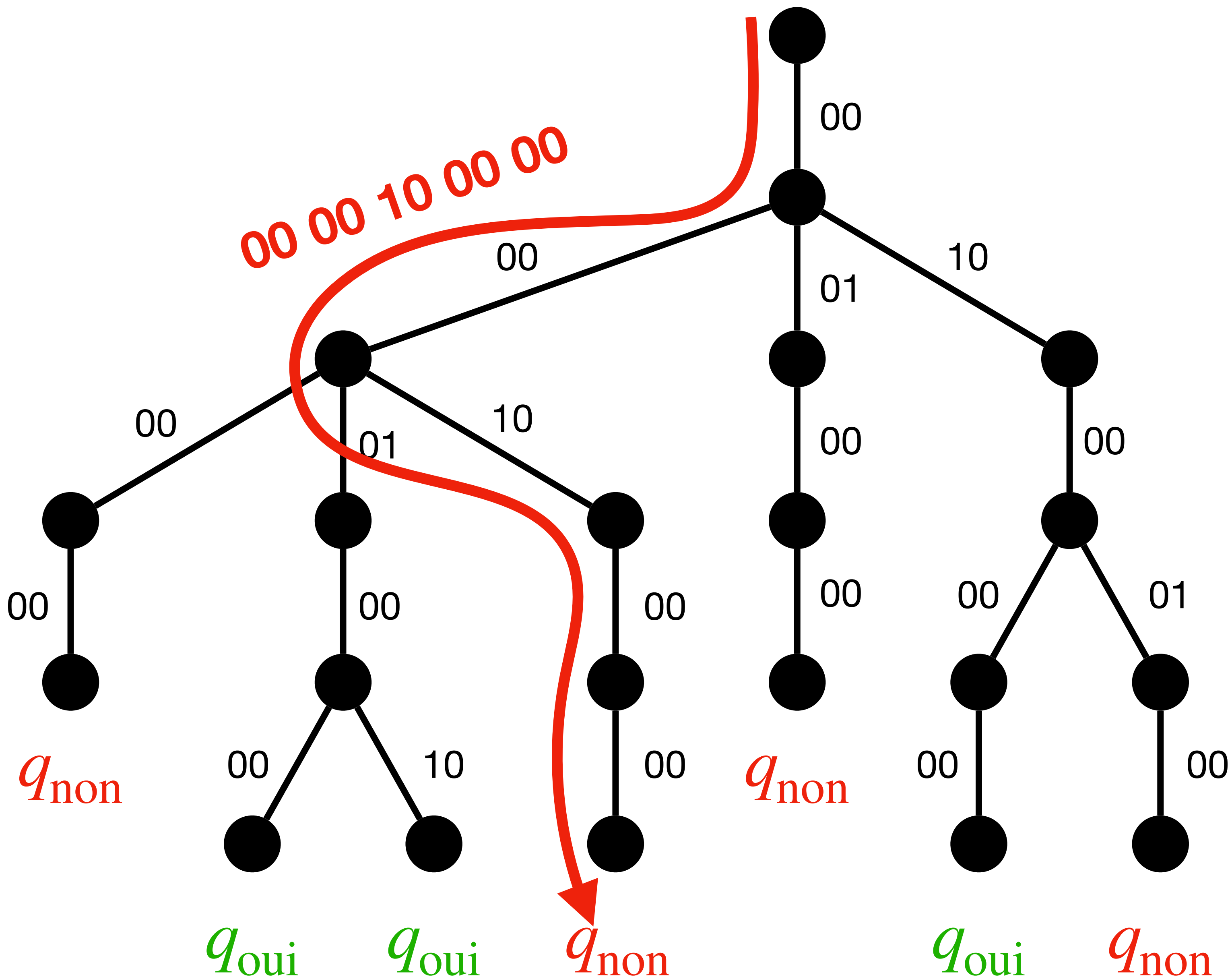
Arbre de calcul de N sur x



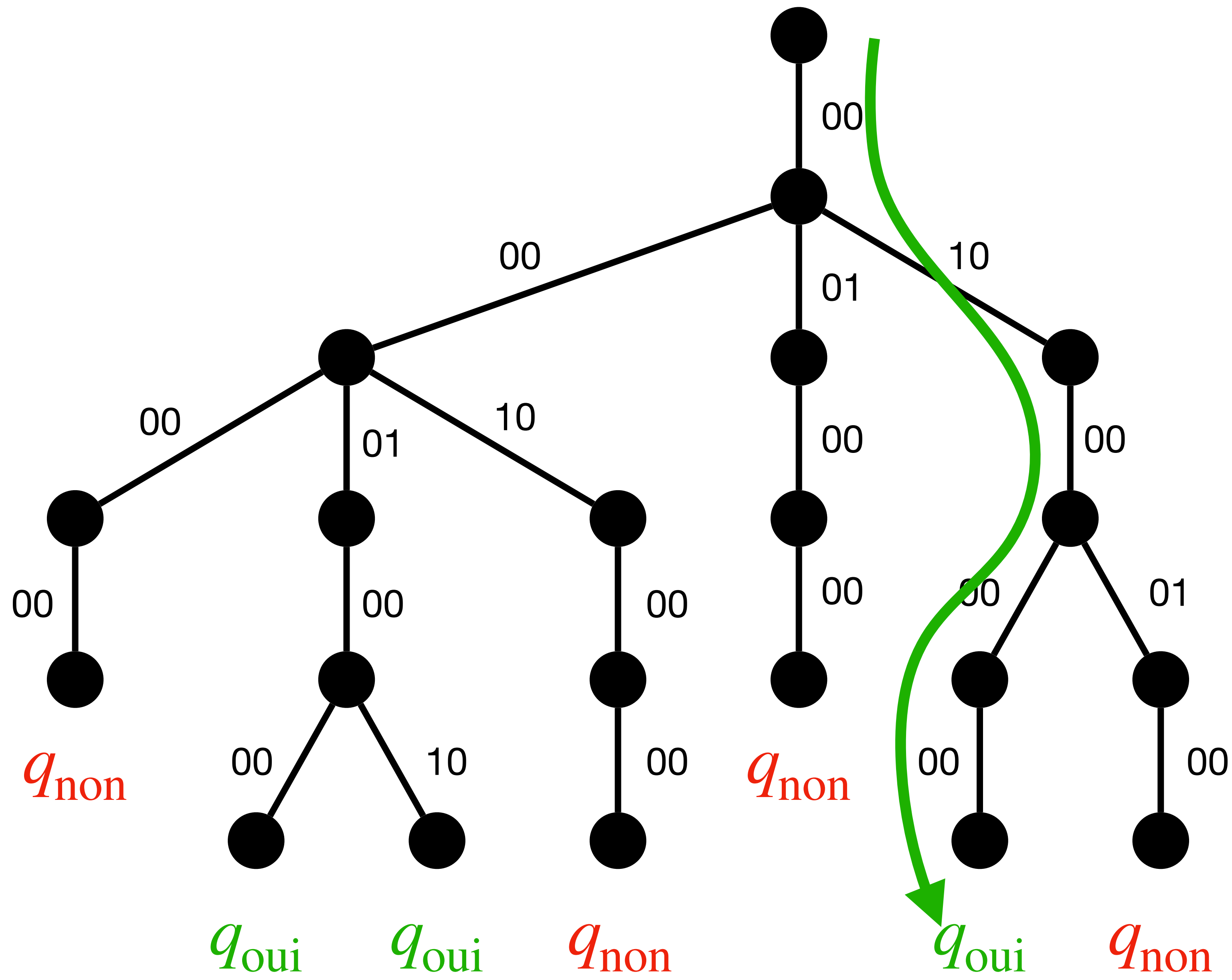
Arbre de calcul de N sur x



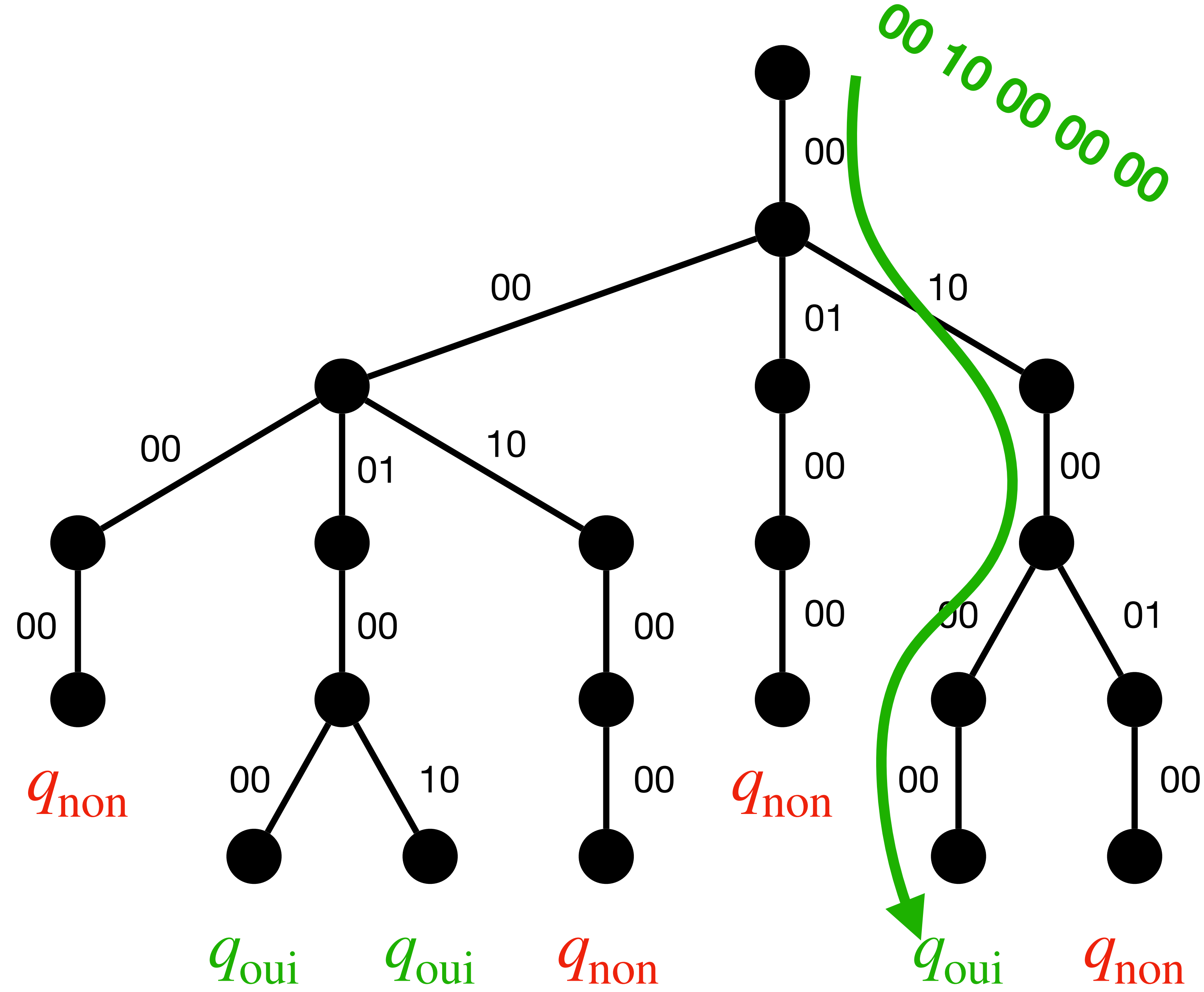
Arbre de calcul de N sur x



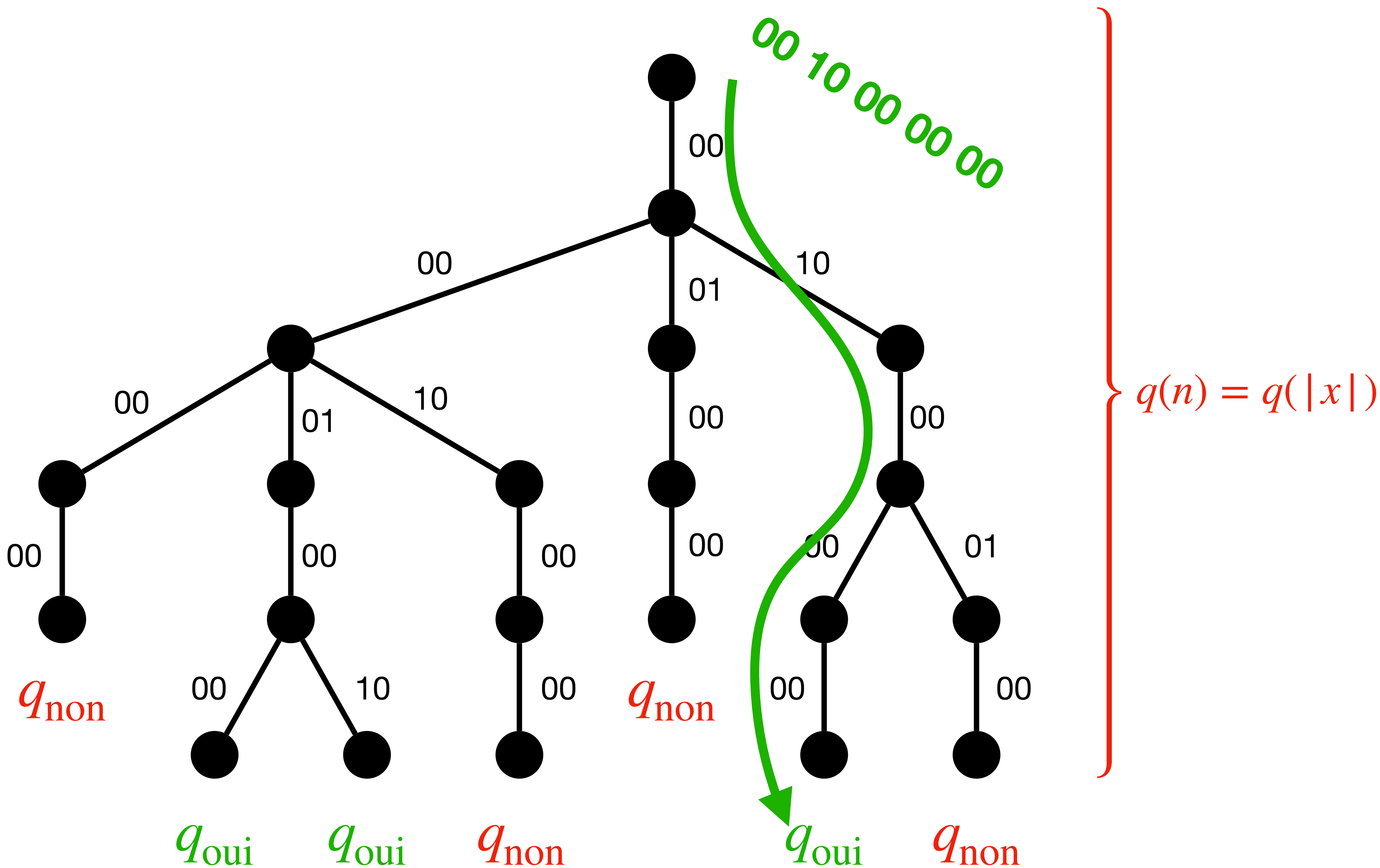
Arbre de calcul de N sur x



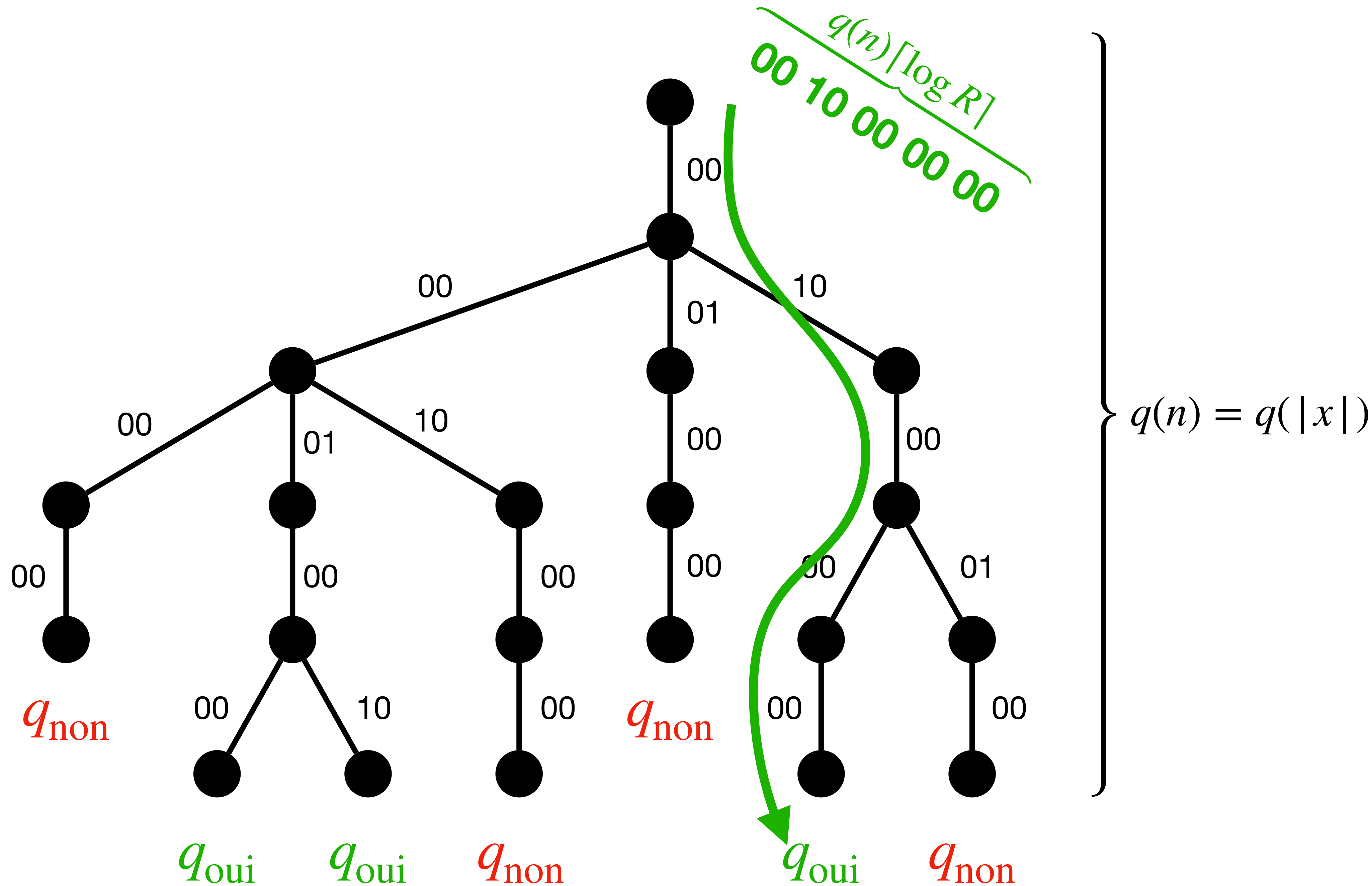
Arbre de calcul de N sur x



Arbre de calcul de N sur x

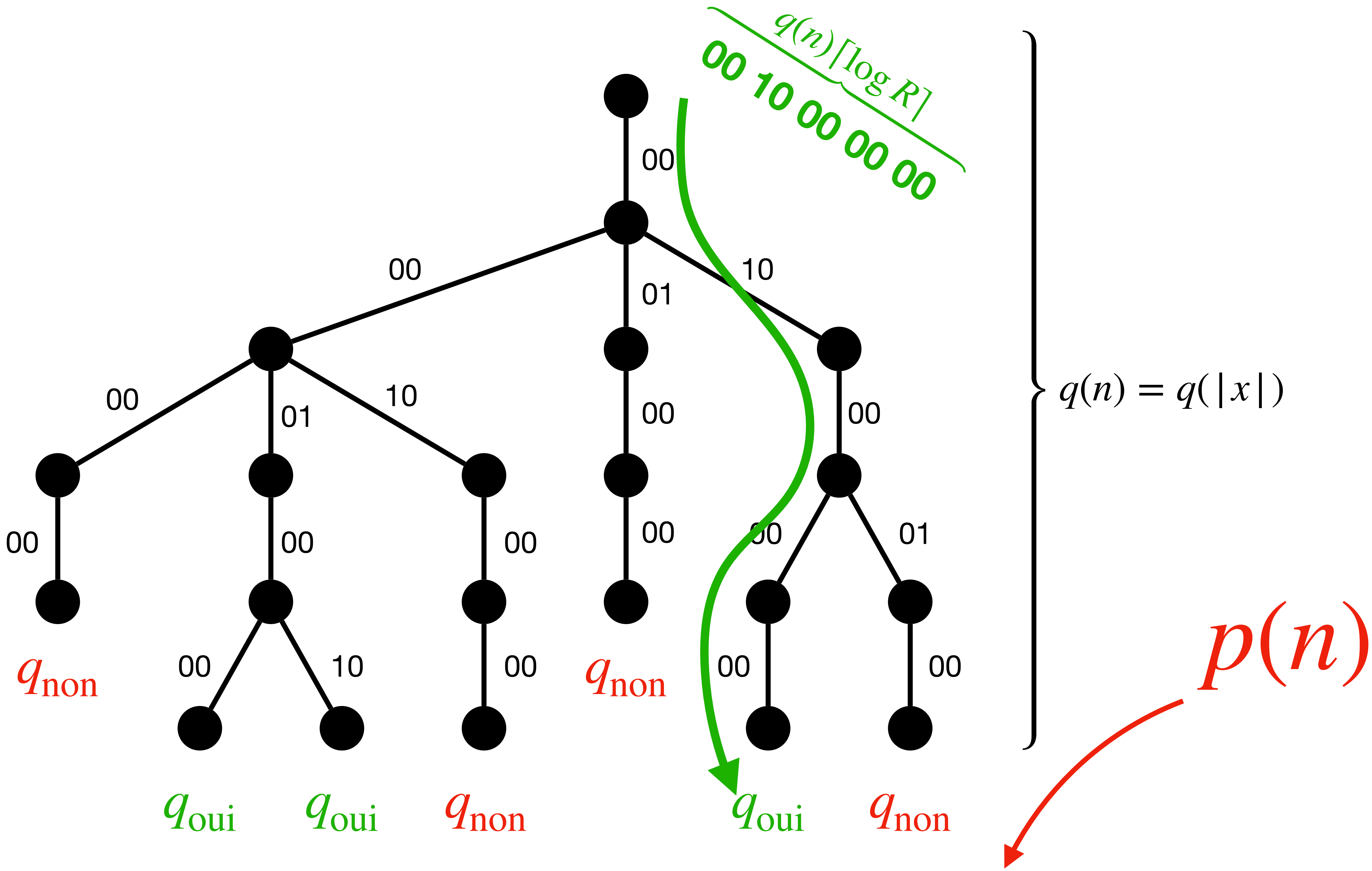


Arbre de calcul de N sur x



Chaque chemin y est décrit par $q(n)\lceil \log R \rceil$ bits

Arbre de calcul de N sur x



Chaque chemin y est décrit par $q(n)\lceil \log R \rceil$ bits

Machine déterministe $M(x, y)$

- Simuler la machine non déterministe N sur l'entrée x
- À chaque étape simulée, choisir la transition indiqué par y
- Comme on connaît le chemin à simuler, on peut faire ça en temps polynomial (déterministe)
- La machine M accepte ssi la machine N a un chemin acceptant

Première implication 👍

- Le langage L est reconnu en temps polynomiale par une machine de Turing non déterministe N

$$\Downarrow$$

- Il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que

$$x \in L \text{ ssi } \exists y \in \{0,1\}^{p(n)}\ M(x,y) \text{ accepte}$$

Deuxième implication 🤔

- Le langage L est reconnu en temps polynomiale par une machine de Turing non déterministe N

$$\Uparrow$$

- Il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que

$$x \in L \text{ ssi } \exists y \in \{0,1\}^{p(n)}\ M(x, y) \text{ accepte}$$

Machine non déterministe $N(x)$

- Deviner un certificat $y \in \{0,1\}^{p(n)}$
- Simuler $M(x, y)$ et accepter ssi cette machine accepte
- Si M fonctionne et temps $q(n)$, sa simulation prend du temps

$$\underbrace{p(n)}_{\textbf{divination}} \quad + \quad \underbrace{q(n + p(n))}_{\textbf{simulation de } M \textbf{ sur } (x,y) \textbf{ vu que } |(x,y)| = n + p(n)}$$

Deuxième implication 👍

- Le langage L est reconnu en temps polynomiale par une machine de Turing non déterministe N

$$\Uparrow$$

- Il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que

$$x \in L \text{ ssi } \exists y \in \{0,1\}^{p(n)}\ M(x,y) \text{ accepte}$$

Algorithmes non déterministes

fonction hamiltonien(V, E)
    $n := |V|$
    $perm :=$ tableau(n)
    pour $i := 0$ **à** $n - 1$ **faire**
        $perm[i] :=$ devine($0, \ldots, n - 1$)
    si $perm$ contient des sommets répétés **alors**
        rejeter
    si $perm$ ne contient pas tous les sommets **alors**
        rejeter
    pour $i := 0$ **à** $n - 1$ **faire**
        si $(perm[i], perm[(i + 1) \bmod n]) \notin E$ **alors**
            rejeter
    accepter
fin

Vérificateurs déterministes

fonction vérificateur-hamiltonien(V, E, $perm$)

  $n := |V|$

  si $perm$ contient des sommets répétés **alors**

    rejeter

  si $perm$ ne contient pas tous les sommets **alors**

    rejeter

  pour $i := 0$ **à** $n - 1$ **faire**

    si $(perm[i], perm[(i+1) \bmod n]) \notin E$ **alors**

      rejeter

  accepter

fin

$\mathbf{P}$ vs $\mathbf{NP}$

- $\mathbf{P}$ est la classe des problèmes faciles à résoudre (de façon déterministe)
- $\mathbf{NP}$ est la classe des problèmes avec des solutions faciles à vérifier (de façon déterministe)
- Facile à résoudre implique facile à vérifier, donc $\mathbf{P} \subseteq \mathbf{NP}$
- On pense que facile à vérifier n'implique nécessairement pas facile à résoudre, donc $\mathbf{P} \neq \mathbf{NP}$
- Mais ça reste un problème ouvert, d'ou le 1000000 $

Proposition 2-BA (p. 62)

Caractérisation universelle de $\mathbf{coNP}$

Un langage L appartient à $\mathbf{coNP}$ ssi il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que

$$x \in L \text{ ssi } \forall y \in \{0,1\}^{p(n)}\ M(x,y) \text{ accepte}$$