Probabilistic Aspects of Computer Science - Part II.1

Anne Bouillard

November 14, 2020
Organization of the course

1. The probabilistic method and probabilistic algorithms
   - The probabilistic method
   - Streaming algorithms
   - Random generation

2. Random graphs
   - Erdös - Renyi random graphs
   - Emergence of the giant component
   - Various types of random graphs
Bibliography


Different types of algorithms

Deterministic algorithm

Each time the algorithm is run on a given data $x$:

- it returns the same result
- it has the same execution time

It is not always possible to have a result that is both correct and fast to compute with deterministic algorithms.
For example, even $O(n^3)$-time algorithm might not be fast enough for large data.
If $P \neq NP$, it is even not possible to have the result in polynomial time...
Probabilistic algorithms

Each time the algorithm is run on a given data \( x \):

- it may return a different result
- it may have a different same execution time

\[
\text{Data } x \rightarrow \text{algorithm } A \rightarrow \text{Result } A(x, r)
\]

Random bits \( r \)

Desirable properties for a probabilistic algorithm

- Either a correct answer in most cases but always fast response
- or a fast response in most cases and always correct.

We need to quantify ”most cases”.

Check the equality of two polynomials

Consider two polynomials of degree $d$, for example $F$ and $G$ below:

$$F(x) = (x + 1)(x - 2)(x + 3)(x - 4)(x + 5)(x - 6) = x^6 - 7x^3 + 25 = G(x)$$

There are several ways to check the equality.

**Algorithm 1: Deterministic algorithm**

Write the two polynomials in a canonical form: $\sum_{i=0}^{d} c_i x^i$;

Check the equality of the coefficients.

A naive algorithm needs $O(d^2)$ operations to develop a polynomial, if we assume that arithmetical operations are performed in constant time (one could also use a divide-and-conquer algorithm in time $O(d \log d)$ but we here keep the naive approach for this example).
A probabilistic algorithm

**Algorithm 2: Probabilistic algorithm**

Choose $r$ in the set $\{1, \ldots, 100d\}$ uniformly at random;
Compute $F(r)$ and $G(r)$ (in time $O(d)$);

if $F(r) = G(r)$ then

| Return $F = G$

else

| Return $F \neq G$

- If $F = G$, then the algorithm returns the correct answer
- If $F \neq G$
  - If $F(r) \neq G(r)$, then the algorithm returns the correct answer
  - If $F(r) = G(r)$, then the algorithm returns the wrong answer

The algorithm is incorrect only if $F \neq G$ and $r$ is a root of $F - G$. But this polynomial has at most $d$ roots in $\{1, \ldots, 100d\}$. The probability that the algorithm returns a wrong answer is at most 1/100.
Las Vegas and Monte Carlo

### Las Vegas

(Ex: Quick sort with random pivot)

- Always returns the correct answer to the problem
- with a finite complexity in average (The goal is to minimize it)

### Monte Carlo

(Ex: Check the equality of two polynomials)

- Returns an approximate answer to the problem (with a controlled error)
- with a complexity that is a deterministic function of the data.

Error probability $\lambda$ of a Monte-Carlo algorithm

- for each entry the error probability is at most $\lambda$
- the proportion of entries with wrong result is $\lambda$
**From Monte-Carlo to Las Vegas**

Consider a Monte-Carlo algorithm with complexity $C_{MC}$ that return:
- either the correct answer
- or error (with probability at most $\lambda$)

It can be transformed into a Las Vegas algorithm be repeating the algorithm while the returned result is error.

The complexity of the Las Vegas algorithm is then

$$E(C_{LV}) \leq \frac{C_{MC}}{1 - \lambda}.$$
Types of "errors" in Monte Carlo algorithms

**Unilateral error**

A Monte Carlo algorithm that returns either `true` or `false` have unilateral errors if it is incorrect only on one type of answer (like when checking the equality between two polynomials).

To minimize the error, it is enough to repeat the algorithm several times.

**Bilateral error**

A Monte Carlo algorithm that returns either `true` or `false` have bilateral errors if it is incorrect for both types of answer.

If the probability of error is strictly less than 1/2, then this probability can be reduced by running the algorithm several times and returning the majority answer.
Termination

- An algorithm terminates with probability $\lambda$ if for each entry the algorithm terminates with probability at least $\lambda$.
- An algorithm almost surely terminates if for each entry it terminates with probability 1.
- An algorithm (surely) terminates if it terminates on all entries, whatever the random bits.
Probabilistic algorithm as a distribution

Las Vegas algorithm = random variable on the set of probabilistic algorithms
This interpretation allows to compare Las Vegas algorithms with deterministic algorithms.

- $\mathcal{A}$ the set of deterministic algorithms solving a problem $P$
- $\mathcal{X}$ the set of entries for this algorithm of size $n$
- $c(a, x)$ the cost (for example the complexity) of algorithm $a$ on entry $x$

1. Probabilistic algorithm $A = \text{random variable on } \mathcal{A}$. The average cost of algorithm $A$ on entry $x$ is $\mathbb{E}[c(A, x)]$, and we are interested in the worst entry regarding this average cost $\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)]$.

2. Consider a distribution on the entries and $X$ random variable on the entries according to this distribution. The average cost of the deterministic algorithm $a$ is $\mathbb{E}[c(a, X)]$, and we are interested in the average cost of the best deterministic algorithm: $\min_{a \in \mathcal{A}} \mathbb{E}[c(a, X)]$. 
Yao’s principle

The complexity of a probabilistic algorithm cannot be better on all entries than the average complexity of the best deterministic algorithm.

Theorem (Yao’s Principle)

Let $A$ be a random variable on $\mathcal{A}$ and $X$ be a random variable on $\mathcal{X}$. Then

$$\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)] \geq \min_{a \in \mathcal{A}} \mathbb{E}[c(a, X)].$$

We have $\mathbb{E}[c(A, x)] = \sum_{a \in \mathcal{A}} \mathbb{P}(A = a)c(a, x)$ et $\mathbb{E}[c(a, X)] = \sum_{x \in \mathcal{X}} \mathbb{P}(X = x)c(a, x)$. So

$$\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)] = \max_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \mathbb{P}(A = a)c(a, x) \geq \sum_{x \in \mathcal{X}} \mathbb{P}(X = x) \sum_{a \in \mathcal{A}} \mathbb{P}(A = a)c(a, x)$$

$$\min_{a \in \mathcal{A}} \mathbb{E}[c(a, X)] = \min_{a \in \mathcal{A}} \sum_{x \in \mathcal{X}} \mathbb{P}(X = x)c(a, x) \leq \sum_{a \in \mathcal{A}} \mathbb{P}(A = a) \sum_{x \in \mathcal{X}} \mathbb{P}(X = x)c(a, x).$$
The probabilistic method

Goal
Prove the existence of objects satisfying some properties using probabilistic argument. In some cases, it is also possible to build some of these objects.

1. The counting argument
2. The first-moment method (expectation argument)
3. Lovász Local Lemma
4. The second-order method
The probabilistic method

Counting argument

Consider a collection of objects \((a_i)_{i \in I}\) where \(I\) is at most denumerable. One wants to prove that at least one of these objects satisfy property \(\mathcal{P}\).

Idea

1. Choose an object \(a_i\) at random by introducing a random variable \(X\) on \(\{a_i\}_{i \in I}\).
2. If \(P(X\text{ satisfait } \mathcal{P}) > 0\), then there exists \(a_i\) satisfying \(\mathcal{P}\).
Application: Ramsey number

coloring the edges of a complete graph $K_n$ with two colors, red and blue such that there is no monochromatic clique of large size.

$R(k)$ is the minimal size of the graph $(n)$ such that is is impossible to find an edge-coloring with no monochrome clique of size $k$.

**Theorem**

If $2^{1-\binom{k}{2}} < 1$, then $R(k) > n$. In other words, it is possible to color the edges of $K_n$ such that there is no monochrome clique of size $k$. 
Ramsey Number

- There are \(2^{\binom{n}{2}}\) possible edge-colorings of \(K_n\) with two colors.
- Let us choose a coloring uniformly at random. This is equivalent to color each edge in red or blue, each color with probability 1/2, independently of the colors of the other edges.
- Let \(i = 1, \ldots, \binom{n}{k}\) be an enumeration of the cliques of size \(k\).
- Let \(A_i\) be the event "\(i\) is a monochrome clique". Then

\[
P(A_i) = 2^{-\binom{k}{2}+1} \quad \text{(deux choix parmi } 2\binom{k}{2}).
\]

\[
P\left(\bigcup_{i=1}^{\binom{n}{k}} A_i\right) \leq \sum_{i=1}^{\binom{n}{k}} P(A_i) = \binom{n}{k} 2^{-\binom{k}{2}+1} < 1.
\]

and \(P\left(\bigcap_{i=1}^{\binom{n}{k}} A_i^c\right) = 1 - P\left(\bigcup_{i=1}^{\binom{n}{k}} A_i\right) > 0\) and there exists a coloring with the desired property.
Construction of a coloring

($k$ is a constant) we use a Monte-Carlo algorithm.

- Color each edge uniformly and independently.
- Check that there is no monochrome clique of size $k$ in the graph: $O(n^k)$
- Error probability: $p = P\left(\bigcup_{i=1}^{\binom{n}{k}} A_i\right)$.
- Transform into a Las Vegas algorithm: repeat until we find a graph with the desired property. $E(\text{temps d’exécution}) = O\left(\frac{n^k}{1-\binom{n}{k}2^{-(k/2)+1}}\right)$.
First-order method

We use arguments based on the expectation of a random variable, and especially the Markov inequality.

**Expectation** of a random variable on $\mathbb{N}$: if $X \sim (p_i)_{i \in \mathbb{N}}$, $E[X] = \sum_{i \in \mathbb{N}} ip_i$

**Linearity of the expectation**

$$E[aX + bY] = aE[X] + bE[Y].$$

**Markov inequality**

$$P[X \geq a] \leq \frac{E[X]}{a}.$$
Expectation argument

Idea: If the expectation of a random variable $X$ is $\mu$, then with strictly positive probability, $X$ takes values at least and at most $\mu$

Theorem

Let $X$ be a real random variable. Then $P(X \geq \mathbb{E}[X]) > 0$ and $P(X \leq \mathbb{E}[X]) > 0$.

$$
\mathbb{E}[X] = \sum_x xP(X = x). \text{ If } P(X \geq \mathbb{E}[X]) = 0, \text{ then } \\
\mathbb{E}[X] = \sum_{x < \mathbb{E}[X]} xP(X = x) < \sum_{x < \mathbb{E}[X]} \mathbb{E}[X]P(X = x) = \mathbb{E}[X],
$$

which is a contradiction.

The same occurs if $P(X \leq \mathbb{E}[X]) = 0$. 

A. Bouillard
PACS
November 14, 2020 20 / 40
Application 1: MAXSAT

**Problem:** $F$ a formula in conjunctive normal form (CNF) (for example, $F = (x_1 \lor x_2 \lor x_3) \land (x_2 \lor \neg x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_4) \ldots$).

**Question:** what is the maximal number of satisfiable clauses?

**Associated decision problem** (NP-complet):

**Data:** $F, k$.

**Question:** Does there exists an assignment of the variables such that $k$ clauses at least are satisfiable?

**Theorem**

Let $F$ be a formula with $m$ clauses, $k_i$ be the number of literals of the $i$-th clause and $k = \min_{i=1}^{m} k_i$. There exists an assignment of the variables that satisfies at least

$$\sum_{i=1}^{m} (1 - 2^{-k_i}) \geq m(1 - 2^{-k})$$

clauses.
MAXSAT: proof

1 Probabilistic space:
   - We denote the variables of the formula by $x_1, \ldots, x_n$.
   - We assign to each variable true or false independently and uniformly: if $X_i = 1_{x_i \text{ is true}}$ then $X_i \sim \text{Ber}(1/2)$.

2 First-moment:
   - Define $E_j$ the event "the $j$-th clause is satisfied", $Y_j = 1_{E_j}$ and $Y = \sum_{j=1}^{m} Y_j$.
   - $\mathbb{E}[Y_j] = 1 - 2^{-k_j}$
   - $\mathbb{E}[Y] = \sum_j 1 - 2^{-k_j} \geq \sum_j (1 - 2^{-k}) = m(1 - 2^{-k})$.
   - Expectation argument: $\mathbb{P}(Y \geq \mathbb{E}[Y]) > 0$, so there exists at least one assignment of the variables satisfying at least that number of clauses.
**MAXSAT: Construction**

**Idea:** use the expectation conditional to at partly defined assignment of the variables.

\[
\mathbb{E}[Y] = \mathbb{E}[Y | X_1 = 1] \mathbb{P}(X_1 = 1) + \mathbb{E}[Y | X_1 = 0] \mathbb{P}(X_1 = 0)
\]
\[
= \frac{1}{2} (\mathbb{E}[Y | X_1 = 1] + \mathbb{E}[Y | X_1 = 0])
\]
\[
\leq \max(\mathbb{E}[Y | X_1 = 1], \mathbb{E}[Y | X_1 = 0]).
\]
MAXSAT: Construction (2)

\[ \mathbb{E}[Y] \leq \max(\mathbb{E}[Y \mid X_1 = 1], \mathbb{E}[Y \mid X_1 = 0]). \]

- Either \( \mathbb{E}[Y \mid X_1 = 1] \) or \( \mathbb{E}[Y \mid X_1 = 0] \) is at least \( \mathbb{E}[Y] \).
- We choose the assignment that maximizes the expectation: \( x_i \) is true if \( \mathbb{E}[Y \mid X_1 = 1] \geq \mathbb{E}[Y \mid X_1 = 0] \).
- We do the same for the other variables: if the variables \( X_1, \ldots, X_k \) are set to \( x_1, \ldots, x_k \) respectively, one can fix that of \( X_{k+1} \):

\[
\mathbb{E}[Y \mid X_1 = x_1, \ldots, X_k = x_k] = \mathbb{E}[Y \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 1]P(X_{k+1} = 1) + \mathbb{E}[Y \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 0]P(X_{k+1} = 0) \\
\leq \max(\mathbb{E}[Y \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 1], \mathbb{E}[Y \mid X_1 = x_1, \ldots, X_k = x_k, X_{k+1} = 0])
\]

and one fix \( X_{k+1} \) to 0 or 1 according to the largest expectation.
Application 2: independent set in a graph

Let $G = (V, E)$ be an undirected graph. One notes $|V| = n$ and $|E| = m$. A subset of vertices $I \subseteq V$ is an independent set of $G$ if $\forall u, v \in I, (u, v) \notin E$ ($I$ is a clique of $G' = (V, \overline{E})$).

**Theorem**

Let $G = (V, E)$ be a connected graph with $n$ vertices and $m$ edges. If $\frac{2m}{n} \geq 1$, then $G$ has an independent set of size at least $\frac{n^2}{4m}$.

Soit $p \in [0, 1]$.

**Algorithm 3:** Construction of an independent set

Erase each vertex of $G$ independently with probability $1 - p$;
For each remaining edge, remove it as well as one of its extremities (vertex)
Independent set of a graph: proof

If a graph contains more vertices than edges, then there exists an independent set of size at least \( n - m \). Indeed, one can erase one vertex at one extremity of each edge, we obtain a graph without edge, with at least \( n - m \) vertices.

First phase: it consists in removing vertices (are their adjacent edges) to build such a graph. One can compute the expectation of the number of vertices and edges at the end of the first phase:

1. Let \( X \) be the number of vertices at the end of the first phase: \( \mathbb{E}[X] = np \).
2. Let \( Y \) be the number of edges at the end of the first phase. An edge is kept if its two extremities are kept, which happens with probability \( p^2 \). So, \( \mathbb{E}[Y] = mp^2 \).
Independent set of a graph: proof (2)

Second phase: at most one vertex per remaining edge is removed. The number of remaining vertices is then at least $X - Y$ and


This expectation is maximized for $p = \frac{n}{2m}$, which is less than 1 by hypothesis. One then obtain

$$E[X - Y] = \frac{n^2}{2m} - \frac{n^2}{4m} = \frac{n^2}{4m}.$$

Note that $d = \frac{1}{p} = \frac{2m}{n}$ is the average degree of the graph.
Beyond independence and union bound

Probabilistic methods usually use independence of event or the union bound.

If $E_1, \ldots, E_n$ are bad event and we want to show that there are situations where none of these bad events happens, it suffices to show that

$$P(\bigcup E_i) > 0.$$ 

Two solutions are possible:

1. $P(\bigcup E_i) = 1 - P(\bigcap E_i) = 1 - \sum_{i=1}^{n} P(E_i)$ (Union bound)
2. $P(\bigcup E_i) = P(\bigcap E_i) = \prod_{i=1}^{n} P(E_i)$ if the events $E_i$ are mutually independent.

How to do when the first method is not enough and event are not independent?

The Lovász local lemma gives an answer when the event are not too much dependent.
Dependency graph

An $E$ is **mutually independent** of $E_1, \ldots, E_n$ if for all $I \subseteq \{1, \ldots, n\}$, $\mathbb{P}(E \mid \cap_{i \in I} E_i) = \mathbb{P}(E)$.

**Definition (Dependency graph)**

A dependency graph of $E_1, \ldots, E_n$ is a graph $G = (V, E)$ such that $V = \{1, \ldots, n\}$ and $E_i$ is mutually independent of $\{E_j \mid (i, j) \notin E\}$.

**Example:**

$$F = (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_3 \lor x_4) \land (x_4 \lor x_5 \lor \neg x_6).$$

Bad events are $E_j$: "the $j$-th clause is not satisfied".

Suppose the assignment of the variables are mutually independent. Then $E_4$ is mutually independent of $E_1$ and $E_2$ because the 4-th clause do not share any literal with the to first clauses.
The probabilistic method

Lovász Local Lemma

Symmetric Lovász Local Lemma

**Theorem (Symmetric Lovász Local Lemma)**

Let $E_1, \ldots, E_n$ be events such that

1. $\forall i \in \{1, \ldots, n\}, \mathbb{P}(E_i) \leq p$.
2. The degree of the vertices of the dependency graph of $E_1, \ldots, E_n$ is at most $d$.
3. $4pd \leq 1$.

Then

$$\mathbb{P}\left(\cap_{i=1}^{n} \bar{E}_i\right) > 0.$$ 

Using the union bound, one should have $pn < 1$. The local lemma is better as soon as $d \leq n/4$. 
Example: \(k\)-SAT

**Data**: A formula \(F\) in CNF with exactly \(k\) distinct variables per clause.

**Question**: Is \(F\) satisfiable?

**Theorem**

If no variable appears more than \(2^k/4k\) times or if no clause shares common variables with \(2^{k-2}\) other clauses, then \(F\) is satisfiable.

- Let \(E_j\) the event "the \(j\)-th clause is not satisfied".
- The variables are assigned uniformly and independently then \(P(E_j) \leq 2^{-k}\).
- \(E_j\) is mutually independent of event concerning clauses that do not share any variables in common with the \(j\)-th clause.
- Each variable appears at most \(2^k/4k\) times, so \(d \leq k \times 2^k/4k = 2^{k-2}\). (the other option lead to the same degree).
- \(4dp \leq 1\).

With \(k = 3\), a clause can share variables with two other clauses.
Constructive proof

We assume a probabilistic space with the following structure:

$$\Omega = C_1 \times \cdots \times C_m$$

where $C_j$ is a finite set and the dimensions are mutually independent: there exists $P_j$ a distribution on $C_j$ such that for all $(a_1, \ldots, a_m) \in C_1 \times \cdots \times C_m$,

$$P(\{a_1, \ldots, a_m\}) = \prod_{j=1}^{m} P_j(\{a_j\}).$$

$A_j$ represent a random variable on $C_j$.

We assume that $E_i$ can be expressed with events of type $A_j \in \ldots$.

For all $i \in \{1, \ldots, n\}$, we denote $e(i)$ the minimal set of $j$ such that $E_i$ is expressed by $A_j$s.

The dependency graph then follows from the edge relation

$$(i, j) \in E \iff e(i) \cap e(j) \neq \emptyset.$$
Example: \( k \)-SAT

\( (i, j) \in E \iff e(i) \cap e(j) \neq \emptyset \).

\( \Omega = \{0, 1\}^n \), where \( n \) is the number of variables in \( F \) and the assignment of variables is independent.

\( F = (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_3 \lor x_4) \land (x_4 \lor x_5 \lor \neg x_6) \).

- \( E_1 \) is "\( \{x_1 = \text{false}\} \cap \{x_2 = \text{false}\} \cap \{x_3 = \text{true}\} \)" et \( e(1) = \{1, 2, 3\} \).
- \( e(3) = \{2, 3, 4\} \)...
The probabilistic method

Lovász Local Lemma

The constructive algorithm

Theorem

The algorithm below builds an affectation the $A_j$ that satisfies $\bigcap E_i$ if $P(A_j) \leq p$ and $p\frac{(d+1)^{d+1}}{d^d} \leq 1$.

Moreover, for fixed $d$ and $n$, this algorithm has an average complexity in $O(m)$.

This is a stronger version as before since $\frac{(d+1)^{d+1}}{d^d} \leq 4d$.

Algorithm 4: Moser and Tardos algorithm(2009)

Assign the variables $(A_j)$ according to their distribution;

while there exists $i \in \{1, \ldots, n\}$ such that $E_i$ happens do

choose $i$ such that $E_i$ happens;

Re-assign the variables $A_j$ for $j \in e(i)$ independently according to their distribution

[RESET]
Proof

**Correction:** The algorithm stops when the assignment of variables is in $\cap E_i$. Detailing the steps of the algorithm:

- $\text{LOG} = i_1, i_2, \ldots, i_t, \ldots$ is the sequence of indices of the event chosen at each step
- pour tout $i$, $\text{COUNT}(i)$ the number of times $i$ appears in $\text{LOG}$

**Average complexity:** If we show that $\mathbb{E}[\text{COUNT}(i)] \leq \frac{1}{d}$, then $\mathbb{E}[|\text{LOG}|] \leq \frac{n}{d} \leq \frac{m(d+1)}{d} \leq 2m$. Indeed, $n \leq (d + 1)m$ because each variable appears in at least $d + 1$ events (otherwise the dependence degree would be $> d$).
Dependency tree $\text{TREE}(t)$

why do we reset $(A_j)_{j \in e(i_t)}$ at step $t$?
what are the steps of the algorithms that let to $E_i$ happening?

$\text{TREE}(t)$ is a labeled tree:

1. The root is labeled by $i_t$, the chosen event for the RESET.
2. If $e(i_t) \cap e(i_{t-1}) = \emptyset$, then $i_{t-1}$ does not appear in the tree. Intuitively, whatever the assignment of variables at step $t - 1$. it has no influence of $E_i$ happening.
3. If $e(i_t) \cap e(i_{t-1}) \neq \emptyset$, then $i_{t-1}$ is a child of $i_t$.
4. Once nodes $i_{u+1}, \ldots, i_t$ have potentially been inserted, we treat $i_u$.
   - If for all $i$ label of the tree, $e(i_u) \cap e(i) = \emptyset$, then $i_u$ does not appear in the tree.
   - Otherwise, $s$ the lowest node among $v$ such that $e(i_v) \cap e(i_u) \neq \emptyset$. Then $i_u$ is a child of $i_s$ (if there are multiple possibilities, choose $s$ minimal among all the possibilities).
Properties of the trees

In the execution of the algorithm, trees TREE(1), ..., TREE(t), ... are built.

1. All trees TREE(t), t ≥ 1 are different:
   - Either TREE(s) and TREE(t) do not have the same label at the root
   - or they have the same label i at the root (same RESET event), but then each occurrence of i in LOG until iteration t appears, and then the number of nodes labeled i are different.
   Moreover, the number of nodes is different.

2. If \( e(i_u) \cap e(i_v) \neq \emptyset \) and \( u < v \), then \( e(i_u) \) is strictly lower than \( \text{est } e(i_v) \).

3. The maximal number of children for each node is \( d + 1 \).
Probability of a tree

Let $T$ be a tree with labels in $i \in \{1, \ldots, n\}$ satisfying the previous properties.

Let $\text{OCCUR}(T)$ be the even "\(\exists t, \text{TREE}(t) = T\)".

**Lemma**

\[
P(\text{OCCUR}(T)) \leq p^{|T|}.
\]

By induction on the size of the tree:

1. **Initialization**: $T = [i]$: this can happen only if all events chosen before are independent of $E_i$, and $E_i$ happens from the start, which is with probability at most $p$.

2. **General case**: let us consider the tree bottom-up.

   - **The lowest leave** ($\ell$ with label $i$) **has a probability at most** $p$ ($E_i$ happens from the start).
   - $T' = T - \{\ell\}$ **has probability at most** $p^{|T|-1}$: HR + the assignment of variables after RESET of the leave has the initial distribution.
   - $T'$ **is independent of leave** $\ell$.

The probability of obtaining $T$ is then at most $p^{|T|}$. 
Counting the trees

Each tree appears only once during the execution of the algorithm, so

\[
\mathbb{E}(\text{COUNT}(i)) = \sum_{T \text{ potential tree with root } i} \mathbb{P}(\text{OCCUR}(T)) \leq \sum_{T \text{ ptwr } i} p^{|T|} \leq \sum_{T \in T_{d+1}} p^{|T|},
\]

where \( T_{d+1} \) is the set of finite sub-trees with at most \( d + 1 \) children for each node.

Let \( g_s \) be the number of finite trees with degree at most \( d + 1 \), with height at most \( s \), and each node has a multiplicative weight \( p \):

\[
\begin{align*}
g_0 &= p \\
g_{s+1} &= p(1 + (d + 1)g_s + \binom{d + 1}{2}g_s^2 + \cdots + \binom{d + 1}{k}g_s^k + \cdots + g_s^{d+1}) \\
&= p(1 + g_s)^{d+1} = f(g_s)
\end{align*}
\]
The probabilistic method Lovász Local Lemma

Counting the trees (2)

\[
g_0 = p \\
g_{s+1} = p(1 + g_s)^{d+1} = f(g_s)
\]

\((g_s)\) is a non-decreasing sequence, that converges to infinity or its least fix-point. By hypothesis,

\[
p(1 + \frac{1}{d})^{d+1} = p\left(\frac{d}{d+1}\right)^{d+1} = \frac{1}{d} \left[p \left(\frac{d+1}{d^d}\right)^{d+1}\right] \leq \frac{1}{d}.
\]

So \(g_s\) has a fix-point and

\[
E(\text{COUNT}(i)) = \sum_{T \in T_{d+1}} p^{|T|} \leq \frac{1}{d}.
\]