Average case, worst case and the universal distribution

Armando Matos

Departamento de Ciência de Computadores
Universidade de Porto

2009

Goals. . .

Goals...

Main result: For any algorithm, if data is distributed according to the universal distribution **m**:

$$\text{average case time} = \text{worst case time} \tag{1}$$

Goals. . .

Main result: For any algorithm, if data is distributed according to the universal distribution **m**:

$$\text{average case time} = \text{worst case time} \tag{1}$$

⋆ SHow why worst case instances have short descriptions. /

Goals. . .

Main result: For any algorithm, if data is distributed according to the universal distribution **m**:

$$\text{average case time} = \text{worst case time} \tag{1}$$

⋆ SHow why worst case instances have short descriptions. /
Present definitions and results needed to explain (1) /

Goals. . .

Main result: For any algorithm, if data is distributed according to the universal distribution **m**:

$$\text{average case time} = \text{worst case time} \tag{1}$$

- ⋆ SHow why worst case instances have short descriptions. / Present definitions and results needed to explain (1) / Prove (1) /

Goals...

Main result: For any algorithm, if data is distributed according to the universal distribution $\mathbf{m}$:

$$\text{average case time} = \text{worst case time} \tag{1}$$

- ⋆ SHow why worst case instances have short descriptions. / Present definitions and results needed to explain (1) / Prove (1) / Reflect about (1)

Goals...

Main result: For any algorithm, if data is distributed according to the universal distribution **m**:

$$\text{average case time} = \text{worst case time} \qquad (1)$$

- ⋆ SHow why worst case instances have short descriptions. / Present definitions and results needed to explain (1) / Prove (1) / Reflect about (1)
- ⋆ Present/review some optimality results:
  - Minimality of K

Goals...

Main result: For any algorithm, if data is distributed according to the universal distribution $\mathbf{m}$:

$$\text{average case time} = \text{worst case time} \tag{1}$$

- ⋆ SHow why worst case instances have short descriptions. / Present definitions and results needed to explain (1) / Prove (1) / Reflect about (1)
- ⋆ Present/review some optimality results:
  - Minimality of K
  - Maximality of $\mathbf{m}$

Goals...

Main result: For any algorithm, if data is distributed according to the universal distribution $\mathbf{m}$:

$$\text{average case time} = \text{worst case time} \tag{1}$$

- ⋆ SHow why worst case instances have short descriptions. / Present definitions and results needed to explain (1) / Prove (1) / Reflect about (1)
- ⋆ Present/review some optimality results:
  - Minimality of K
  - Maximality of $\mathbf{m}$
  - Existence of optimal algorithms for NP

Outline

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\mathrm{pr}(x|n) = 2^{-n}$

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\mathrm{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n): execution time is 2^n
- For 99% of the possible inputs (0.99×2^n): execution time is n

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n
- For 99% of the possible inputs (0.99×2^n):
  execution time is n

What is the average case running time?

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n
- For 99% of the possible inputs (0.99×2^n):
  execution time is n

What is the average case running time?
Answer: $t_{\text{av}}(n) = (0.01 \times 2^n) + (0.99 \times n)$

... an example ...

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n
- For 99% of the possible inputs (0.99×2^n):
  execution time is n

What is the average case running time?
Answer: $t_{\text{av}}(n) = (0.01 \times 2^n) + (0.99 \times n) = \Theta(2^n)$

...an example ...

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n
- For 99% of the possible inputs (0.99×2^n):
  execution time is n

What is the average case running time?
Answer: $t_{\text{av}}(n) = (0.01 \times 2^n) + (0.99 \times n) = \Theta(2^n)$
- For 99% of the inputs, the execution time is $O(n)$...

. . . an example . . .

x is the input, $n = |x|$, 2^n possible inputs with length n.
Uniform distribution, $\text{pr}(x|n) = 2^{-n}$
An algorithm behaves as follows:

- For 1% of the possible inputs (0.01×2^n):
  execution time is 2^n
- For 99% of the possible inputs (0.99×2^n):
  execution time is n

What is the average case running time?
Answer: $t_{\text{av}}(n) = (0.01 \times 2^n) + (0.99 \times n) = \Theta(2^n)$
- For 99% of the inputs, the execution time is $O(n)$. . .
- . . . yet the average time is exponential!

On the number of "bad" inputs

On the number of "bad" inputs

How can $t_{\mathrm{av}}(n)$ be $\Theta(n)$?

On the number of “bad” inputs

How can $t_{\text{av}}(n)$ be $\Theta(n)$? Only if the number “bad” inputs decreases fast enough when n increases.

On the number of “bad” inputs

How can $t_{\text{av}}(n)$ be $\Theta(n)$? Only if the number “bad” inputs decreases fast enough when n increases.
We have

$$t_{\text{av}}(n) = b(n) \times 2^n + (1 - b(n)) \times n$$

where $b(n)$ be the fraction of “bad” inputs – those corresponding to worst case time behavior.

On the number of “bad” inputs

How can $t_{\text{av}}(n)$ be $\Theta(n)$? Only if the number “bad” inputs decreases fast enough when n increases.
We have

$$t_{\text{av}}(n) = b(n) \times 2^n + (1 - b(n)) \times n$$

where $b(n)$ be the fraction of “bad” inputs – those corresponding to worst case time behavior.
If the contribution of bad inputs to $t_{\text{av}}(n)$ is $O(n)$, we must have

On the number of "bad" inputs

How can $t_{av}(n)$ be $\Theta(n)$? Only if the number "bad" inputs decreases fast enough when n increases.

We have

$$t_{av}(n) = b(n) \times 2^n + (1 - b(n)) \times n$$

where $b(n)$ be the fraction of "bad" inputs – those corresponding to worst case time behavior.

If the contribution of bad inputs to $t_{av}(n)$ is $O(n)$, we must have

$$b(n) \leq c \times \frac{n}{2^n}$$

(in this presentation c will always denote a constant.)

Thus there can only exist $b(n) \times 2^n = O(n)$ bad inputs with length n.

On the Kolmogorov complexity of bad inputs

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.
We can describe x by the following 2 parts:

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.
We can describe x by the following 2 parts:

(1) n: from n we get the set of B bad inputs.

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.
We can describe x by the following 2 parts:

(1) n: from n we get the set of B bad inputs.

(2) the index i of x in B

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.
We can describe x by the following 2 parts:

(1) n: from n we get the set of B bad inputs.

(2) the index i of x in B

$$K(x) \leq \overbrace{\log n}^{(1)} + \overbrace{\log n}^{(2)} + O(log(logn)) = O(\log n)$$

On the Kolmogorov complexity of bad inputs

Let x be a bad input. There are $\leq c(n/2^n) \times 2^n = cn$ bad inputs.
We can describe x by the following 2 parts:

(1) n: from n we get the set of B bad inputs.

(2) the index i of x in B

$$K(x) \leq \overbrace{\log n}^{(1)} + \overbrace{\log n}^{(2)} + O(log(logn)) = O(\log n)$$

Bad inputs have short descriptions!

Average complexity under the universal distribution

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

$\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

- $\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small
- $\Rightarrow$ They occur with high probability, $\geq c \times 2^{-2\log n} \sim 1/n^2$, while "good" inputs have probability $\approx 2^{-n}$

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

- $\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small
- $\Rightarrow$ They occur with high probability, $\geq c \times 2^{-2\log n} \sim 1/n^2$, while "good" inputs have probability $\approx 2^{-n}$
- $\Rightarrow$ the average execution time is exponential: $\geq (1/n^2) \times 2^n$

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

- $\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small
- $\Rightarrow$ They occur with high probability, $\geq c \times 2^{-2\log n} \sim 1/n^2$, while "good" inputs have probability $\approx 2^{-n}$
- $\Rightarrow$ the average execution time is exponential: $\geq (1/n^2) \times 2^n$

This analysis was approximate.

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

- $\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small
- $\Rightarrow$ They occur with high probability, $\geq c \times 2^{-2\log n} \sim 1/n^2$, while "good" inputs have probability $\approx 2^{-n}$
- $\Rightarrow$ the average execution time is exponential: $\geq (1/n^2) \times 2^n$

This analysis was approximate.

Exercise. What are the inaccuracies?

Average complexity under the universal distribution

Now assume that the probability distribution (apart from a normalizing constant) is $\mathbf{m}(x) = 2^{-K(x)}$

- $\Rightarrow$ Bad inputs x have short descriptions; $K(x)$ is very small
- $\Rightarrow$ They occur with high probability, $\geq c \times 2^{-2\log n} \sim 1/n^2$, while "good" inputs have probability $\approx 2^{-n}$
- $\Rightarrow$ the average execution time is exponential: $\geq (1/n^2) \times 2^n$

This analysis was approximate.

Exercise. What are the inaccuracies?

In fact: the average case and the worst case execution times have exactly the same order of magnitude!

Outline

Functions computable from above and below

Functions computable from above and below

Definition

$f : \mathbb{N} \to \mathbb{R}$ is computable from above if the set

$$\{(x, y) \; : \; y \geq f(x)\}$$

is recursively enumerable.

Functions computable from above and below

Definition

$f : \mathbb{N} \to \mathbb{R}$ is computable from above if the set

$$\{(x, y) : y \geq f(x)\}$$

is recursively enumerable.

Similarly for computable from below.

Functions computable from above and below

Definition

$f : \mathbb{N} \to \mathbb{R}$ is computable from above if the set

$$\{(x, y) : y \geq f(x)\}$$

is recursively enumerable.

Similarly for computable from below.
Given a value x we can obtain values $y \leq f(x)$ but we may never know when $y = f(x)$ or how far we are from $f(x)$.

Functions computable from above and below

Definition

$f : \mathbb{N} \to \mathbb{R}$ is computable from above if the set

$$\{(x, y) \ : \ y \geq f(x)\}$$

is recursively enumerable.

Similarly for computable from below.
Given a value x we can obtain values $y \leq f(x)$ but we may never know when $y = f(x)$ or how far we are from $f(x)$.

Example. $K(x)$ is computable from above.

Functions computable from above and below

Definition

$f : \mathbb{N} \to \mathbb{R}$ is computable from above if the set

$$\{(x, y) : y \geq f(x)\}$$

is recursively enumerable.

Similarly for computable from below.
Given a value x we can obtain values $y \leq f(x)$ but we may never know when $y = f(x)$ or how far we are from $f(x)$.

Example. $K(x)$ is computable from above.
Example. $\mathbf{m}(x) = 2^{-K(x)}$ is computable from below.

The universal distribution $\mathbf{m}(x)$

The universal distribution $\mathbf{m}(x)$

The universal distribution:

$$\mathbf{m}(x) = 2^{-K(x)}$$

The universal distribution $\mathbf{m}(x)$

The universal distribution:

$$\mathbf{m}(x) = 2^{-K(x)}$$

Simplest objects have highest probability, while random objects x have probability $\approx 2^{-|x|}$.

The universal distribution $\mathbf{m}(x)$

The universal distribution:

$$\mathbf{m}(x) = 2^{-K(x)}$$

Simplest objects have highest probability, while random objects x have probability $\approx 2^{-|x|}$.
Notes:

The universal distribution $\mathbf{m}(x)$

The universal distribution:

$$\mathbf{m}(x) = 2^{-K(x)}$$

Simplest objects have highest probability, while random objects x have probability $\approx 2^{-|x|}$.

Notes:

- $\mathbf{m}$ is not a distribution because $\sum_x \mathbf{m}(x) \neq 1$. A normalizing constant is needed...

The universal distribution $\mathbf{m}(x)$

The universal distribution:

$$\mathbf{m}(x) = 2^{-K(x)}$$

Simplest objects have highest probability, while random objects x have probability $\approx 2^{-|x|}$.

Notes:

- $\mathbf{m}$ is not a distribution because $\sum_x \mathbf{m}(x) \neq 1$. A normalizing constant is needed...
- $\mathbf{m}$ is computable from below.

$\mathbf{m}(x)$ maximizes every computable distribution

$\mathbf{m}(x)$ maximizes every computable distribution

Definition A probability distributions is “enumerable” if it is computable from below.

$\mathbf{m}(x)$ maximizes every computable distribution

Definition A probability distributions is "enumerable" if it is computable from below.

Theorem (Existence of optimal (maximum) enumerable probability distributions)

For every enumerable probability distribution $\mu(x)$, there is a constant $c_\mu \in \mathbb{R}^+$ such that

$$\forall x: \ \mu(x) \leq c_\mu \mathbf{m}(x)$$

$\mathbf{m}(x)$ maximizes every computable distribution

Definition A probability distributions is "enumerable" if it is computable from below.

Theorem (Existence of optimal (maximum) enumerable probability distributions)

For every enumerable probability distribution $\mu(x)$, there is a constant $c_\mu \in \mathbb{R}^+$ such that

$$\forall x: \ \mu(x) \leq c_\mu \mathbf{m}(x)$$

$\mathbf{m}(x)$ is called an universal distribution

Solomonoff, Kolmogorov, Chaitin (1964–1975)

Outline

$K(x)$ minimizes every algorithmic description method

$K(x)$ minimizes every algorithmic description method

Theorem (Existence of optimal (minimum) recursive descriptions)

There exists a partial recursive function $\phi_u(x)$ such that, for every partial recursive function ϕ_i

$$\exists c_i \in \mathbb{R}^+, \forall x : \; C_{\phi_i}(x) \geq C_{\phi_u}(x) + c_i$$

where $C_f(x)$ is the (plain) Kolmogorov complexity of x relatively to partial recursive function f.

$K(x)$ minimizes every algorithmic description method

Theorem (Existence of optimal (minimum) recursive descriptions)

There exists a partial recursive function $\phi_u(x)$ such that, for every partial recursive function ϕ_i

$$\exists c_i \in \mathbb{R}^+,\ \forall x :\ C_{\phi_i}(x) \geq C_{\phi_u}(x) + c_i$$

where $C_f(x)$ is the (plain) Kolmogorov complexity of x relatively to partial recursive function f.
ϕ_u is an universal distribution

$K(x)$ minimizes every algorithmic description method

Theorem (Existence of optimal (minimum) recursive descriptions)

There exists a partial recursive function $\phi_u(x)$ such that, for every partial recursive function ϕ_i

$$\exists c_i \in \mathbb{R}^+, \forall x : \; C_{\phi_i}(x) \geq C_{\phi_u}(x) + c_i$$

where $C_f(x)$ is the (plain) Kolmogorov complexity of x relatively to partial recursive function f.
ϕ_u is an universal distribution

Compare with the result in previous slide!

Solomonoff (1964), Levin (197?)

For every NP problem there is a fastest algorithm!

About NP problems

For every NP problem there is a fastest algorithm!

About NP problems

- NP problem: characterized by a binary relation $r(x,y)$ computable in time polynomial in $|x| + |y|$); x is the input, y is the "solution" or "witness".

About NP problems

- NP problem: characterized by a binary relation $r(x, y)$ computable in time polynomial in $|x| + |y|$); x is the input, y is the "solution" or "witness".
- Language associated with r: $\mathcal{L}(r) = \{x \ : \ \exists y,\ r(x, y)\}$

For every NP problem there is a fastest algorithm!

About NP problems

- NP problem: characterized by a binary relation $r(x, y)$ computable in time polynomial in $|x| + |y|$); x is the input, y is the "solution" or "witness".
- Language associated with r: $\mathcal{L}(r) = \{x : \exists y,\ r(x, y)\}$
- Algorithm A solves $r(x, y)$: The input is x.
  If $x \in \mathcal{L}(r)$, outputs some y such that $r(x, y)$ holds.
  (If $x \notin \mathcal{L}(r)$, the behavior of A is unspecified)

For every NP problem there is a fastest algorithm!

Theorem (Existence of an optimal (fastest) algorithm for NP problems)

For every NP relation $r(x, y)$ there is an algorithm A that solves $r(x, y)$ and a polynomial $p(\cdot)$ so that, if A' is any algorithm that solves $r(x, y)$

$$\exists c, \forall x : \ \mathrm{time}_A(x) \leq c \times \mathrm{time}_{A'}(x) + p(|x|)$$

In words: A is optimal up to a fixed constant and an additive polynomial!

For every NP problem there is a fastest algorithm!

Theorem (Existence of an optimal (fastest) algorithm for NP problems)

For every NP relation $r(x, y)$ there is an algorithm A that solves $r(x, y)$ and a polynomial $p(\cdot)$ so that, if A' is any algorithm that solves $r(x, y)$

$$\exists c, \forall x : \ \mathrm{time}_A(x) \leq c \times \mathrm{time}_{A'}(x) + p(|x|)$$

In words: A is optimal up to a fixed constant and an additive polynomial! Example: some A' is polynomial $\Rightarrow$ A is polynomial.

Compare with previous results!

For every NP problem there is a fastest algorithm!

Theorem (Existence of an optimal (fastest) algorithm for NP problems)

For every NP relation $r(x, y)$ there is an algorithm A that solves $r(x, y)$ and a polynomial $p(\cdot)$ so that, if A' is any algorithm that solves $r(x, y)$

$$\exists c, \forall x : \ \mathrm{time}_A(x) \leq c \times \mathrm{time}_{A'}(x) + p(|x|)$$

In words: A is optimal up to a fixed constant and an additive polynomial! Example: some A' is polynomial $\Rightarrow$ A is polynomial.

Compare with previous results!

This result is in a 2 page paper by Levin (1973) where he also proves the existence of NP-complete problems (Cook's theorem)!

Levin I – On the existence of maximal (or minimal) elements in algorithmic theories

Levin I – On the existence of maximal (or minimal) elements in algorithmic theories

Levin in "Randomness conservation inequalities" (1984):

BEGIN LEVIN . . . the general Theory of Algorithms is very similar to descriptive set theory. There is, however, an important exception in the existence of universal algorithms. The set of all (countable) sets of integers is uncountable while the set of r.e. sets is r.e..

Levin I – On the existence of maximal (or minimal) elements in algorithmic theories

Levin in "Randomness conservation inequalities" (1984):

BEGIN LEVIN ...the general Theory of Algorithms is very similar to descriptive set theory. There is, however, an important exception in the existence of universal algorithms. The set of all (countable) sets of integers is uncountable while the set of r.e. sets is r.e.. This rather abstract difference opens, however, new analytical possibilities having no analogies in "non-algorithmic" analysis.

Levin II

Levin II

Let us illustrate this with a simple but important example. Let $l_1 \subseteq \mathbb{R}^{\mathbb{N}}$ be the space of absolutely summable real sequences.

Levin II

Let us illustrate this with a simple but important example. Let $l_1 \subseteq \mathbb{R}^{\mathbb{N}}$ be the space of absolutely summable real sequences. Its recursive analogue $\overline{l_1} \subseteq l_1$ consists of elements of l_1 whose sub-graph $\{(r, x) \ : \ p(x) > r \in \mathbb{Q}\}$ is r.e.

Levin II

Let us illustrate this with a simple but important example. Let $l_1 \subseteq \mathbb{R}^{\mathbb{N}}$ be the space of absolutely summable real sequences. Its recursive analogue $\overline{l_1} \subseteq l_1$ consists of elements of l_1 whose sub-graph $\{(r, x)\ :\ p(x) > r \in \mathbb{Q}\}$ is r.e.
It is known in calculus that no element is maximal in l_1 within a constant factor. In contrast to this, $\overline{l_1}$ has an "absorbing" element m (a universal measure) such that

$$\forall q \in \overline{l_1} :\ \sup\{q(x)/m(x)\} < \infty$$

where $m(x) = \ldots$
END LEVIN

Optimal objects I

Optimal objects I

Sumary of optimality results:

Optimal objects I

Sumary of optimality results:

- There is an optimal (minimum) algorithmic description method

Optimal objects I

Sumary of optimality results:

- There is an optimal (minimum) algorithmic description method
- There is an optimal (greatest) algorithmic probability distribution

Optimal objects I

Sumary of optimality results:

- There is an optimal (minimum) algorithmic description method
- There is an optimal (greatest) algorithmic probability distribution
- For every NP problem there is an optimal (fastest) algorithm

Optimal objects I

Sumary of optimality results:

- There is an optimal (minimum) algorithmic description method
- There is an optimal (greatest) algorithmic probability distribution
- For every NP problem there is an optimal (fastest) algorithm

Optimal objects II

Optimal objects II

Why are there optimal objects in the "recursive world"?

Optimal objects II

Why are there optimal objects in the "recursive world"?

Because there are machines (=algorithms) capable of simulating any other machine!

Optimal objects II

Why are there optimal objects in the "recursive world"?

Because there are machines (=algorithms) capable of simulating any other machine!

`END OF DIGRESSION`!

Outline

The main theorem

The main theorem

Theorem (Input distribution $\mathbf{m}$ $\Rightarrow$ worst=average)

Assume that the inputs of an algorithm A (which terminates for all inputs) are distributed according to $\mathbf{m}$*. Then, the average case and the worst case time complexity of A have the same order of magnitude.*

The proof (I)

Proof. Let $t_w(n)$ be the worst case complexity of A. Define a particular probability distribution μ by

The proof (I)

Proof. Let $t_w(n)$ be the worst case complexity of A. Define a particular probability distribution μ by

- Let $a_n = \sum_{|x|=n} \mathbf{m}(x)$

The proof (I)

Proof. Let $t_w(n)$ be the worst case complexity of A. Define a particular probability distribution μ by

- Let $a_n = \sum_{|x|=n} \mathbf{m}(x)$
- For each $n \in \mathbb{N}$ define for each x with $|x| = n$:
  - $\mu(x) = a_n$ if $t(x) = t_w(n)$ (x is a bad input) and x is the lexicographically least such input.
  - $\mu(x) = 0$ otherwise

Thus, for each $n \in \mathbb{N}$ there is exactly one x with $|x| = n$ and $\mu(x) > 0$.

The proof (I)

Proof. Let $t_w(n)$ be the worst case complexity of A. Define a particular probability distribution μ by

- Let $a_n = \sum_{|x|=n} \mathbf{m}(x)$
- For each $n \in \mathbb{N}$ define for each x with $|x| = n$:
  - $\mu(x) = a_n$ if $t(x) = t_w(n)$ (x is a bad input) and x is the lexicographically least such input.
  - $\mu(x) = 0$ otherwise

  Thus, for each $n \in \mathbb{N}$ there is exactly one x with $|x| = n$ and $\mu(x) > 0$.

$\mathbf{m}$ is enumerable $\Rightarrow$ $\mu(x)$ is enumerable

The proof (I)

Proof. Let $t_w(n)$ be the worst case complexity of A. Define a particular probability distribution μ by

- Let $a_n = \sum_{|x|=n} \mathbf{m}(x)$
- For each $n \in \mathbb{N}$ define for each x with $|x| = n$:
  - $\mu(x) = a_n$ if $t(x) = t_w(n)$ (x is a bad input) and x is the lexicographically least such input.
  - $\mu(x) = 0$ otherwise

  Thus, for each $n \in \mathbb{N}$ there is exactly one x with $|x| = n$ and $\mu(x) > 0$.

$\mathbf{m}$ is enumerable $\Rightarrow$ $\mu(x)$ is enumerable
Notice that $\sum_{|x|=n} \mu(x) = \sum_{|x|=n} \mathbf{m}(x)$.

(to be continued)

The proof (II)

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{\text{av}}^{\mathbf{m}}(n)$: average case time under **m**

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{\text{av}}^{\mathbf{m}}(n)$: average case time under **m**

$$t_{\text{av}}^{\mathbf{m}}(n) \quad = \quad \frac{\sum_{|x|=n} \mathbf{m}(x) t(x)}{\sum_{|x|=n} \mathbf{m}(x)}$$

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{\rm av}^{\mathbf{m}}(n)$: average case time under **m**

$$\begin{aligned} t_{\rm av}^{\mathbf{m}}(n) &= \frac{\sum_{|x|=n} \mathbf{m}(x) t(x)}{\sum_{|x|=n} \mathbf{m}(x)} \\ &\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mathbf{m}(x)} t_w(n) \end{aligned}$$

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{av}^{\mathbf{m}}(n)$: average case time under **m**

$$
\begin{aligned}
t_{av}^{\mathbf{m}}(n) &= \frac{\sum_{|x|=n} \mathbf{m}(x) t(x)}{\sum_{|x|=n} \mathbf{m}(x)} \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mathbf{m}(x)} t_w(n) \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mu(x)} t_w(n)
\end{aligned}
$$

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{av}^{\mathbf{m}}(n)$: average case time under **m**

$$
\begin{aligned}
t_{av}^{\mathbf{m}}(n) &= \frac{\sum_{|x|=n} \mathbf{m}(x) t(x)}{\sum_{|x|=n} \mathbf{m}(x)} \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mathbf{m}(x)} t_w(n) \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mu(x)} t_w(n) \\
&\geq \frac{1}{c_\mu} t_w(n)
\end{aligned}
$$

The proof (II)

$t_w(n)$: worst case time of A (which is the same any distribution!)
$t_{\text{av}}^{\mathbf{m}}(n)$: average case time under **m**

$$
\begin{aligned}
t_{\text{av}}^{\mathbf{m}}(n) &= \frac{\sum_{|x|=n} \mathbf{m}(x) t(x)}{\sum_{|x|=n} \mathbf{m}(x)} \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mathbf{m}(x)} t_w(n) \\
&\geq \frac{1}{c_\mu} \sum_{|x|=n} \frac{\mu(x)}{\sum_{|x|=n} \mu(x)} t_w(n) \\
&\geq \frac{1}{c_\mu} t_w(n)
\end{aligned}
$$

and, as $t_w(n) \geq t_{\text{av}}^{\mathbf{m}}(n)$: $\boxed{t_w(n) = \Theta(t_{\text{av}}^{\mathbf{m}}(n))}$

Outline

Raising a few questions
about the average case behavior

Comments and reflections I

Comments and reflections I

- What about best case behavior?

Comments and reflections I

- What about best case behavior?
- What about space complexity?

Comments and reflections I

- What about best case behavior?
- What about space complexity?
- Are "real life" data distribution more "enumerable" than "random"? $\longrightarrow$

Comments and reflections II

Comments and reflections II

- Most real life data distributions are far from random.

Comments and reflections II

- Most real life data distributions are far from random.
- With high probability, enumerable distributions are close to **m**: for every $k > 0$

$$\sum \left\{ \mu(x) \ : \ \mu(x) \in \left[\frac{\mathbf{m}(x)}{k}, \, 2^{K(\mu)+O(1)}\mathbf{m}(x) \right] \right\} \geq 1 - 1/k$$

Comments and reflections II

- Most real life data distributions are far from random.
- With high probability, enumerable distributions are close to **m**: for every $k > 0$

$$\sum \left\{ \mu(x) \ : \ \mu(x) \in \left[\frac{\mathbf{m}(x)}{k}, \, 2^{K(\mu)+O(1)} \mathbf{m}(x) \right] \right\} \geq 1 - 1/k$$

"In absence of any a priory knowledge of the actual distribution, apart from the fact that it is enumerable, studying the average behavior under **m** is considerably more meaningful than studying the average behavior under any other particular enumerable distribution" (Li, Vitanyi)

Comments and reflections III

Comments and reflections III

How to test quicksort with computer generated data?

Comments and reflections III

How to test quicksort with computer generated data?

- Data used for tests is usually enumerable.

Comments and reflections III

How to test quicksort with computer generated data?

- Data used for tests is usually enumerable.
  How can quicksort (say) behave well in such tests?

Comments and reflections III

How to test quicksort with computer generated data?

- Data used for tests is usually enumerable.
  How can quicksort (say) behave well in such tests?
- In particular, how can the "algorithmic shuffling" (using pseudo-random generators) be effective for generating well behaved data for quicksort?

The reader can attack himself these questions…

…or look (in vain?) for the answers in the literature.

Bibliographic notes

- The universal distribution **m** was discovered by Ray Solomonoff in 1964.
- For any algorithm, if the inputs are distributed according to **m**, the average time equals the worst time. This property is applied to some particular problems in T. Jiang, M. Li, and P. Vitanyi, "Average-case analysis of algorithms using Kolmogorov complexity", Journal of Computer Science and Technology, 15:5(2000), pp 402–408, url=`http://www.cwi.nl/~paulv/papers/jcst00.ps`.
- A paper by where the existence of fastest algorithms for NP problems is discussed: Levin, "Randomness conservation inequalities", Information and Control 61:1(1984), pp 15-37, url=`http://www.cs.bu.edu/fac/lnd/research/dvi/inf.dvi`.
- ...and, of course, the (current) bible on Kolmogorov complexity is always useful: M. Li and P.M.B. Vitanyi, An Introduction to Kolmogorov Complexity and its Applications, Springer-Verlag, New York, Second Edition, 1997.

The end. . .