The Incompressibility Method using Kolmogorov Complexity

- A key problem in algorithms: analyze the average case performance of an algorithm.

- Using the Incompressibility Method:
  - Give the program a random input (with high Kolmogorov complexity)
  - Analyze the program with respect to this single and fixed input. This is usually easier using the fact this input is incompressible.
  - The running time for this single input is the average case running time of all inputs!
Using the The Incompressibility Method for Solution to open questions

- We will discuss the histories of some open questions and the ideas of how they were solved by the incompressibility method.

- We will not be able to give detailed proofs to these problems … but hopefully by telling you the ideas, you will be convinced enough and able to reconstruct the details on your own.

- This will Illustrate the power of Kolmogorov complexity to provide proofs that are short and elegant.
Kolmogorov Theory Applications to Complexity Theory

- Proofs that certain sets are not regular
- Complexity Lower Bounds for 1 Tape TMs
- Communication Lower Bounds: What is the distance between two pieces of information carrying entities? For example, distance from an internet query to an answer.
Selected list of results proved by the incompressibility method

- Lower Bound $\Omega(n^2)$ for simulating 2 tapes by 1 (open 20 years)
- $k$ heads > $k-1$ heads for PDAs (open 15 years)
- $k$ one-ways heads can’t do string matching (open 13 yrs)
- 2 heads are better than 2 tapes (open 10 years)
- Average case analysis for heapsort (open 30 years)
- $k$ tapes are better than $k-1$ tapes. (open 20 years)
- Many theorems in combinatorics, formal language/automata, parallel computing, VLSI
- Simplify old proofs (Hastad Lemma).
- Shellsort average case lower bound (open 40 years)
Example: Show \( L = \{0^i1^i \mid k>0 \} \) not regular.

Proof: For sake of contradiction, assume DFA \( M \) accepts \( L \).
Choose \( i \) so that \( K(i) >> 2|\langle M \rangle| \). Simulate \( M \):

\[
\begin{array}{c}
\text{000 \ldots 0} \\
\text{111 \ldots 1}
\end{array}
\]

If \( M \) is in state \( q \) just after scanning \( 0^i \), then \( K(i) < |\langle M \rangle| + q + O(1) \) which is \(< 2|\langle M \rangle| \), Contradiction.

Remark: Incompressibility method is easier to use than “pumping lemmas”.
A simple Turing machine lower bound

Consider one-tape TM. Input tape is also work tape, allow read/write, two-way head.

Theorem. It takes $\Omega(n^2)$ time for such TM M to accept $L=\{ww \mid w \in \Sigma^*\}$.

Proof (W. Paul). Take an incompressible x where $K(x) \geq |x| = n - c$ for some appropriate constant. Consider M’s computation on input: $ww$ where $w = 0^n x$. Consider, for each $i=1, \ldots, n$, the $(i+2n)$th tape cell’s crossing sequence (giving the moves of the TM’s head on that tape cell):

- If the crossing sequence is $o(n)$, we can use this crossing sequence to find x by simulating on the “right side” of the crossing sequence by trying all the strings of length n. Then $K(x) = o(n)$, contradiction.

- If the crossing sequence is $\Omega(n)$, then the computation time is $\Omega(n^2)$. QED
More on formal language theory

Let $L \subseteq V^*$ be a regular language.
Let $L_x = \{y: xy \in L\}$ be the set of suffixes.

Li-Vitanyi Lemma If $y$ is the $n$-th element of a lexical order enumeration of $L_x$, then $K(y) \leq K(n)+c$ for a constant $c$ (for all $x,y,n$).

Proof. The $n$th string $y$ can be described by

(i) $n$
(ii) the finite state machine recognizing $L$, and
(iii) the state of that machine after processing $x$.

Hence the required total information is $K(n)+O(1)$.

QED.
Characterizing regular sets

For any enumeration of $\Sigma^*={y_1, y_2, \ldots}$, define characteristic sequence of $L_x=${$y_i : xy_i \in L$} by Booleans:

$$B_i = 1 \text{ iff } xy_i \in L$$

**Theorem.** $L$ is regular iff there is a constant $c$ where for all $x$ and $n$,

$$K(B_n|n) \leq c$$
Theorem. \{1^p : p \text{ is prime}\} is not regular.

Proof. Assume is it regular.
Let \( P_i, i=1,2 \ldots \), be the list of primes.
Let string \( xy = 1^{P_{k+1}} \) with \( P_{k+1} \) the \((k + 1)\)th prime.
Let string \( x = 1^{P_k} \), with \( P_k \) the \( k \)th prime.
Then \( y = 1^{P_{k+1} - P_k} \) is lex. 1st element in \( L_x = \{y: xy \in L\} \).
Hence, by the Li-Vitanyi Lemma, \( K(P_{k+1} - P_k) = O(1) \).
But by Prime Number Theorem, the difference between consecutive primes grows unboundedly.
Since there are only \( O(1) \) descriptions of length \( O(1) \), we have a contradiction. QED
Further Results in Complexity Theory Proved using Kolmogorov Theory

- 1 tape vs 2 tape Turing machines
- Lower Bounds for K-head PDA’s
- String-matching by k-DFA
1 tape vs 2 tape Turing machines

- Standard (on-line) TM Model:
  - Input tape $\rightarrow$ one way
  - Finite Control
  - Work tape, two way

- Question since the 1960’s: Are two work tapes better than 1 work tape? How many works tapes are needed?
History of 1 tape vs 2 tape Turing machines

- 1969. Hartmanis & Stearns: 1 work tape TM can simulate $k > 1$ tape TM in $O(n^2)$ time.
- 1963. Rabin: 2 work tapes are better than 1.
- 1966. Hennie-Stearns: 2 work tapes can simulate $k$ tapes in $O(n \log n)$ time.
- 1982. Paul: $\Omega(n(\log n)^{1/2})$ lower bound for 1 vs 2 work tapes.
- 1983. Duris-Galil: Improved to $\Omega(n \log n)$.
- 1985. Maass, Li, Vitanyi: $\Omega(n^2)$ tight bound, by incompressibility method, settling the 20 year effort.
Proof Sketch
1 tape vs 2 tape Turing machines

Here is the language we have used to prove an $\Omega(n^{1.5})$ lower bound:

$L = \{x_1 \# x_2 \# \ldots \# x_k \# y_1 \# \ldots \# y_l : x_i = y_j \}$

Choose c-incompressible $x$, $K(x) \geq |x| = n$, evenly break $x$ into $x_1 \ldots x_k$, $k = \sqrt{n}$.

Then the two work tape machine can easily put $x_i$ blocks on one tape and $y_j$ blocks on the other. Then it accepts this language in linear time.

However, the one work tape machine has trouble where to put these blocks. Whichever way it does it, there bounds to be some $x_i$ and $y_j$ blocks that are far away, then our previous proof works. The proof needs to worry about not many blocks can be stored in a small region (they are non-compressible strings, hence intuitively we know they can’t be). The nice thing about Kolmogorov complexity is that it can directly formulate your intuition into formal arguments.

To improve to $\Omega(n^2)$ lower bound, we just need to make each $x_i$ to be constant size. Then argue there are $O(n)$ pairs of $(x_i, y_j)$ need to be matched and they are $O(n)$ away.
K-head PDA’s

- **k-PDA**: Pushdown automaton with k one-way input heads.
- These are natural extensions of our standard definition of FA and PDA.
- Two conjectures:
  - 1965, Rosenberg Conjecture: \((k+1)\)-FA > k-FA
  - 1968, Harrison-Ibarra Conjecture: \((k+1)\)-PDA > k-PDA
The language we have used is:
\[ L_b = \{ w_1 \# \ldots \# w_b \$ w_b \# \ldots \# w_1 \mid w_i \in \{0,1\}^* \} \]

**Theorem.** \( L_b \) can be accepted by a k-PDA iff \( b \leq k(k-1)/2 \).

When \( b \leq k(k-1)/2 \), then a k-FA can do it by pairing its k heads at right places at right time.

When \( b > k(k-1)/2 \), then we can again choose random \( w \) and break it into \( w_i \) blocks. Then we say there must be a pair of \( (w_i, w_i) \) that are indirectly matched (via the pushdown store).

But when storing into pushdown store, \( w_i \) is reversed, so it cannot be properly matched with its counter part \( w_i \).

We will also need to argue information cannot be reversed, compressed etc.

But these are all easy with Kolmogorov complexity.
String-matching by k-DFA

- String matching problem:
  \[ L = \{ x#y \mid x \text{ is a substring of } y \} \]
- This one of the most important problems in computer science (grep function for example)
- Hundreds of papers written.
- Many efficient algorithms – KMP, BM, KR. Main features of these algorithms:
  - Linear time
  - Constant space (not KMP, BM), i.e. multihead finite automaton. In fact, a two-way 6-head FA can do string matching in linear time (Galil-Seiferas, 1981, STOC)
  - No need to back up pointers in the text (e.g. KMP).
- Galil-Seiferas Conjecture: Can k-DFA for any k, do string matching?
History of String-matching by k-DFA

- Li-Yesha: 2-DFA cannot.
- Gereb-Graus-Li: 3-DFA cannot
- Jiang-Li 1993 STOC: k-DFA cannot, for any k.
Sketch of Proof of String-matching by k-DFA

- Just proof sketch for 2-DFA.
- Remember the heads are one-way, and DFA does not remember much.
- We can play a game with the 2-DFA with input (of course with Kolmogorov random blocks):
  \[ xy \# y' x' \]
  such that \( x' \) can be \( x \) and \( y' \) can be \( y \), so if the 2-DFA decides to match \( x \), \( x' \) directly, then it won’t be able to match \( y \), \( y' \) directly (and vice versa), so then we simply make \( x' \) different from \( x \), but \( y' = y \). Then without the two heads simultaneously at \( y \) and \( y' \), we will argue, as before, that finite control cannot do it.
Consider matrix over GF(2): 0,1 elements, with usually Boolean $\times$,+$ operations. Lower bounds on Rank are needed for example in proving tradeoff optimal bound $TS=\Omega(n^3)$ for multiplying 2 matrices.

**Theorem.** For each $n$, there is an $n \times n$ matrix over GF(2) s.t. every submatrix of $s$ rows and $n-r$ columns has at least rank $s/2$, $2\log n < r, s < n/4$.

**Proof.** Take $|x|=n^2$, $K(x) \geq n^2$. Form an $n \times n$ matrix with $x$, one bit per entry. For any submatrix $R$ with $s$ rows and $n-r$ columns, if $R$ does not have rank $s/2$, then $s/2+1$ rows can be linearly described by other rows. Then you can compress the original matrix, hence $x$ has desired lower bound on rank. QED
Sorting

- Given n elements (in an array). Sort them into ascending order.
- This is the most studied fundamental problem in computer science.

- Heapsort: Open for 40 years: Which is better in average case: Williams or Floyd?

- Shellsort (1959): P passes. In each pass, move the elements “in some stepwise fashion” (Bubblesort)
  - Open for over 40 years: a nontrivial general average case complexity lower bound of Shellsort?
Heapsort

- Immediately it was improved by RW Floyd.
- Worst case $O(n \log n)$.
- Open for 40 years: Which is better in average case: Williams or Floyd?
Heapsort average analysis (I. Munro)

Average-case analysis of Heapsort.

Heapsort: (1) Make Heap. O(n) time. (2) Deletemin, restore heap, repeat.

Fix random heap H, K(H) > n log n. Simulate Step (2). Each round, encode the red path in \( \log n - d \) bits. The \( n \) paths describe the heap! Hence, total \( n \) paths, length \( \geq n \log n \), \( d \) must be a constant. Floyd takes \( n \log n \) comparisons, and Williams takes \( 2n \log n \).
Shellsort Algorithm

- Using $p$ increments $h_1, \ldots, h_p$, with $h_p = 1$
- At $k$-th pass, the array is divided in $h_k$ separate sublists of length $n/h_k$ (taking every $h_k$-th element).
- Each sublist is sorted by insertion/bubble sort.

Application: Sorting networks --- $n \log^2 n$ comparators.
Shellsort history

- Invented by D.L. Shell [1959], using $p_k = n/2^k$ for step $k$. It is a $\Theta(n^2)$ time algorithm.
- Papernow&Stasevitch [1965]: $O(n^{3/2})$ time.
- Pratt [1972]: $O(n\log^2 n)$ time.
- Incerpi-Sedgewick, Chazelle, Plaxton, Poonen, Suel (1980’s) – worst case, roughly, $\Theta(n\log^2 n / (\log \log n)^2)$.
- Average case:
  - Knuth [1970’s]: $\Theta(n^{5/3})$ for $p=2$
  - Yao [1980]: $p=3$  
  - Janson-Knuth [1997]: $\Omega(n^{23/15})$ for $p=3$.
  - Jiang-Li-Vitanayi [J.ACM, 2000]: $\Omega(pn^{1+1/p})$ for any $p$.  


**Theorem.** $p$-pass Shellsort average case $T(n) \geq pn^{1+1/p}$

Proof. Fix a random permutation $\Pi$ with Kolmogorov complexity $n \log n$. I.e. $C(\Pi) \geq n \log n$. Use $\Pi$ as input.

For pass $i$, let $m_{i,k}$ be the number of steps the $k$th element moves. Then $T(n) = \sum_{i,k} m_{i,k}$

From these $m_{i,k}$'s, one can reconstruct the input $\Pi$, hence

$\sum \log m_{i,k} \geq C(\Pi) \geq n \log n$

Maximizing the left, all $m_{i,k}$ must be the same. Call it $m$.

$\sum \log m = pn \log m \geq \sum \log m_{i,k} \geq n \log n \Rightarrow m^p \geq n$.

So $T(n) = pnm > pn^{1+1/p}$. □

Corollary: $p=1$: Bubblesort $\Omega(n^2)$ average case lower bound.
$p=2$: $n^{1.5}$ lower bound. $p=3$, $n^{4/3}$ lower bound
How can Kolmogorov Complexity help?

- We have presented at least a dozen of problems that were solved by the incompressibility methods. There are many more … such problems (these include the important switching lemma in circuit complexity as well quantum complexity bounds).

- But can Kolmogorov complexity help to prove higher bounds? Or it is limited to linear, nlogn, \( n^2 \) bounds?

- Can we import some probabilistic method tools?

- If such a tool simply does not work for certain things, like \( \text{NP} \neq \text{P} \), can we be certain about it? (prove this?)