Proof that $P \neq NP$

Author
Robert DiGregorio
0x51B4908DdCD986A41e2f8522BB5B68E563A358De

Abstract
Using a new tool called a “sorting key” it's possible to imply that $P \neq NP$.

Part 1

- Let $PS(x)$ be the unsorted power list (list of all subsets) of unsorted list of naturals $x$, with each subset folded over the sum operation, such that, given some natural $n$, $PS(x)[n]$ is the $n$th element of $PS(x)$
  - To clarify what "folded over the sum operation" means, here is the set $\{1, 2, 3\}$ folded over the sum operation in pseudocode: "$\{1, 2, 3\}.fold(sum) = 1 + 2 + 3 = 6$"
  - To clarify, $PS(x)$ is the unsorted list of all subset sums of $x$
  - To clarify, "sorted" means smaller naturals are always before larger naturals

- Let a "valid sorting key" be a natural such that, for some list $x$, for all natural $n$, $PS(x)[n \oplus (a valid sorting key of PS(x))]$ is $(sort \ PS(x))[n]$
  - Calculating a valid sorting key that sorts for all elements of $PS(x)$ is identical to sorting $PS(x)$. This is because $PS(x)[n]$ is the $n$th element of $PS(x)$, unsorted, and $PS(x)[n \oplus (a valid sorting key of PS(x))]$ is the $n$th element of $PS(x)$, sorted, so having a valid sorting key that sorts for all elements of $PS(x)$ means you have a sorted $PS(x)$
  - $\oplus$ is the bitwise exclusive or operation. If you apply $\oplus$ against some natural $x$ to every natural from 0 (inclusive) to $2^n$ (exclusive), those naturals are reordered such that every unique $x$ gives a unique order. As such, every power list has at least 1 “sorting key” that sorts it
  - If $KEY$ is the sorting key of some list $x$, reordering $x$ causes $KEY$ to become “invalid” and no longer sort $x$
  - If all elements of $PS(x)$ are unique, there is only 1 valid sorting key for $PS(x)$. Again, 1 valid sorting key sorts all elements of $PS(x)$

- Let $A$ be an unsorted list of naturals, given as input

- Let $KEY$ be a natural, given as input

- Let the decision problem be "given unsorted list $A$ as input and natural $KEY$ as input, is $KEY$ not a valid sorting key of $PS(A)$?"

- A deterministic polynomial time verifier can verify a YES solution to the decision problem if list $A$, natural $KEY$, natural $x$, and natural $y$ are given, such that $(x < y) \neq (PS(A)[x \oplus KEY] < PS(A)[y \oplus KEY])$
If a deterministic polynomial time verifier exists for a YES solution to a decision problem such that all deterministic Turing machines calculate it must run in superpolynomial time, \( P \neq NP \)

- If the decision problem can't be solved in polynomial time, \( P \neq NP \)
- If the decision problem can be solved in polynomial time, see part 2

**Part 2**

- It's implied that ALGORITHM exists such that ALGORITHM can determine if a sorting key is valid in polynomial time

- Let \( HIDE(x) \) be natural \( x \) transformed such that, for every natural \( n \), \( HIDE(x)[n] = x[2n \oplus (2n - 1)] \)
  - For example, \( HIDE(0011011_2) = 0110_2 \)

- Let \( M \) be some deterministic Turing machine such that \( M \) decides "given list \( A \) as input, given natural \( HIDE(KEY) \) as input, does a permutation \( A_p \) of \( A \) exist such that a possible value for KEY is a valid sorting key for \( PS(A_p) \)?"
  - There are \( O(2^{|A|}) \) possible values for \( KEY \)
  - There are \( O(|A|!) \) possible values for \( PS(A_p) \)
  - It is possible that only 1 possible KEY and is a valid sorting key for any possible \( PS(A_p) \)
  - It is possible that no possible KEY are a valid sorting key for any possible \( PS(A_p) \)

- Given \( A \) as input, \( A_p \) as input, and \( KEY \) as input, a verifier can verify \( A_p \) is a permutation of \( A \), then, using ALGORITHM, in polynomial time, verify \( KEY \) is a valid sorting key for \( PS(A_p) \)

- The search space is \( 2^{|A|/2} \) possible values for \( KEY \) and \( |A|! \) possible values for \( PS(A_p) \)

- Presume checking if a possible value for \( KEY \) is the valid sorting key for a possible value of \( PS(A_p) \) requires \( O(1) \) time
  - All possible values for \( KEY \) must be checked, because the only information contained in \( HIDE(KEY) \) is that \( KEY \) could be one of \( 2^{|A|/2} \) possible values
    - This forces the time complexity to be \( \geq O(2^{|A|}) \)
    - Even if you could binary search the search space, the time complexity would still be superpolynomial
  - This implies \( M \)'s decision problem, which can be verified in polynomial time, requires superpolynomial time to decide
    - This implies \( P \neq NP \)