https://en.wikipedia.org/wiki/Hierarchy

Class 13:
Circuit Size Hierarchy

University of Virginia
cs3120: DMT2
Wei-Kai Lin

Quick Recap

Theorem: Every circuit of size s can be written using $O(s \log s)$ bits.

Theorem: There are at most $2^{O(s \log s)}$ many circuits of size s

Corollary: at most $2^{O(s \log s)}$ many functions are in $SIZE(s)$

Corollary: at most $2^{O(s \log s)}$ many functions are in $SIZE(s)$

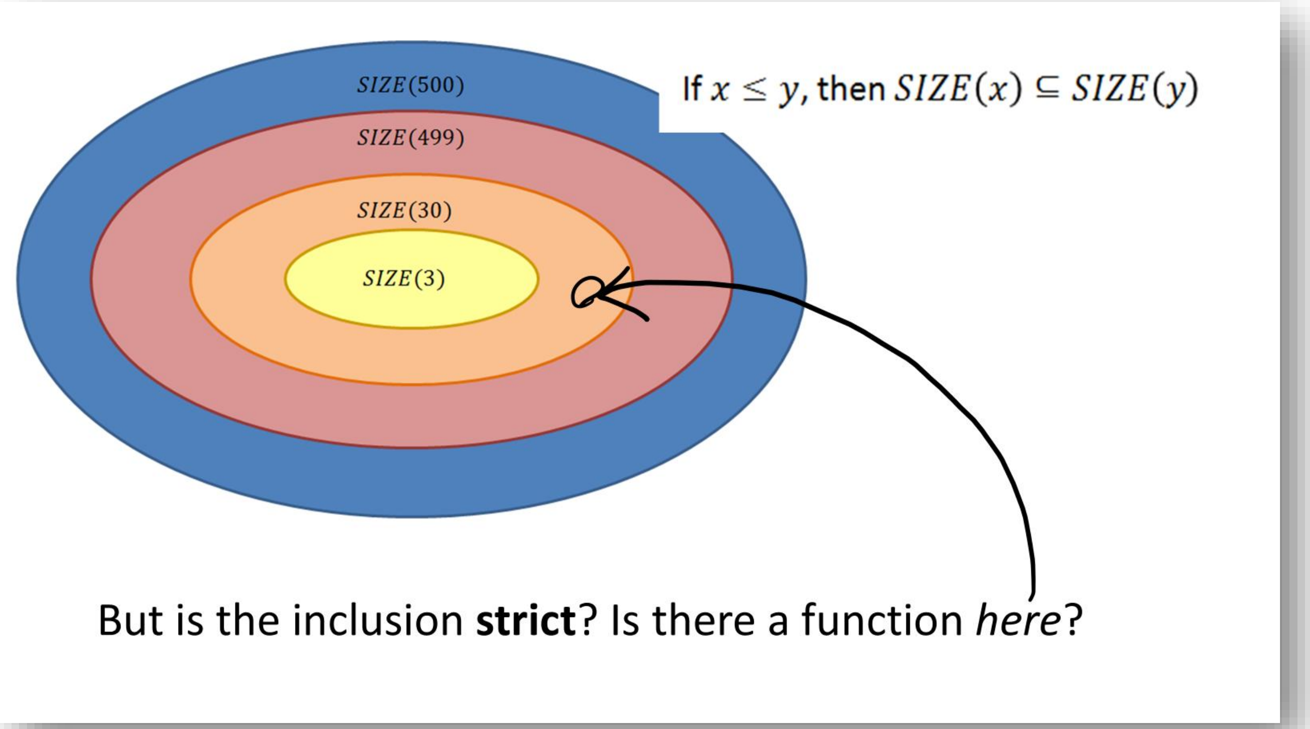
$|SIZE(s)| = 2^{O(s \log s)}$ **for all** s

$|SIZE(3)| \leq 2^{c \cdot 3 \log 3}$

and

$|SIZE(30)| \leq 2^{c \cdot 30 \log 30}$

Did we solve this?

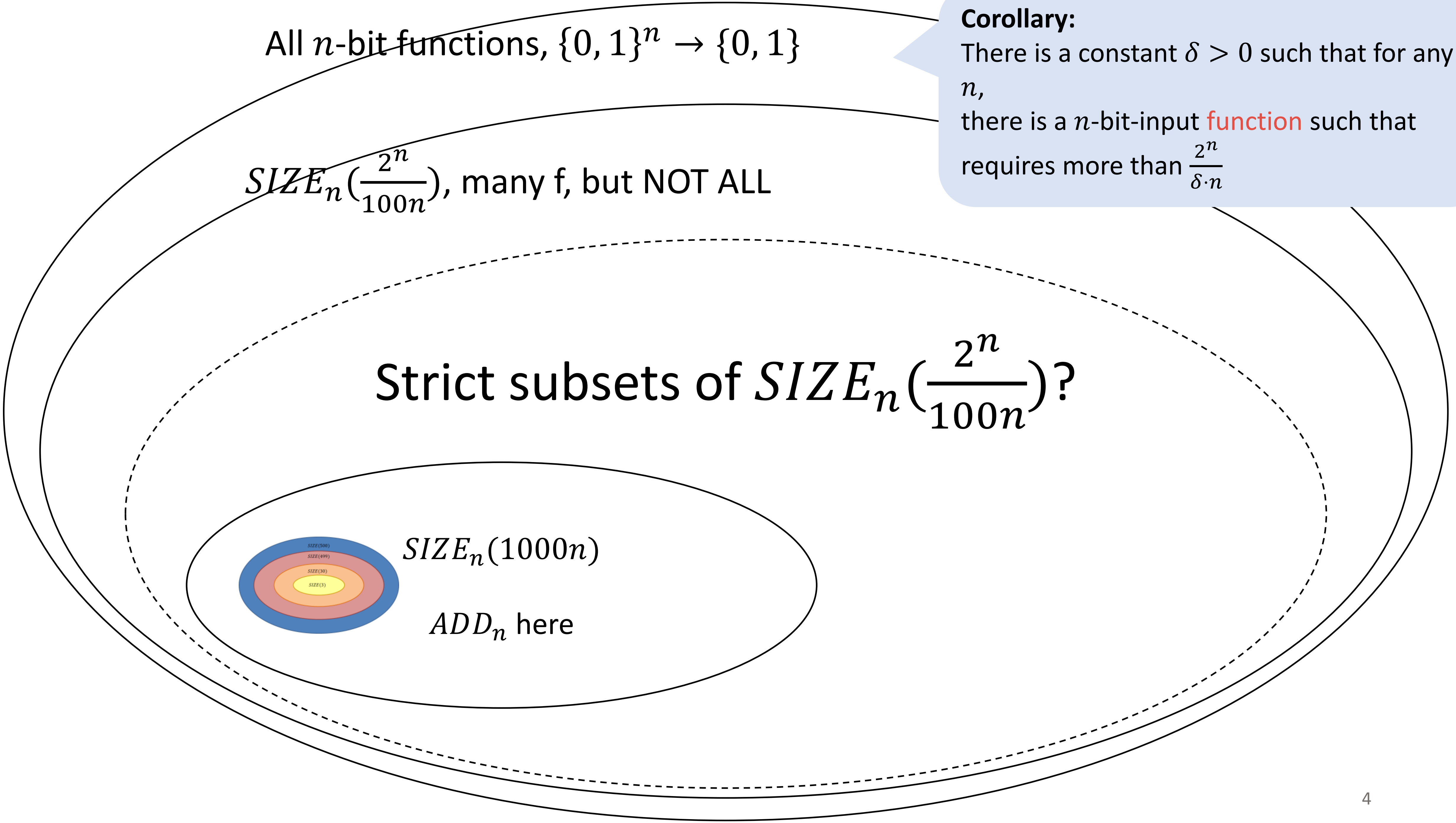


Corollary:
There is a constant $\delta > 0$ such that for any n, there is a n-bit-input function such that requires more than $\frac{2^n}{\delta \cdot n}$

Theorem 5.3 (Counting argument lower bound)

There is a constant $\delta > 0$, such that for every sufficiently large n, there is a function $f : \{0,1\}^n \to \{0,1\}$ such that $f \notin SIZE_n\left(\frac{\delta 2^n}{n}\right)$. That is, the shortest NAND-CIRC program to compute f requires more than $\delta \cdot 2^n/n$ lines.

$$SIZE_n\left(0.1\frac{2^n}{n}\right) \subsetneq SIZE_n\left(10\frac{2^n}{n}\right)$$



Corollary:
There is a constant $\delta > 0$ such that for any n,
there is a n-bit-input function such that requires more than $\frac{2^n}{\delta \cdot n}$

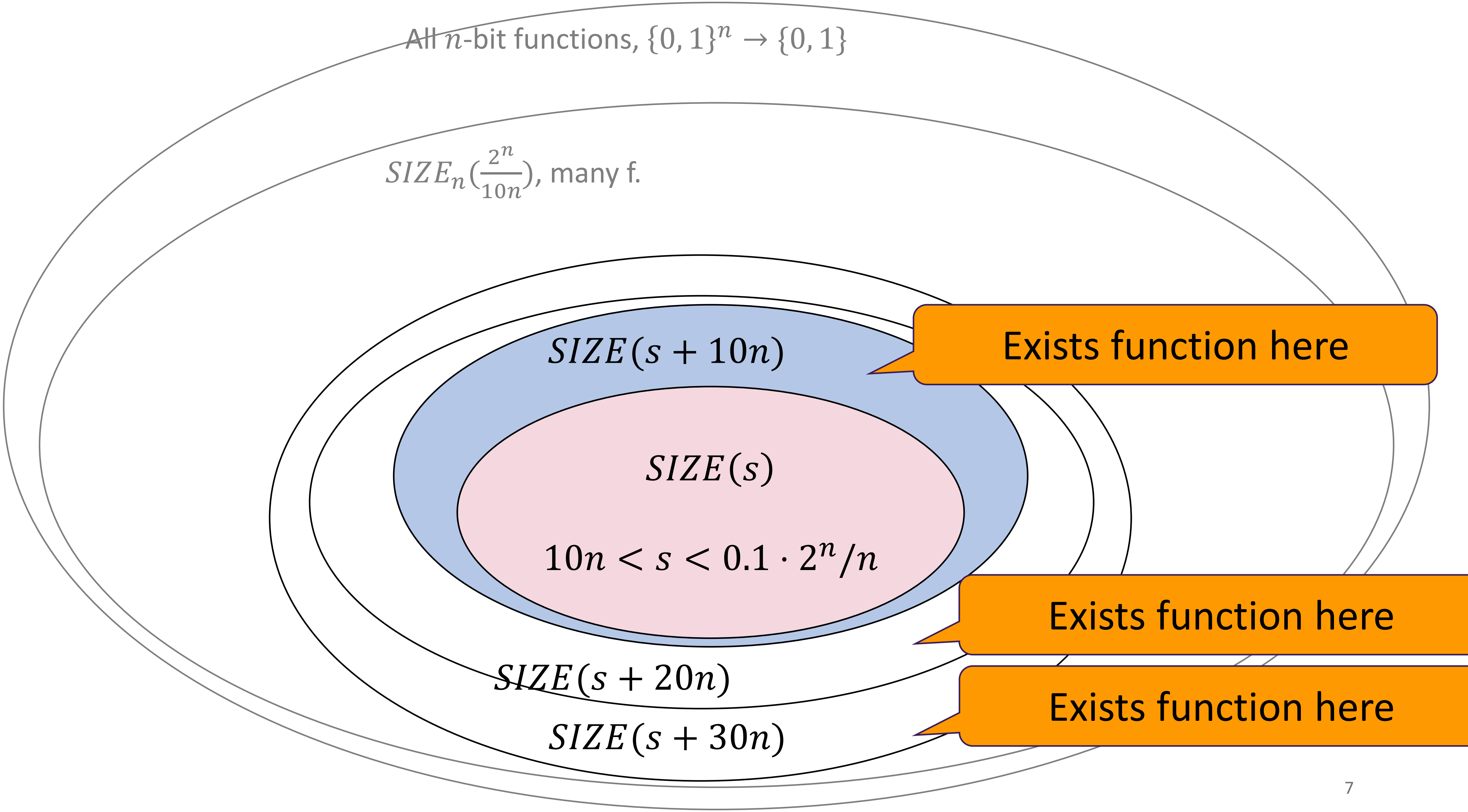
Size Hierarchy Theorem

Size Hierarchy Theorem

Theorem 5.5 (Size Hierarchy Theorem)

For every sufficiently large n *and* $10n < s < 0.1 \cdot 2^n/n$,

$$SIZE_n(s) \subsetneq SIZE_n(s + 10n) \ .$$



All n-bit functions, $\{0,1\}^n \to \{0,1\}$
$SIZE_n(\frac{2^n}{10n})$, many f.
$SIZE(s+10n)$
$SIZE(s)$
$10n < s < 0.1 \cdot 2^n/n$
$SIZE(s+20n)$
$SIZE(s+30n)$
Exists function here
Exists function here
Exists function here

Proof of Size Hierarchy

Proof idea

Theorem 5.5 (Size Hierarchy Theorem)

For every sufficiently large n and $10n < s < 0.1 \cdot 2^n/n$,

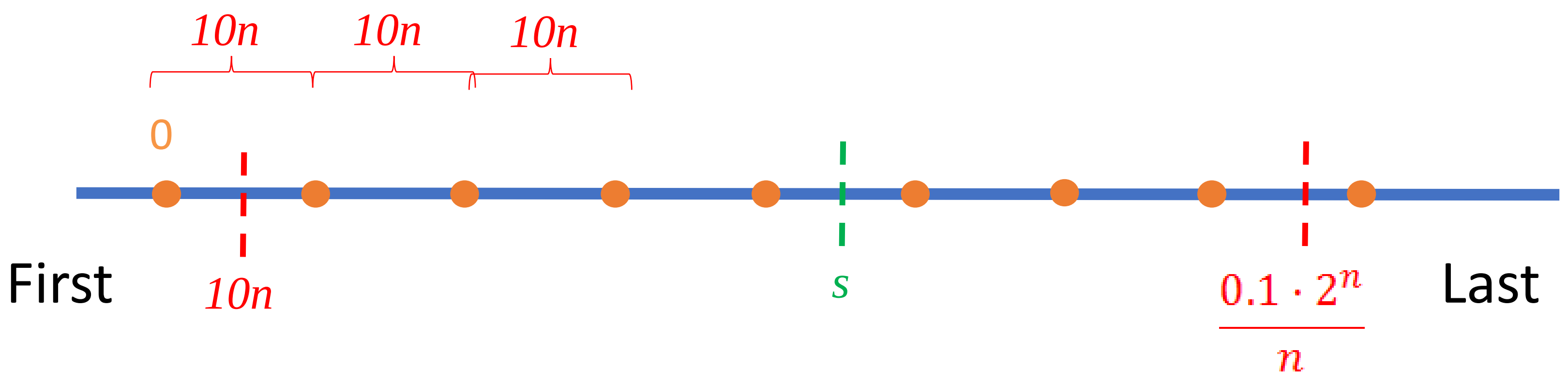
$$SIZE_n(s) \subsetneq SIZE_n(s + 10n) \; .$$

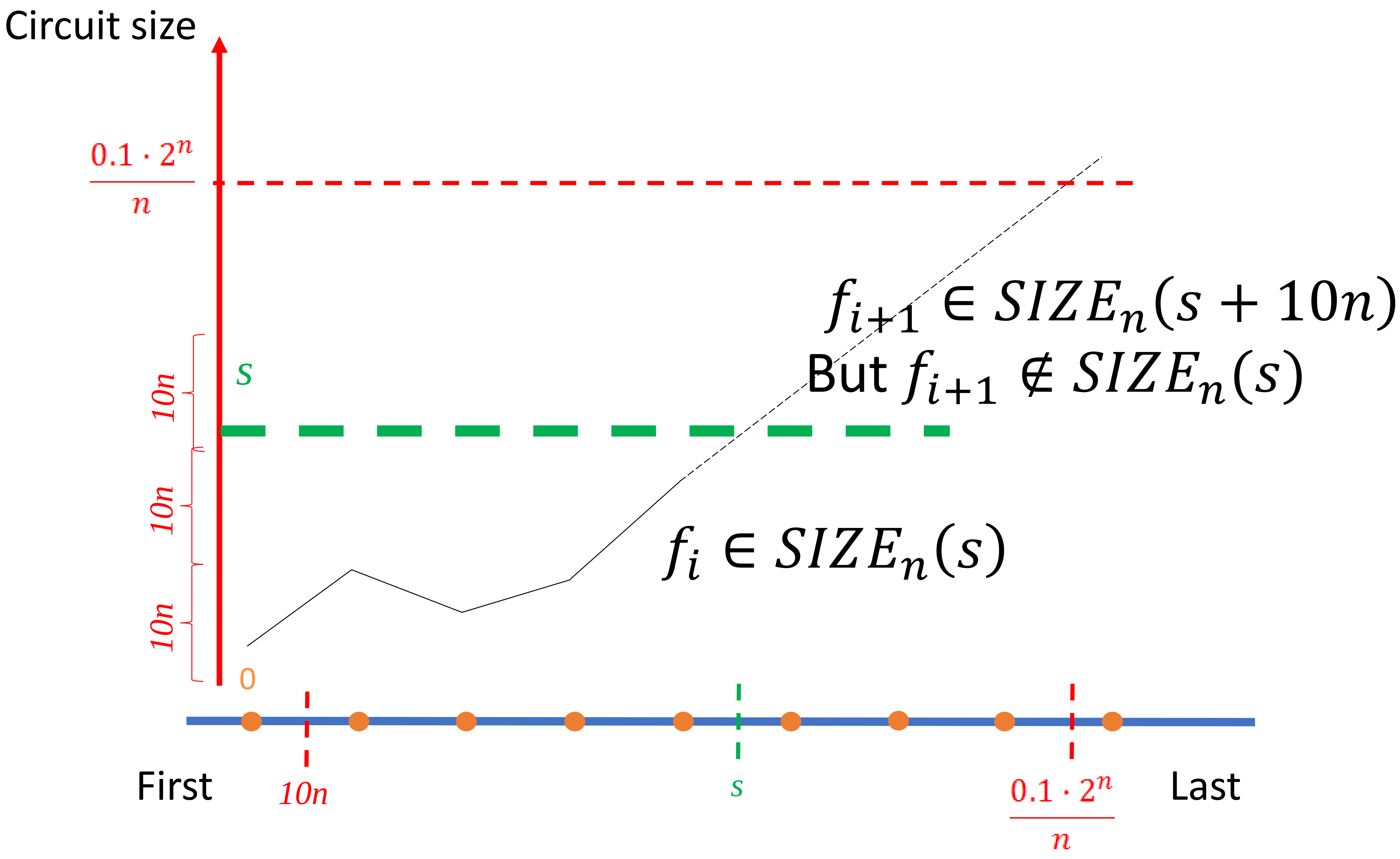
Find a sequence of functions such that:

1. First function **can** be computed using $\leq 10n$ gates.
2. Last function **cannot** be computed by $\frac{0.1 \cdot 2^n}{n}$ gates.
3. For all functions in the sequence, if function i can be computed using t gates, then the function $i + 1$ can be computed using $t + 10n$ gates.

Find a sequence of functions such that:

1. First function **can** be computed using $\leq 10n$ gates.
2. Last function **cannot** be computed by $\frac{0.1 \cdot 2^n}{n}$ gates.
3. For all functions in the sequence, if function i can be computed using t gates, then the function $i+1$ can be computed using $t + 10n$ gates.




Circuit size
$\frac{0.1 \cdot 2^n}{n}$
s
$10n$
$10n$
$10n$
0
$f_{i+1} \in SIZE_n(s + 10n)$
But $f_{i+1} \notin SIZE_n(s)$
$f_i \in SIZE_n(s)$
First
$10n$
s
$\frac{0.1 \cdot 2^n}{n}$
Last

What sequence of functions works?

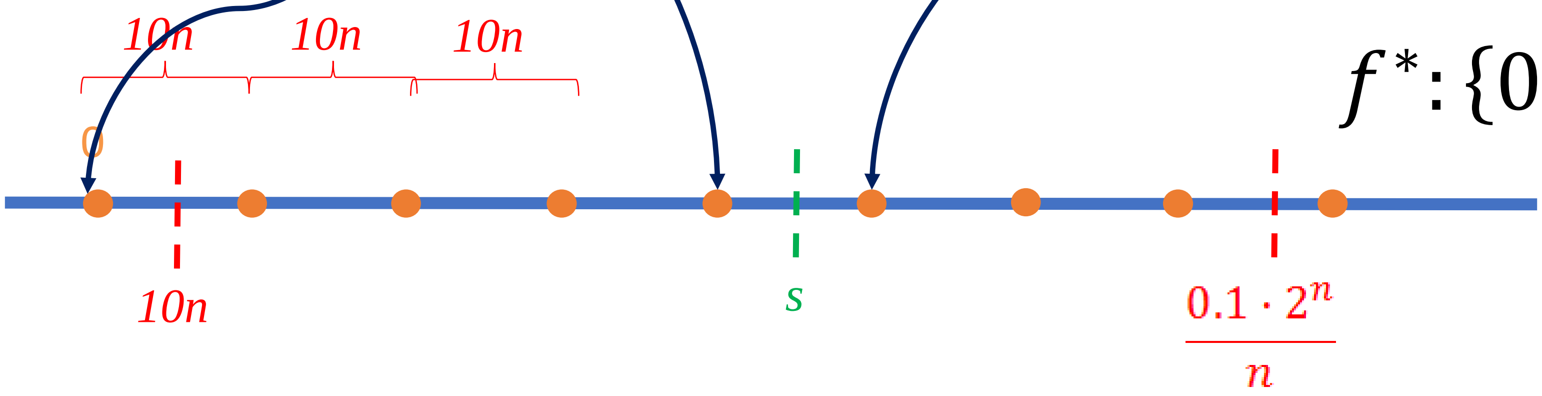
$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(t)$

$f_{i+1}: \{0,1\}^n \to \{0,1\} \in SIZE_n(t + 10n)$

$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$

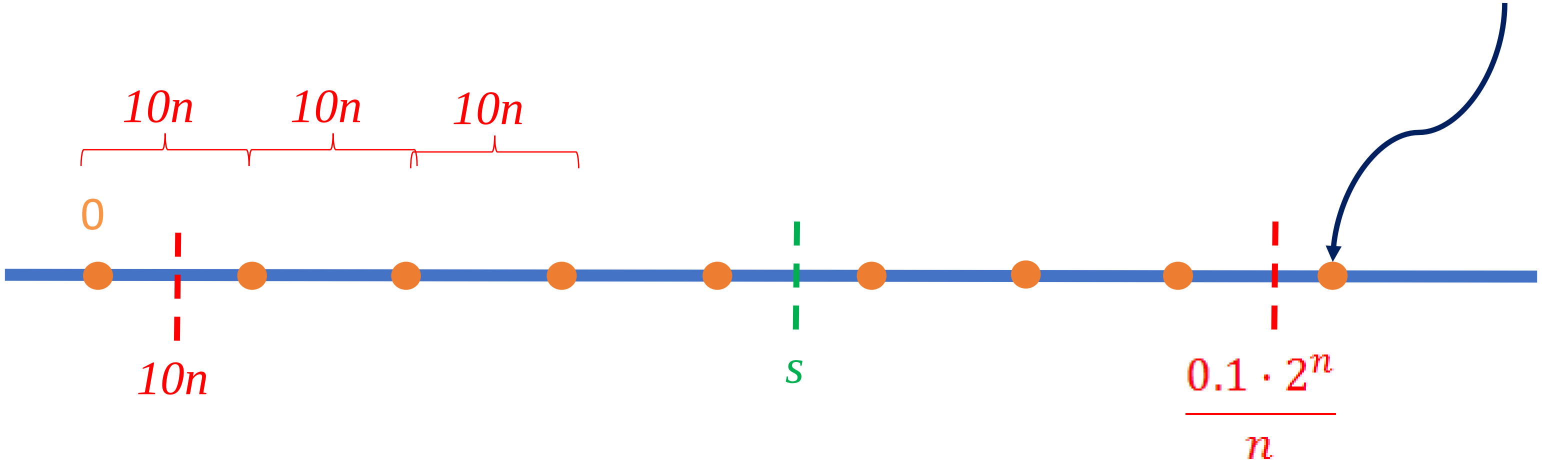
$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$

$f^*: \{0,1\}^n \to \{0,1\}$



How do we know f^* exists?

$$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$$



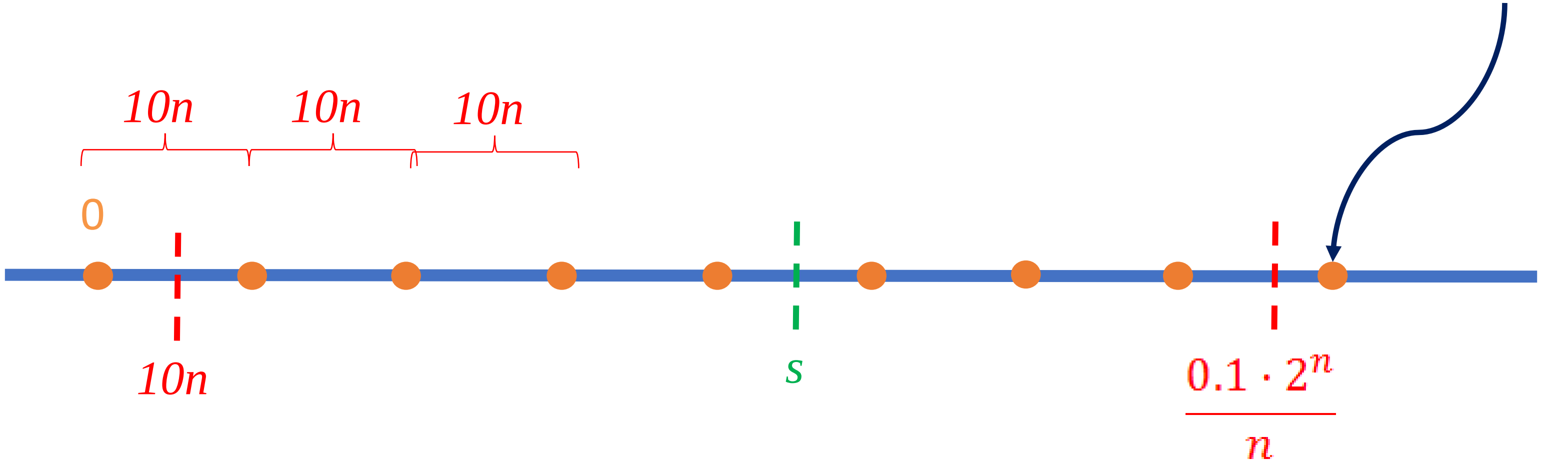
How do we know f^* exists?

Theorem 5.3 (Counting argument lower bound)

There is a constant $\delta > 0$, such that for every sufficiently large n, there is a function $f : \{0,1\}^n \to \{0,1\}$ such that $f \notin SIZE_n\left(\frac{\delta 2^n}{n}\right)$. That is, the shortest NAND-CIRC program to compute f requires more than $\delta \cdot 2^n/n$ lines.

The constant δ is at least 0.1 and in fact, can be improved to be arbitrarily close to $1/2$, see Exercise 5.7.

$$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$$



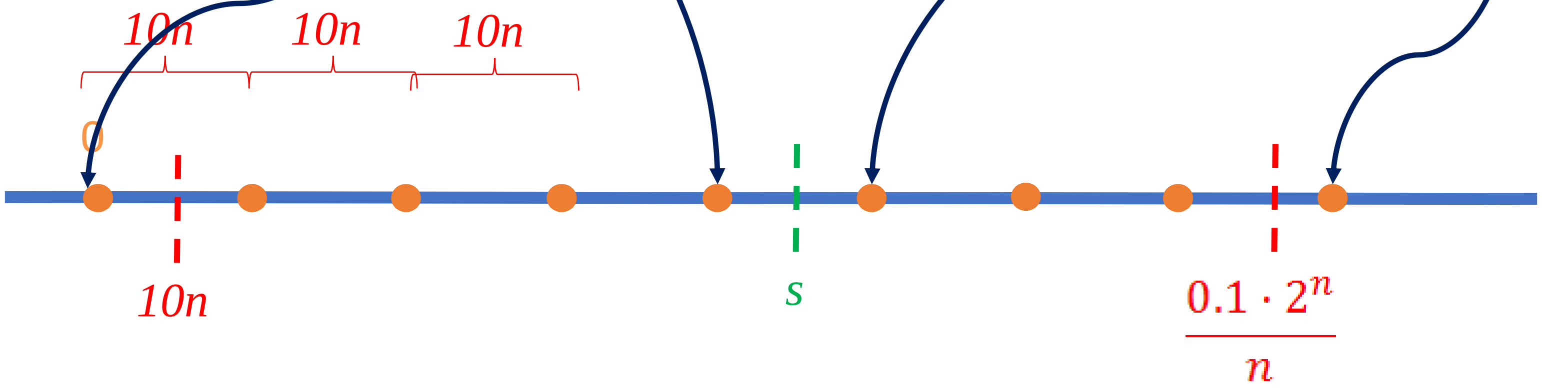
What sequence of functions works?

$$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(t)$$

$$f_{i+1}: \{0,1\}^n \to \{0,1\} \in SIZE_n(t + 10n$$

$$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$$

$$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$$

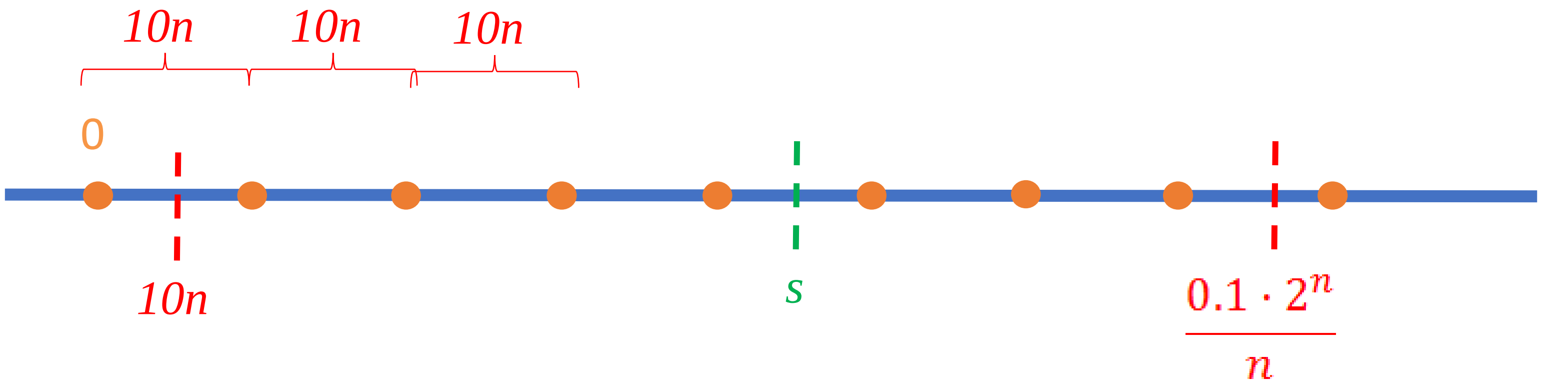


Idea: make function f_i easy for all inputs $> i$

So f_{i+1} is not hugely harder than f_i

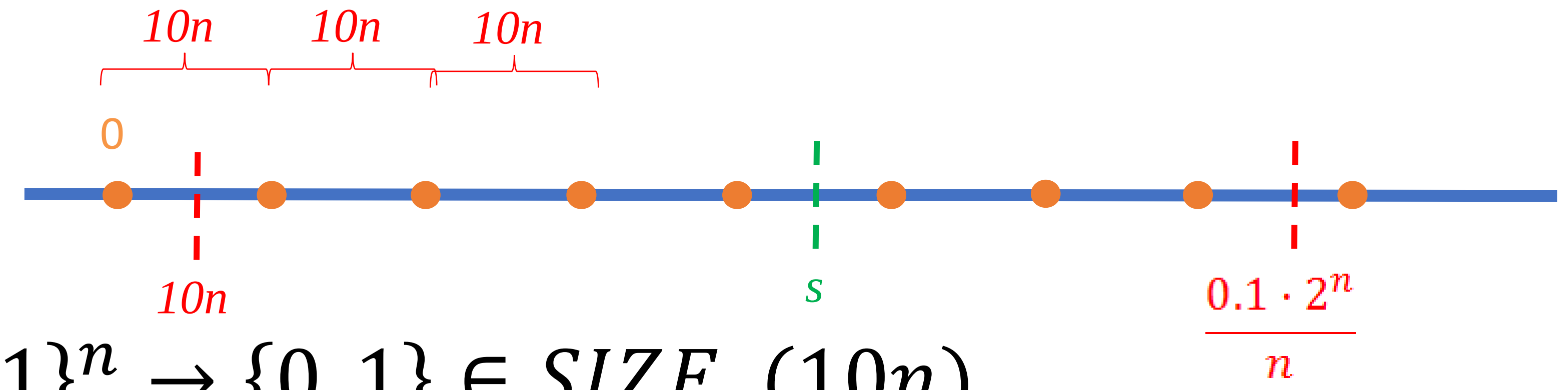
$$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(s)$$

$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$



Does f_0 work?

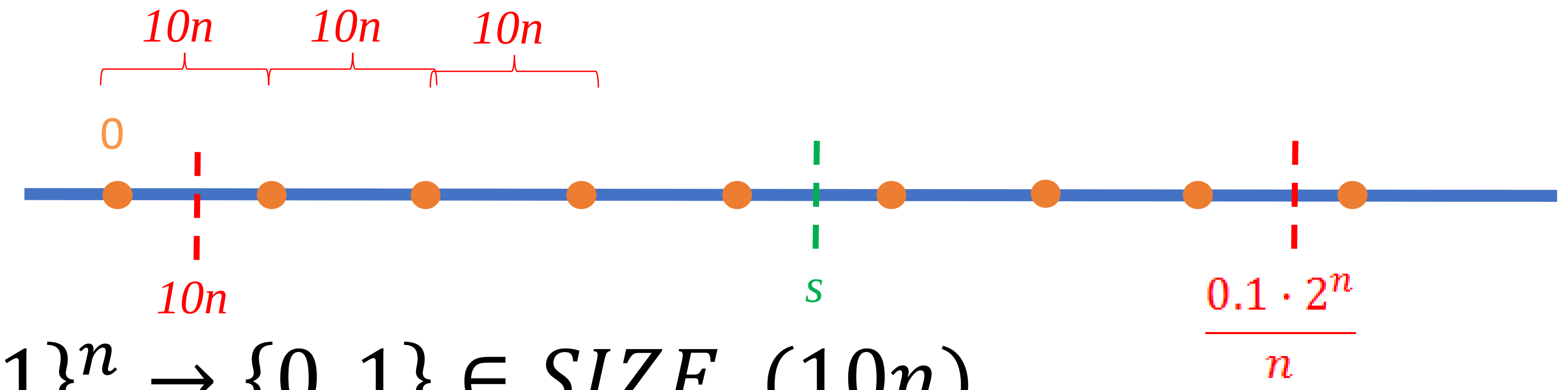
$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$



$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$

Does f_{2^n} work?

$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$

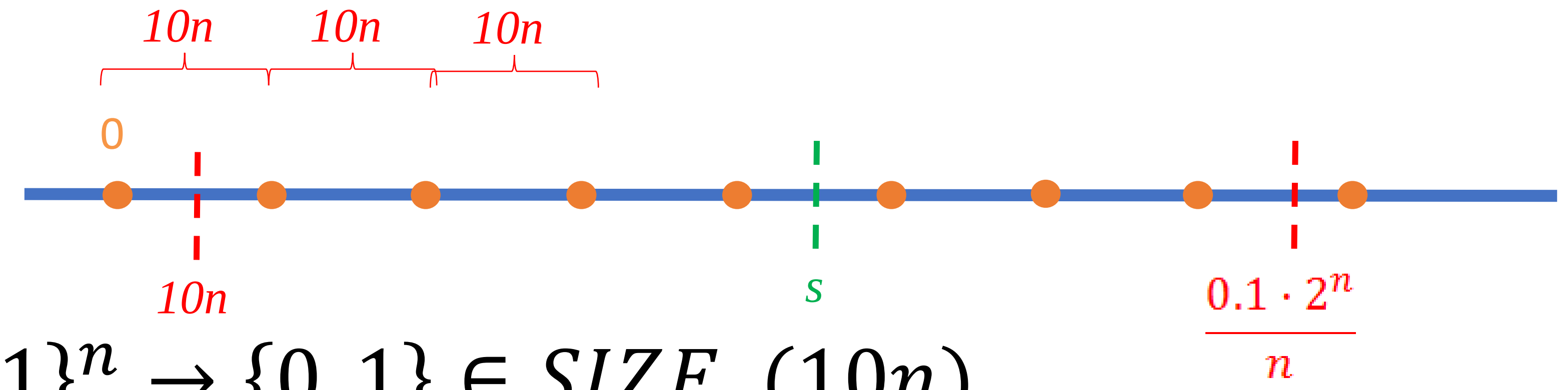


$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$

Does f_{2^n} work?

$$f_{2^n}(x) = f^*(x)$$

$$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$$



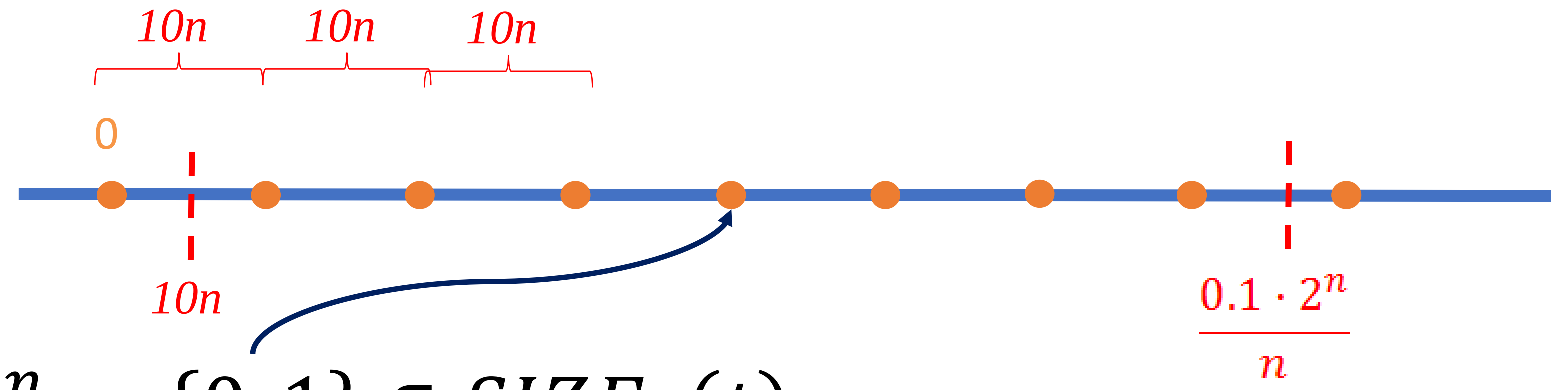
$$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$$

Inductive Step: $f_i \to f_{i+1}$

3. For all functions in the sequence, if function i can be computed using t gates, then the function $i+1$ can be computed using $t + 10n$ gates.

$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$

$$f_{i+1}(x) =$$



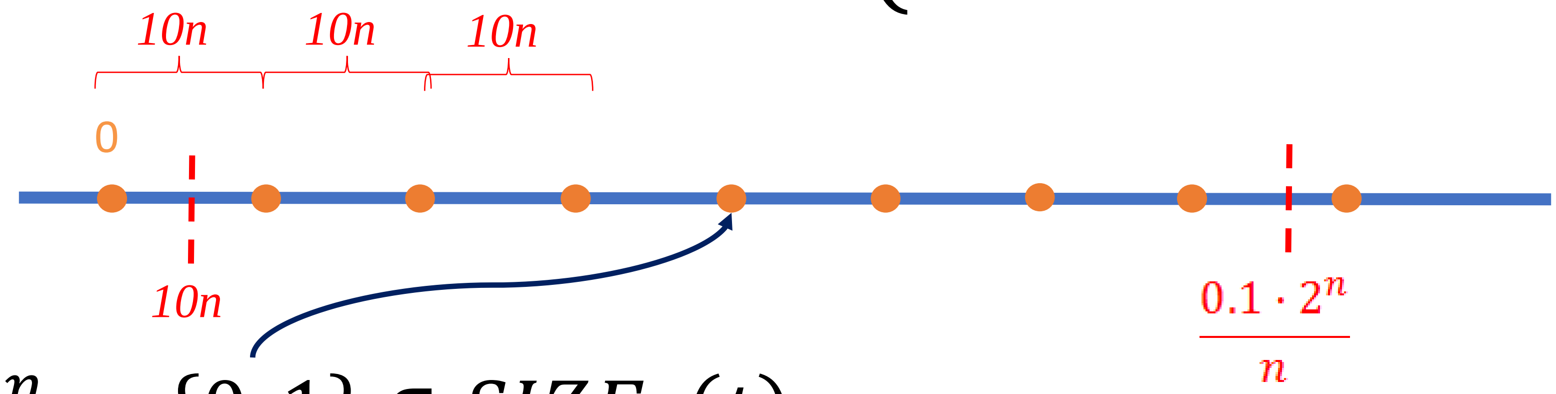
$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(t)$

Inductive Step: $f_i \rightarrow f_{i+1}$

3. For all functions in the sequence, if function i can be computed using t gates, then the function $i+1$ can be computed using $t + 10n$ gates.

$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$

$$f_{i+1}(x) = \begin{cases} f^*(x) & \text{for the } i^{th} \text{ input} \\ f_i(x) & \text{for all other inputs} \end{cases}$$



$f_i: \{0,1\}^n \rightarrow \{0,1\} \in SIZE_n(t)$

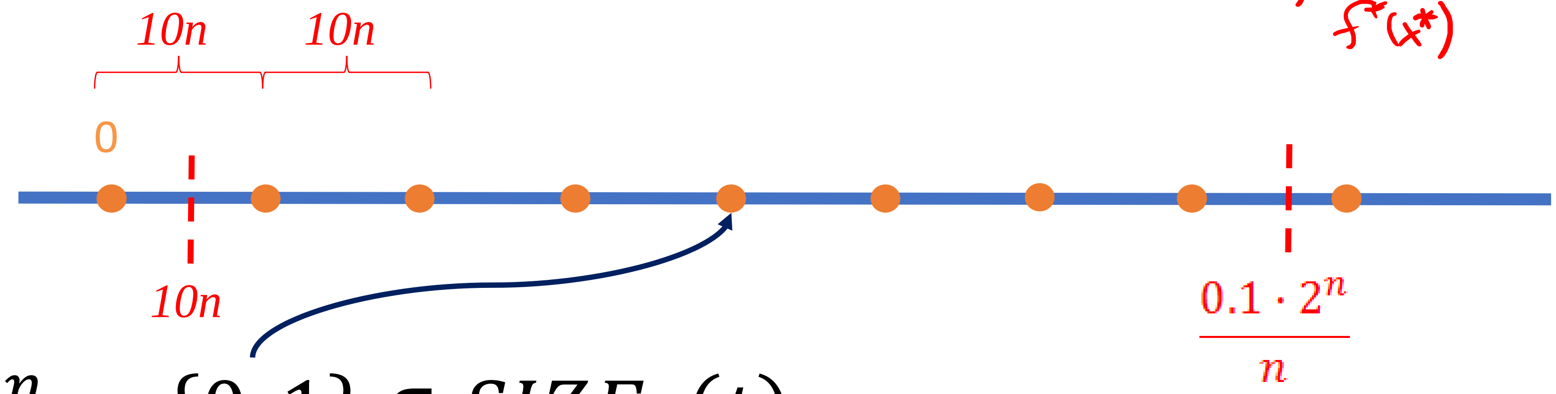
Implementing f_{i+1} in $SIZE_n(t + 10n)$

3. For all functions in the sequence, if function i can be computed using t gates, then the function $i + 1$ can be computed using $t + 10n$ gates.

$$f_{i+1}(x) = \begin{cases} f^*(x) & \text{for the } i^{th} \text{ input} \\ f_i(x) & \text{for all other inputs} \end{cases}$$

$$f_{i+1} = IF\big(EQUAL(x, x^*), f^*(x), f_i(x)\big)$$

$C_{i+1}(x) = IF(EQUAL(x, x^*), f^*(x^*), C_i(x))$



$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(t)$

Ordering the Inputs

$lex(x) \in \{0, 1, \ldots, 2^n\}$ is defined as the position of x in an ordered sequence of all n-bit values

$$f_i(x) = \begin{cases} f^*(x), & lex(x) < i \\ 0, & \text{otherwise} \end{cases}$$

Function	$lex(x) = 0$	$lex(x) = 1$	…	$lex(x) = i$	$lex(x) = i + 1$	…	$lex(x) = 2^n - 1$
$f_0(x)$	0	0	…	0	0	….	0
$f_1(x)$	$f^*(x)$	0	…	0	0	…	0
…	…	…	…	…	…	…	…
$f_i(x)$	$f^*(x)$	$f^*(x)$	…	0	0	…	0
$f_{i+1}(x)$	$f^*(x)$	$f^*(x)$	…	$f^*(x)$	0	…	0
…	…	…	…	…	…	…	…
$f_{2^n}(x)$	$f^*(x)$	$f^*(x)$	…	$f^*(x)$	$f^*(x)$	…	$f^*(x)$

Completing the Proof

Theorem 5.5 (Size Hierarchy Theorem)

For every sufficiently large n and $10n < s < 0.1 \cdot 2^n/n$,

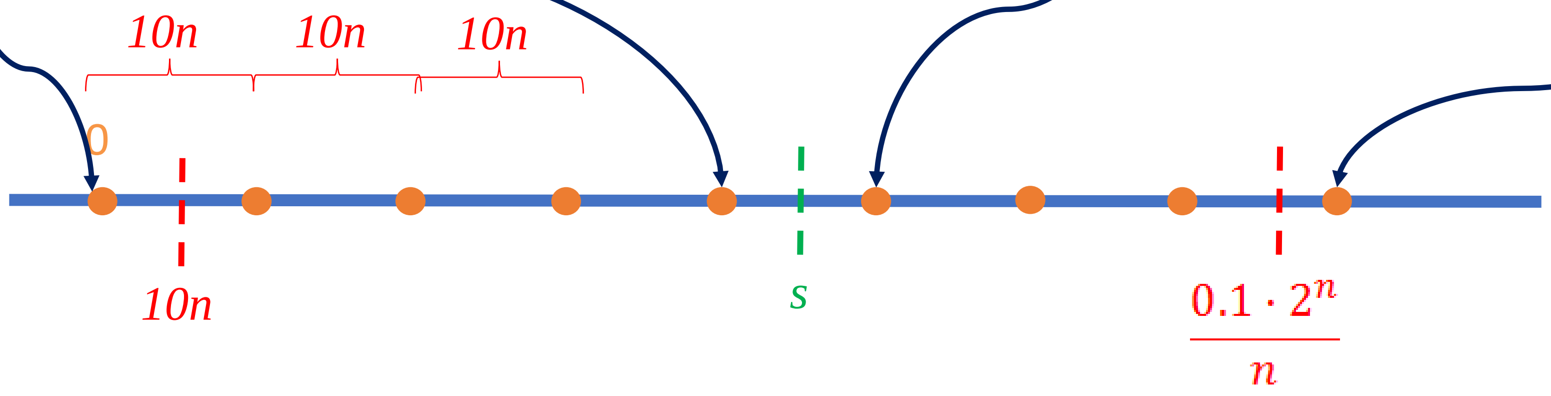
$$SIZE_n(s) \subsetneq SIZE_n(s + 10n) \ .$$

$f_i \in SIZE_n(s)$

$f_{i+1} \in SIZE_n(s + 10n)$
But $f_{i+1} \notin SIZE_n(s)$

$f_0 \in SIZE_n(10n)$

$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$



If s is between $10n$ and $0.1 \cdot \frac{2^n}{n}$ then there are functions on both sides of s.

HW 5 due after Spring break

Quiz 6 due Monday, Mar 9

Class 13: Circuit size hierarchy

University of Virginia
CS3120: DMT2
https://weikailin.github.io/cs3120-toc
Wei-Kai Lin

Plan

Circuit-size hierarchy theorem

Proof

Implication

Textbook [TCS] Section 3 and 4

https://introtcs.org/public/lec_04_code_and_data.html

Code as data, data as code

Proof of Circuit Size Hierarchy

$$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$$

$$f_i(x) = \begin{cases} f^*(x) & \text{for the first } i \text{ inputs, } i \in [2^n] \\ 0 & \text{for all other inputs} \end{cases}$$

This was an *existential* proof (annoying?)

Our proof showed $f_j \in SIZE(s + 10n) \setminus SIZE(s)$ **exists**

We did not "explicitly show" what function f_j we are dealing with

Root cause: we did not construct function f^* to begin with

Even if we did know f^*, it is not easy to identify the value of j

How about this existential proof?

Theorem: there is an irrational real number

Proof: $\sqrt{2}$ is irrational….

Theorem: There are irrational numbers x, y where x^y is rational.

Proof: First let $x = \sqrt{2}$ and $y = \sqrt{2}$. If x^y is rational, we are done, and if not: then let $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$, and we have $x^y = 2$.

irrational ? irrational

$\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2})^2} = (\sqrt{2})^2$

The proof does not tell us which pair is the one we want!

Any "constructive" proof of Size Hierarchy?

Is this a constructive description?

Describe a simple function (in English or math?) that provably has circuit complexity (i.e., necessary number of gates) at least ~~$2^{\Omega(n)}$~~

$\Omega(n^2)$

A candidate function (open to prove circuit lower bond):

Given input of length n, interpret it as a graph G, and then output 1 if G is 3-colorable

Since most functions have large circuits, it is like: "finding hay in haystack".

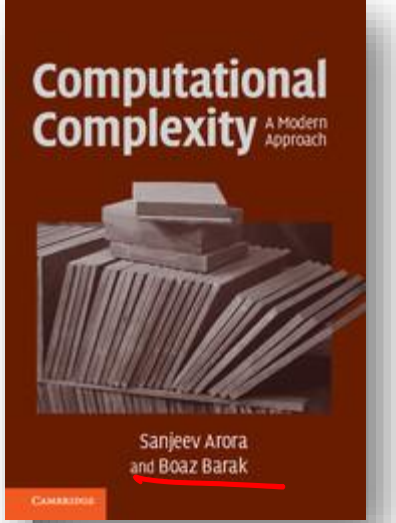
Chapter 14

Circuit lowerbounds

Complexity theory's Waterloo

We believe that **NP** does not have polynomial-sized circuits. We've seen that if true, this implies that **NP** $\neq$ **P**. In the 1970s and 1980s, many researchers came to believe that the route to resolving **P** versus **NP** should go via circuit lowerbounds, since circuits seem easier to reason about than Turing machines. The success in this endeavor was mixed.

Progress on general circuits has been almost nonexistent: a lowerbound of n is trivial for any function that depends on all its input bits. We are unable to prove even a superlinear circuit lowerbound for any **NP** problem—the best we can do after years of effort is $4.5n - o(n)$.



"Complexity theory's Waterloo"

...

"We are unable to prove even a superlinear circuit lowerbound for any NP problem—the best we can do after years of effort is $4.5n - o(n)$."

Plan

Circuit size hierarchy

Proof

[TCS] Textbook, Section 5.2

https://introtcs.org/public/lec_04_code_and_data.html#size-hierarchy-theorem-optional

Next module: Turing machine and computability

HW 5 coming soon, due after Spring break

Quiz 6 due Monday, Mar 9