Problem Set 4 is due
This Friday, Feb 14 (10pm)

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

Class 10: *Circuit Size Hierarchy*

University of Virginia
cs3120: DMT2
Wei-Kai Lin

Recap: Circuit Size of n-bit Functions

- There are 2^{2^n} Boolean functions $\{0,1\}^n \to \{0,1\}$
- There are **at most** $2^{O(s\ \log s)}$ circuits of size s
- There are **at most** $2^{O(s\ \log s)}$ *functions* of size s:

$$|SIZE(s)| \leq 2^{O(s \log s)}$$

$2^{c \cdot s \log s}$

n-bit

So, $SIZE_n(s)$ cannot contain all functions $\{0,1\}^n \to \{0,1\}$

If $s = \frac{2^n}{10n}$, then $|SIZE(s)| \leq 2^{c \cdot s \log s} < 2^{2^n}$

Checking the Definitions

$SIZE(s)$ is defined as the set of all **functions** that can be implemented by a circuit of at most s NAND gates

$SIZE_n(s)$ is defined as the set of all **functions** $\{0,1\}^n \to \{0,1\}$ that can be implemented by a circuit of at most s NAND gates

What is the relationship between $SIZE(3)$ and $SIZE_2(3)$?

$\supseteq$

All n-bit functions, $\{0,1\}^n \to \{0,1\}$

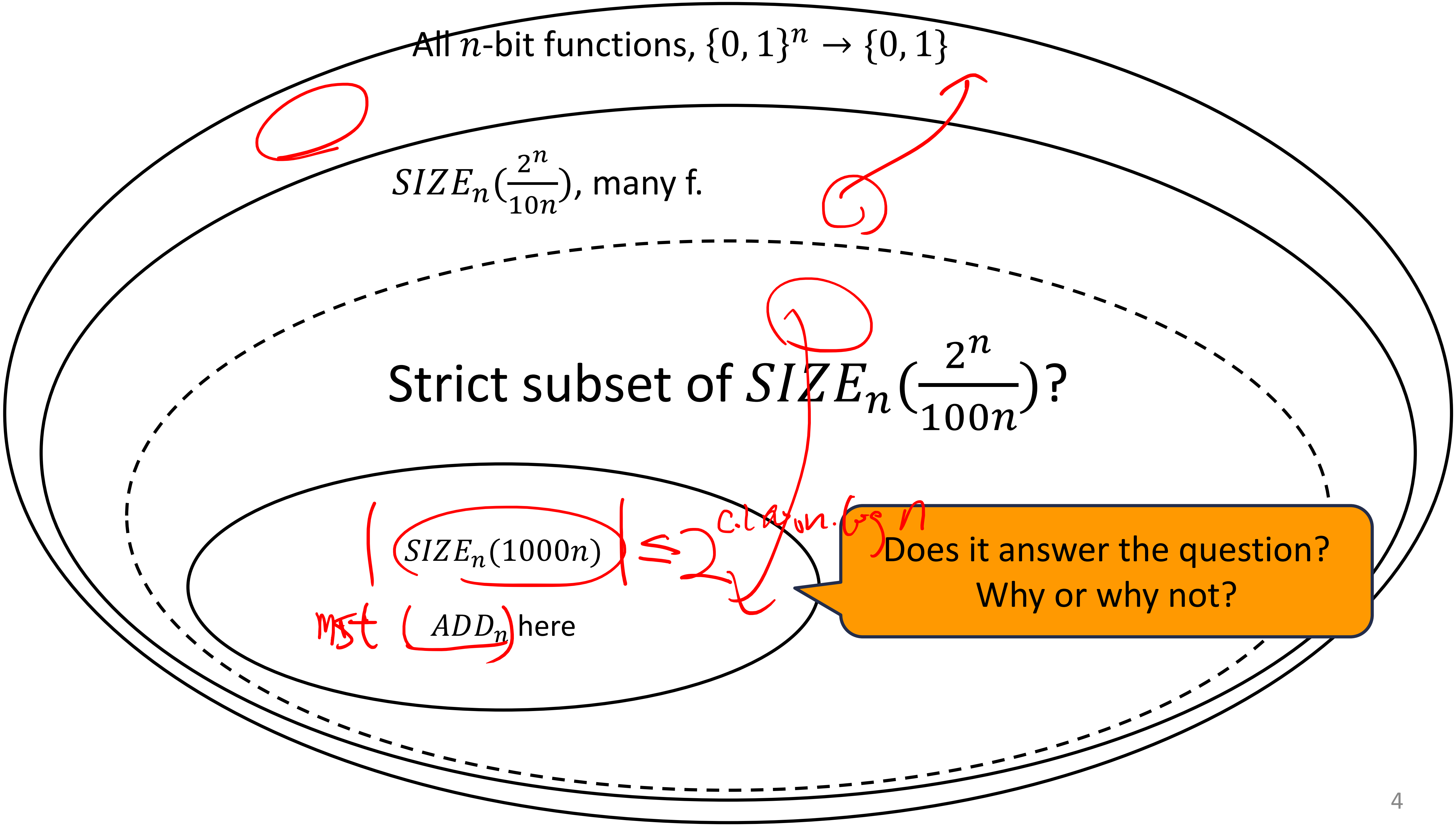
Exists f here

2^{2^n}

$SIZE_n(\frac{2^n}{10n})$, many f. **Strict subset** of all functions

Strict subset of $SIZE_n(\frac{2^n}{100n})$?

Main Question today!


All n-bit functions, $\{0,1\}^n \rightarrow \{0,1\}$
$SIZE_n(\frac{2^n}{10n})$, many f.
Strict subset of $SIZE_n(\frac{2^n}{100n})$?
$|SIZE_n(1000n)| \leq 2^{c \cdot 1000n \cdot \log n}$
not ADD_n here
Does it answer the question?
Why or why not?

Circuit Size Hierarchy

Size Hierarchy Theorem

Theorem 5.5 (Size Hierarchy Theorem)

$\frac{1}{2}$

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

2

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

Interpret the statement:

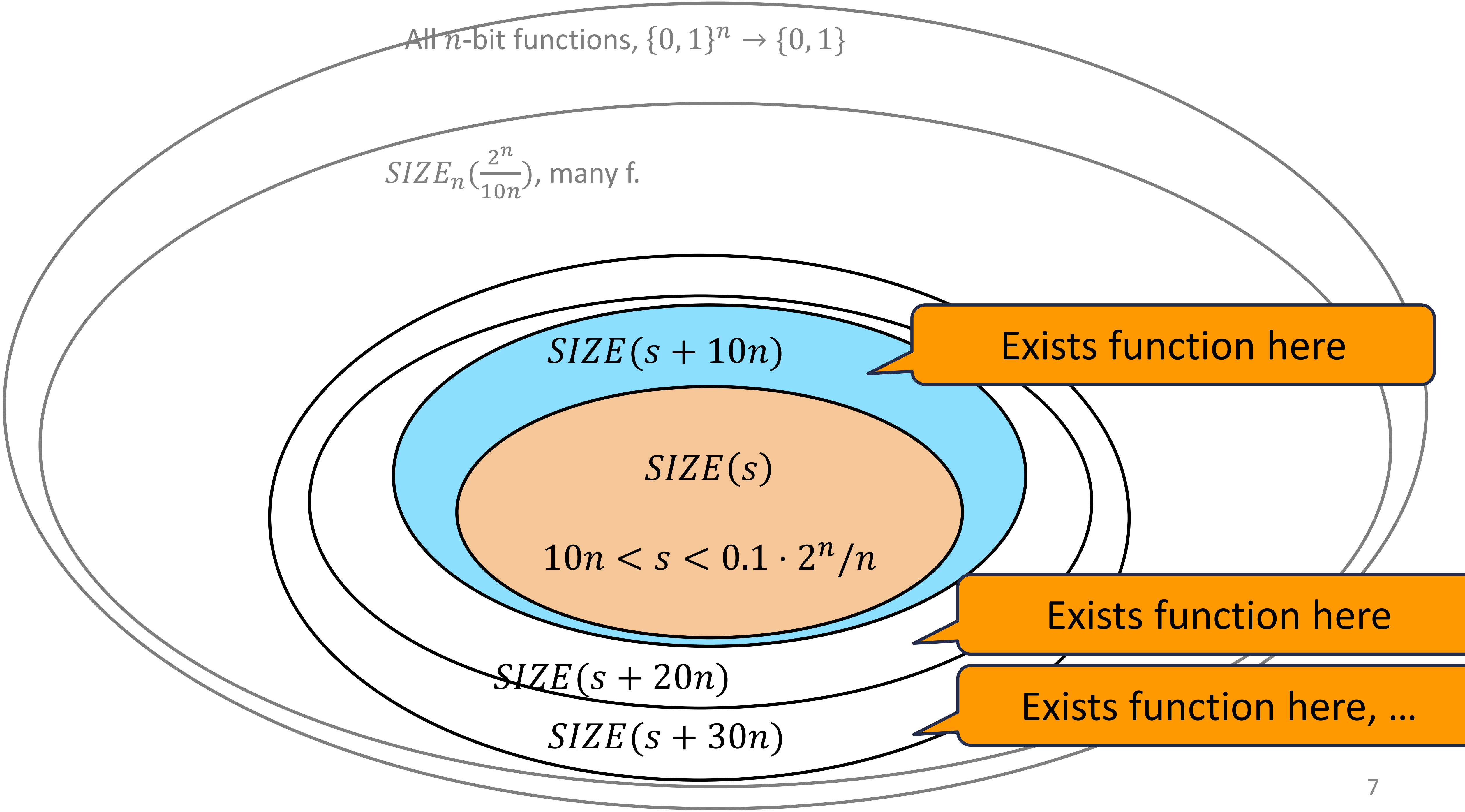
given $n = \cancel{10}\ 20$

$\cancel{100}$ 200 $< s < 0.1 \cdot 1024/10$

$0.1 \quad 1024^2/10 \approx 100000$

$s = 3120$

$s + 10n = 5120$


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

Theorem 5.5 (Size Hierarchy Theorem)

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

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

Proof idea

Find a sequence of functions such that:

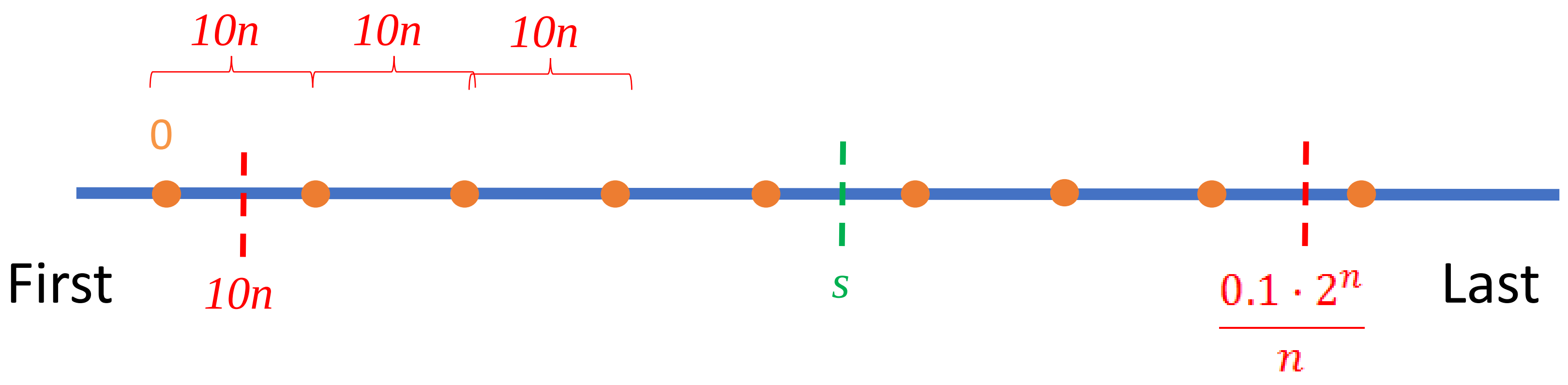
1. First function **can** be computed using $\leq 10n$ gates.
2. Last function **cannot** be computed by $\frac{0.1 \cdot 2^n}{n}$ gates.
3. For all functions in the sequence, if function i can be computed using t gates, then the function $i + 1$ can be computed using $t + 10n$ gates.

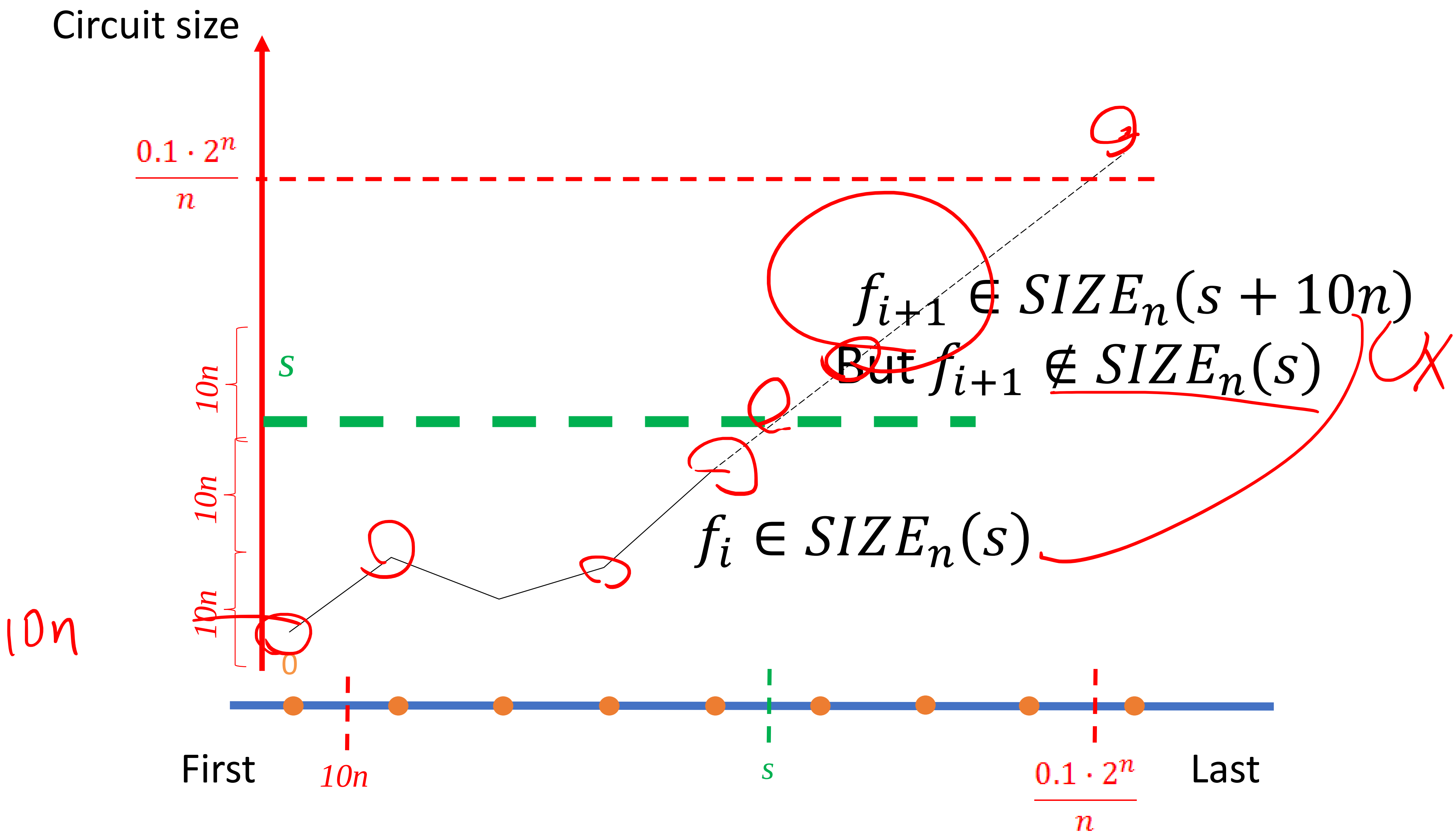
First easy

Last hard

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
$10n$
$10n$
$10n$
$10n$
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?

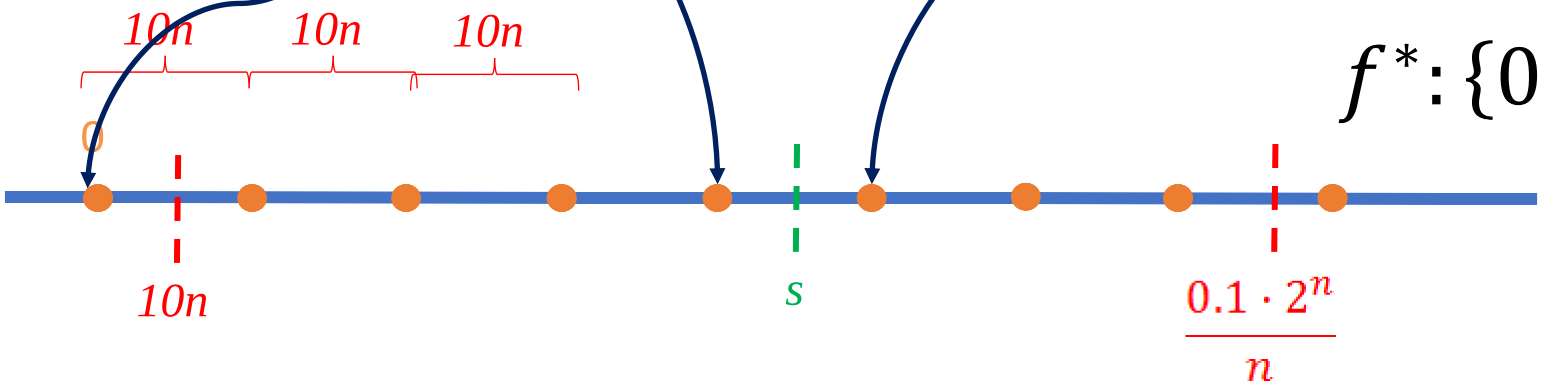
$f_i: \{0,1\}^n \to \{0,1\} \in SIZE_n(t)$

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

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

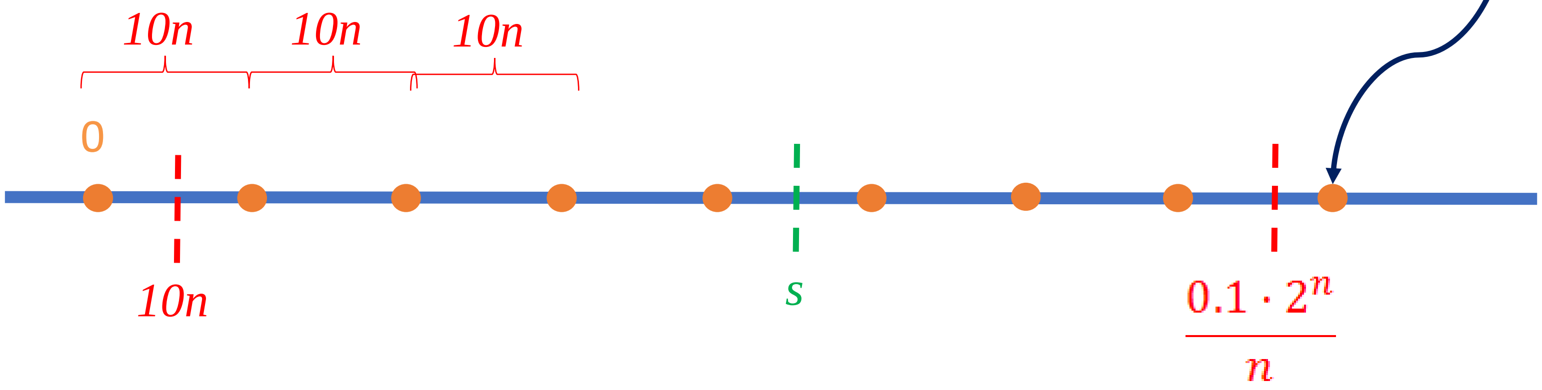
$f^* \notin SIZE_n\left(0.1 \cdot \frac{2^n}{n}\right)$

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



How do we know f^* exists?

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



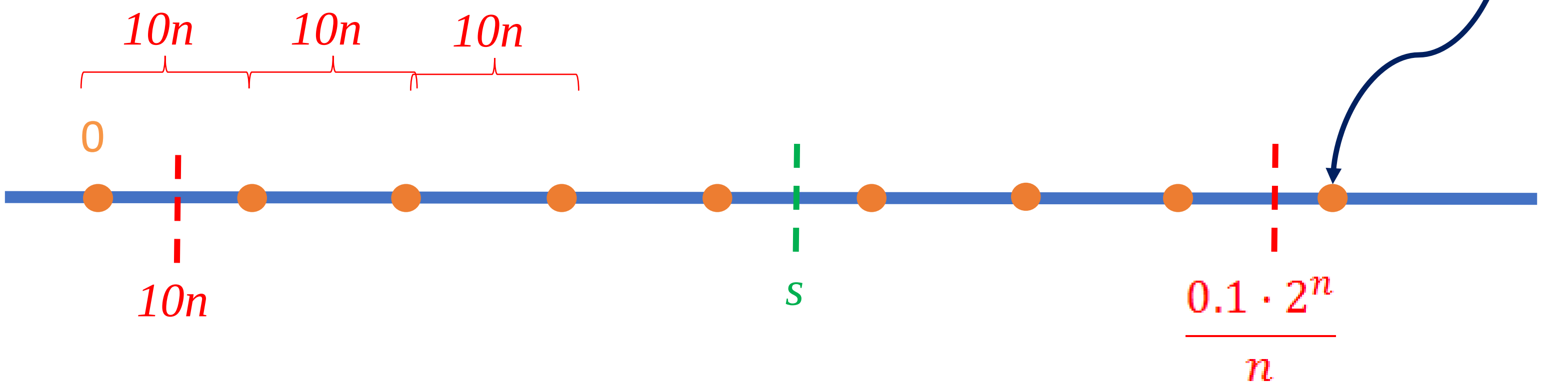
How do we know f^* exists?

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.

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



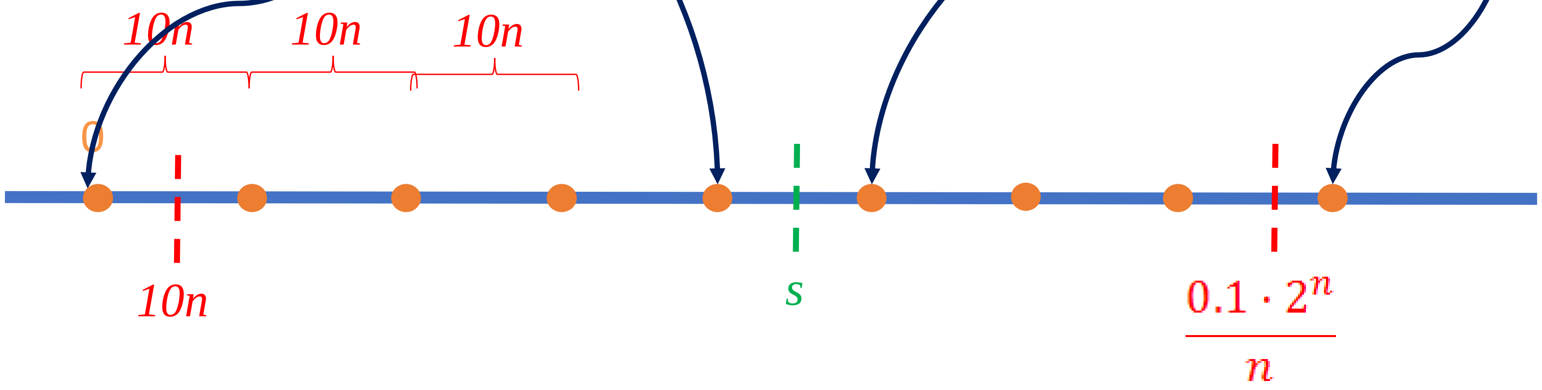
What sequence of functions works?

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

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

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

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

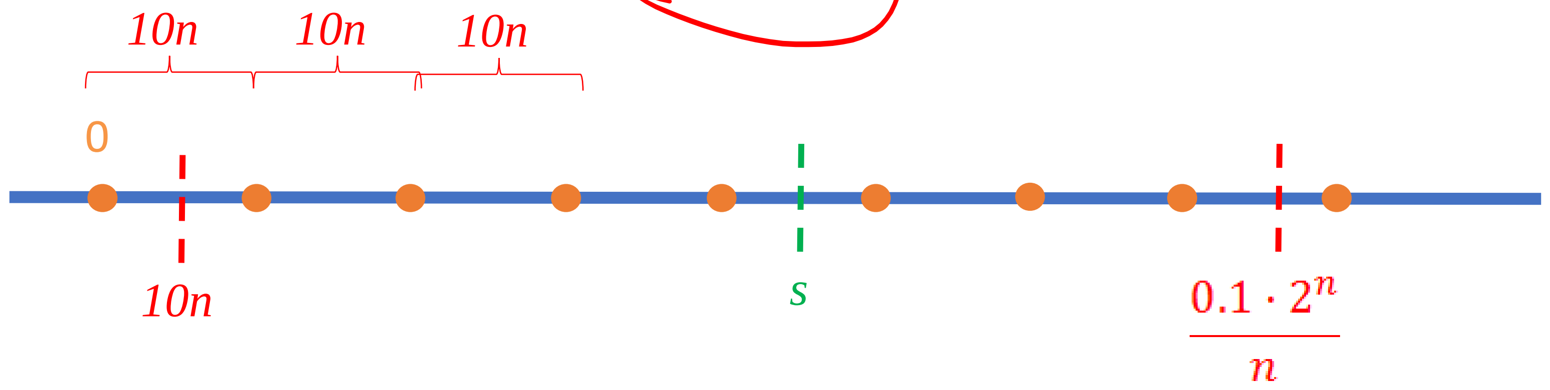


Idea: make function f_i easy for all inputs $> i$

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

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

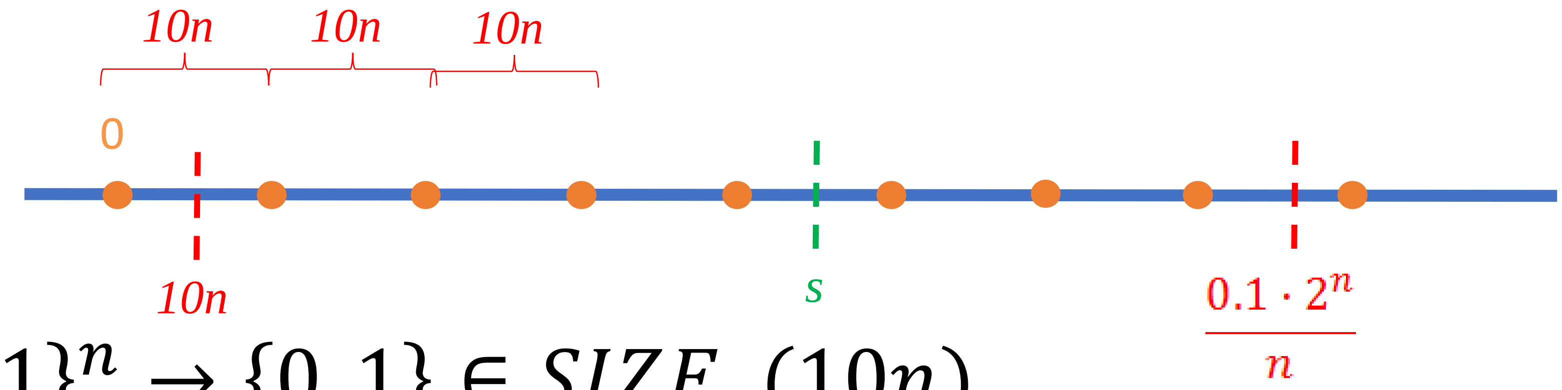
$$f_i(x) = \begin{cases} f^*(x) & \text{for the } \cancel{\text{first } i} \text{ inputs} \\ 0 & \text{for all other inputs} \end{cases}$$



Does f_0 work?

0

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

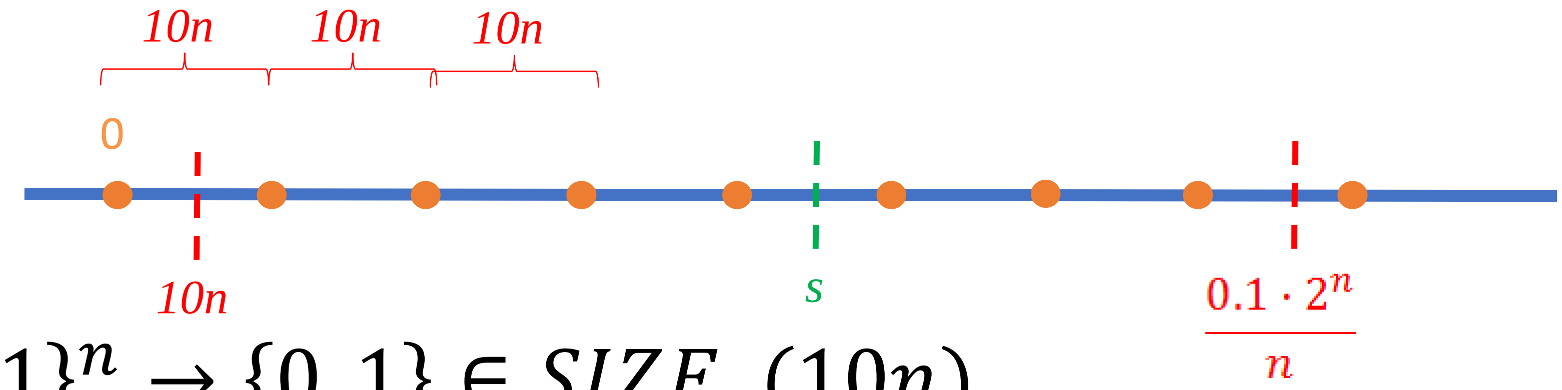


$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$

n-bit input ⇒ 2^n strings

Does f_{2^n} work?

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

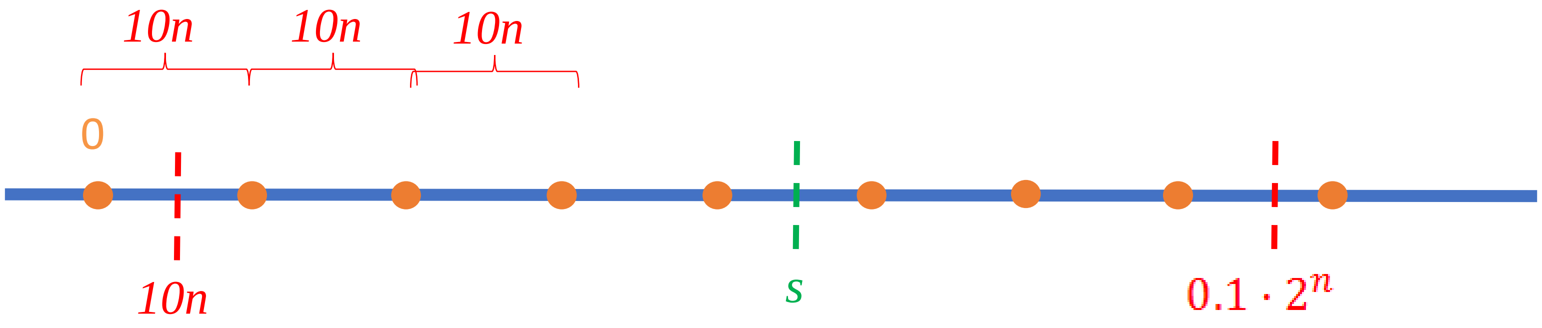


$f_0: \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$

Does f_{2^n} work?

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

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



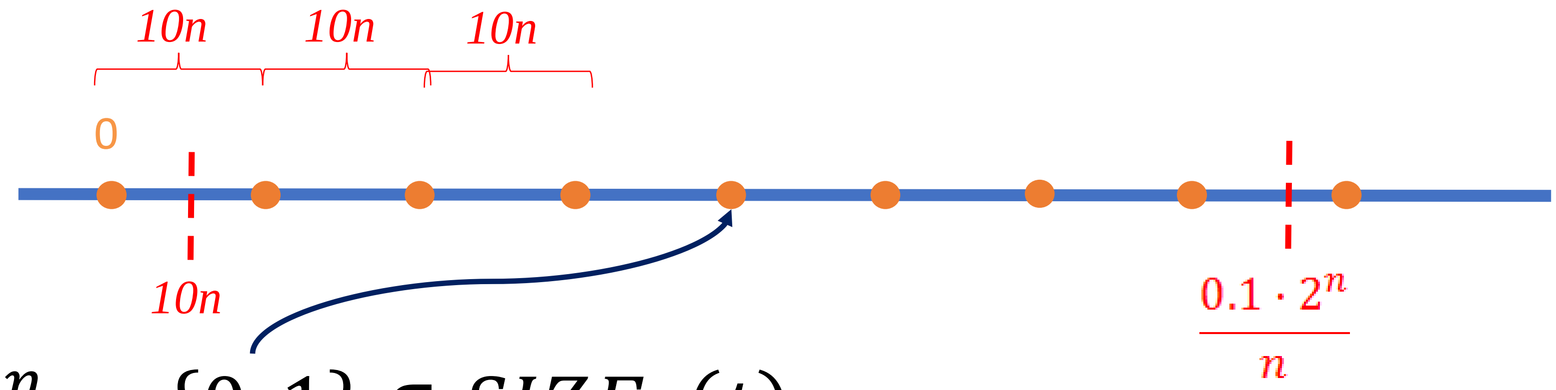
$$f_0 \colon \{0,1\}^n \to \{0,1\} \in SIZE_n(10n)$$

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

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

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

$$f_{i+1}(x) = \begin{cases} f^*(x) & \text{first } i+1 \\ 0 & \text{all other} \end{cases}$$



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

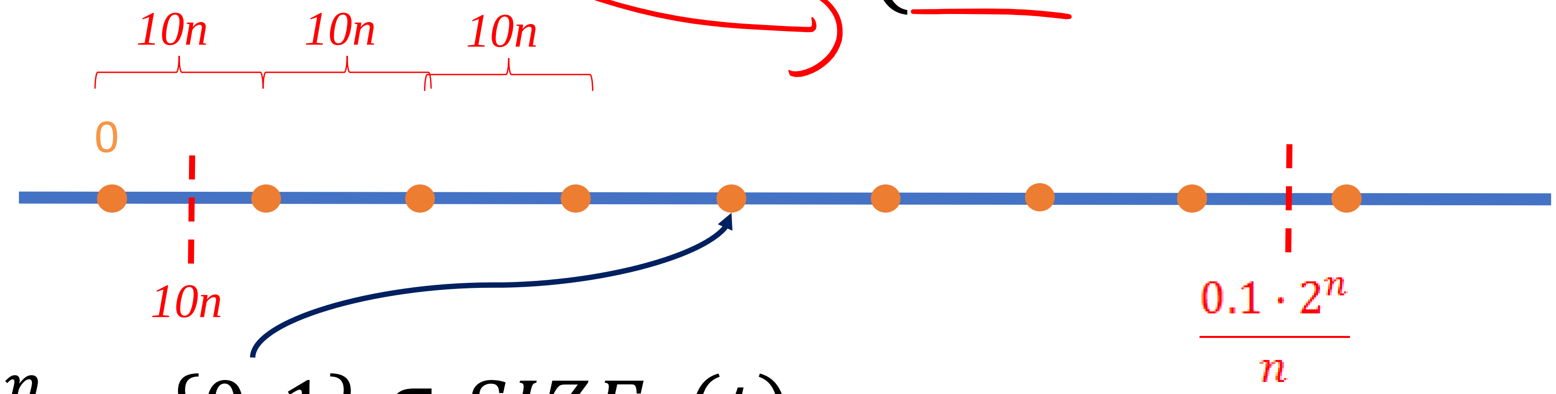
Inductive Step: $f_i \to f_{i+1}$

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

C_i, size t

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

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



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

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

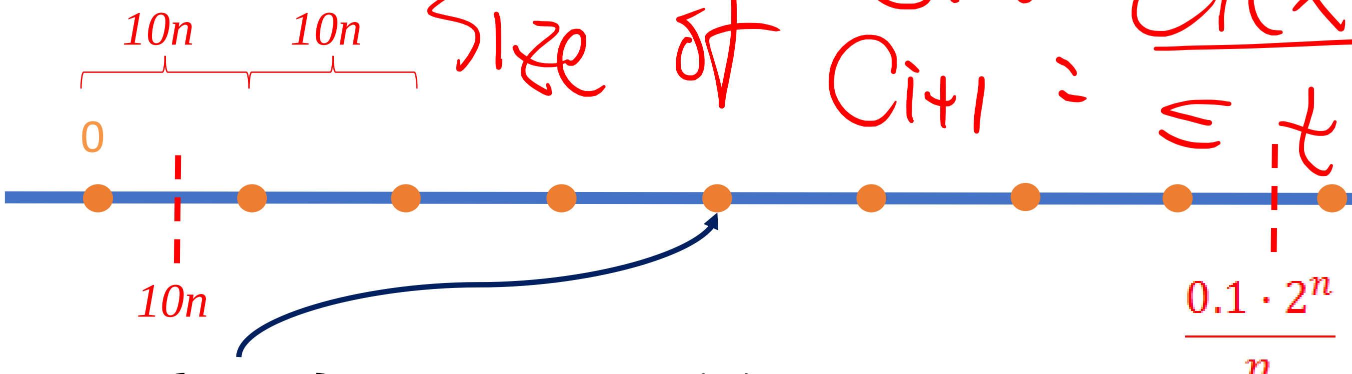
3. For all functions in the sequence, if function i can be computed using t gates, then the function $i+1$ can be computed using $t+10n$ gates.

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

$C_{i+1}(x) =$ if $x = i+1$: ret $f^*(x) = f^*(i+1)$

else ret $C_i(x)$ → size t

size of C_{i+1}: $\le t + 10n$



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

$f_{i+1} \in SIZE(t+10n)$

n-bit string

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

int(x)

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

Completing the Proof

Theorem 5.5 (Size Hierarchy Theorem)

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

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

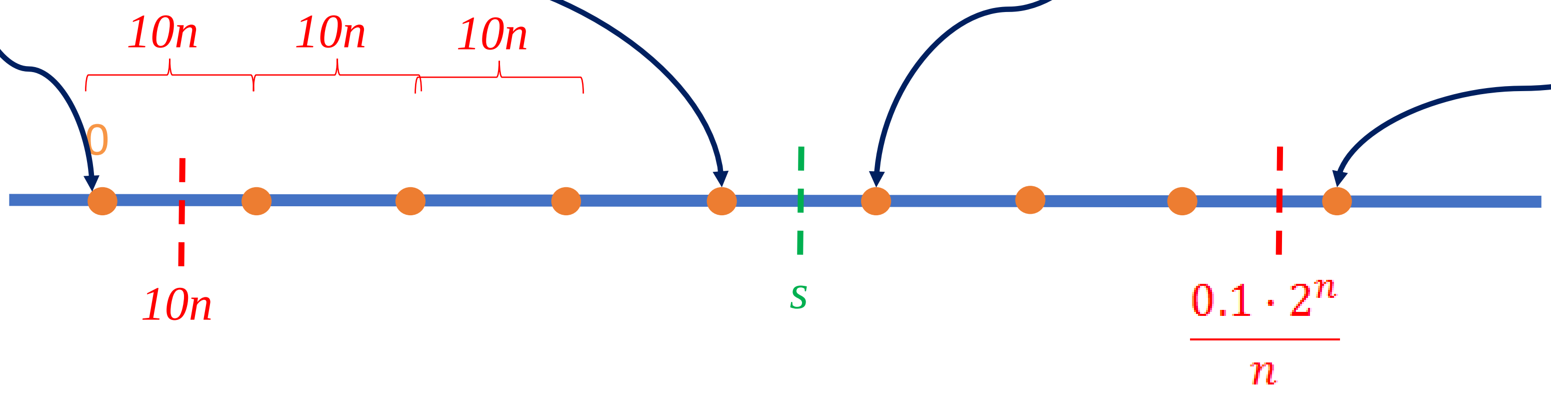
$f_i \in SIZE_n(s)$

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

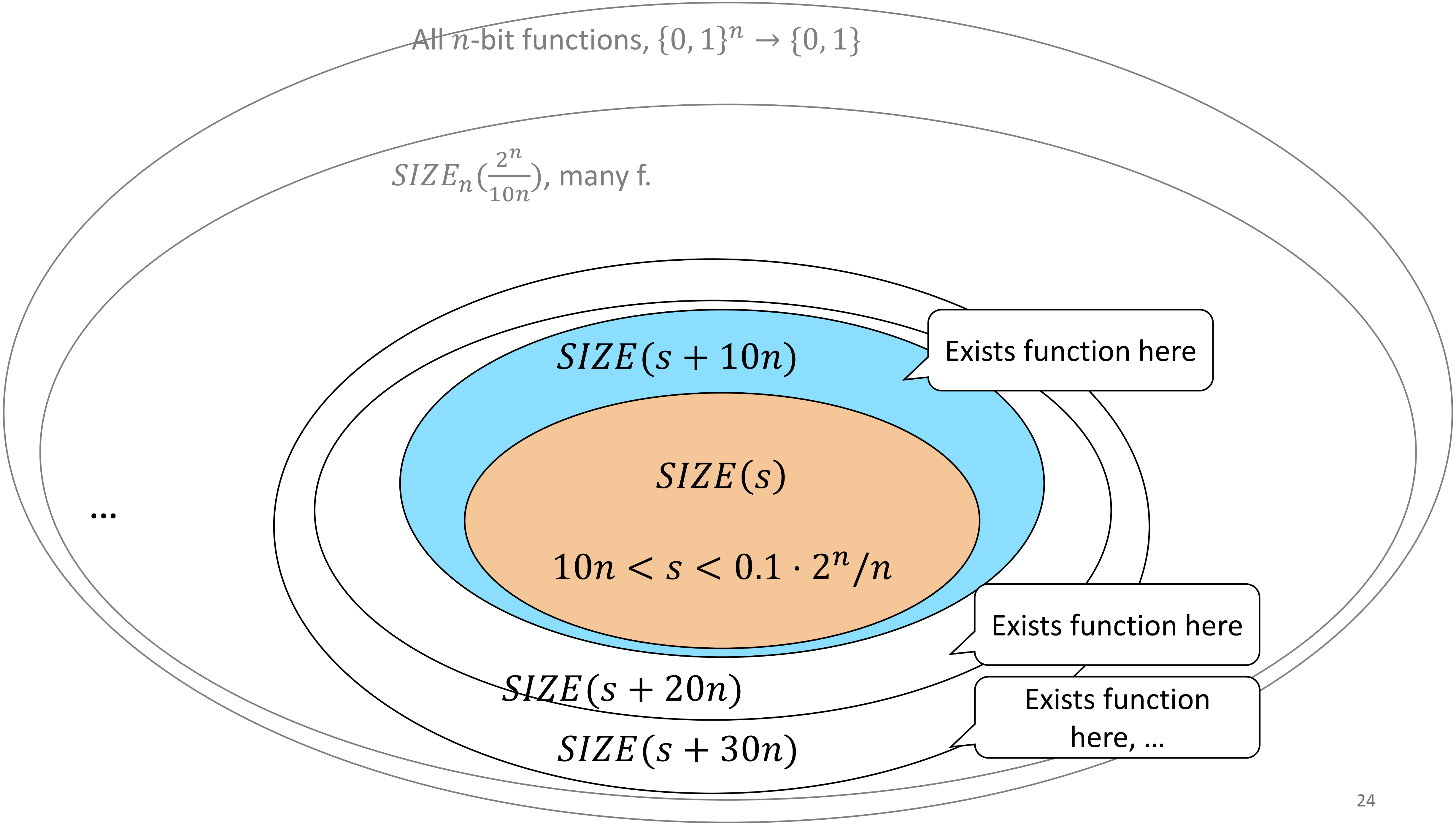
But $f_{i+1} \notin SIZE_n(s)$

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

$f_0 \in SIZE_n(10n)$



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


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

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

Either $(\sqrt{2}, \sqrt{2})$ or $(\sqrt{2}^{\sqrt{2}}, \sqrt{2})$

How about this existential proof?

Fact

Theorem: there is an irrational real number

Proof: $\sqrt{2}$ is irrational….

$\sqrt{2}^{\sqrt{2}}$

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

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

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

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 $2^{\Omega(n)}$

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

Given input of length n, interpret it as a graph G on $m = \sqrt{n}$ vertices and output 1 if G has $m/2$ vertices that are all pairwise connected.

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

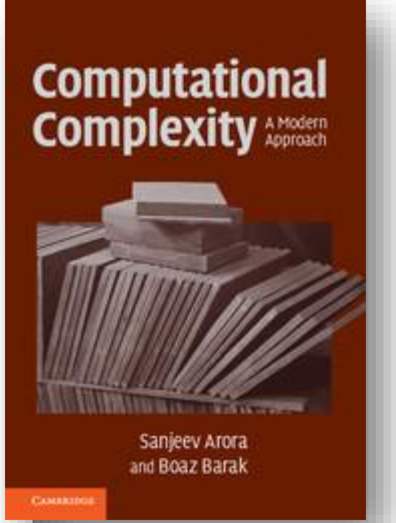
Chapter 14

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 $\mathbf{NP} \neq \mathbf{P}$. 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 $\mathbf{4.5}\boldsymbol{n} - \boldsymbol{o}(\boldsymbol{n})$."

Universal Circuits

Evaluate a NAND ckt using another NAND ckt

Recap: Representing Circuits as Bits

- Equivalent: NAND straightline program
- n-bit input, ℓ lines, m-bit output.
- Circuit size $s = \ell + m$ (# gates)

- Represented by a sequence of:
  – $2(s+1)$ natural numbers (at most)
  – $O(s \log s)$ bits

Consequences of Programs as Data

1. We can count the number of programs of certain size.

2. We can also feed a circuit as input to other circuits.

Consequence of Program as Data

- Can define the following function, whose outputs is based on running a program given as input:

$$EVAL_{s,n,m}(p,x) = \begin{cases} P(x) & p \in \{0,1\}^{S(s)} \text{ represents a size-}s\text{ program } P \text{ with } n \text{ inputs and } m \text{ outputs} \\ 0^m & \text{otherwise} \end{cases}$$

Are the P's the same?

But can we implement U as well?

Theorem 5.9 (Bounded Universality of NAND-CIRC programs)

For every $s, n, m \in \mathbb{N}$ with $s \geq m$ there is a NAND-CIRC program $U_{s,n,m}$ that computes the function $EVAL_{s,n,m}$.

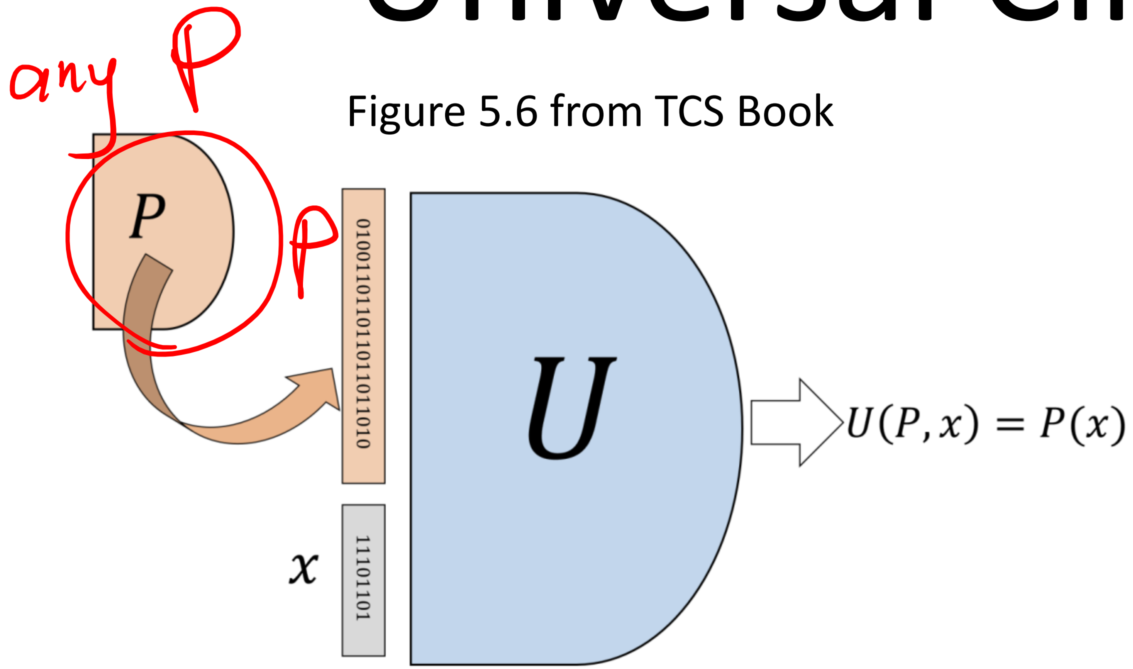
- Proof:

$$\left(LOOKUP_{s+n} \right) \times m \quad \text{~~cop~~ instances}$$

$$\leq 2^{s+n} \text{ size}$$

Universal Circuit/Program

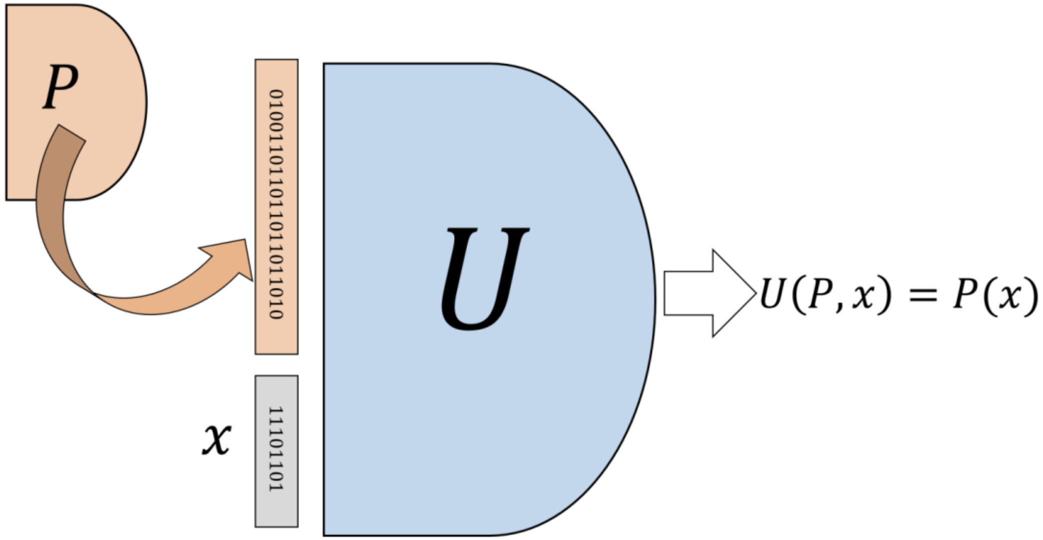
Figure 5.6 from TCS Book



program U takes a program description P and input x as its input, and "simulates" running P on x:

$$U(P, x) = P(x)$$

Points to pause and think



- Note that the fact that we ran a program using another program is already something to pause and appreciate

- But do we really want to use such inefficient simulation?

Charge

Circuit size hierarchy

Many hard functions

Many classes of circuit size

PS4: due this Friday 10:00pm
PRR5: due next Monday 10:00pm

Not yet PS5, yeah!