Algorithms and Complexity

Spring 2018

Aaram Yun

This page is intentionally left blank

Today

- Time complexity
- Oracle machines
- Circuits and advice

Time complexity

- Consider only algorithms that halt on each input
- $t_A : \{0,1\}^* \to \mathbb{N}$
  - With input $x \in \{0,1\}^*$, the algorithm A halts after $t_A(x)$ steps
- $T_A : \mathbb{N} \to \mathbb{N}$
  - $T_A(n) := \max_{x \in \{0,1\}^n} (t_A(x))$
- *Time complexity* of A

If $T_A(n) = B(n)$, then it means,
no matter what x is, as long as
$|x| = n$, $t_A(x) \leq B(n)$, or A halts on x
within $B(n)$ steps.
(time complexity of T is $B(n)$, in the "worst case")

Time complexity

$n^3 + 3n \le O(n^4)$

$T_A(n) \le O(n^c)$

» Efficient algorithm

  » A is efficient, if it is a *polynomial-time algorithm*

  » In other words, $T_A(n)$ is *polynomially bounded*

  » In other words, $T_A(n) \le n^c$ for some $c > 0$

$= O(n^c)$

$n^{1000000}$

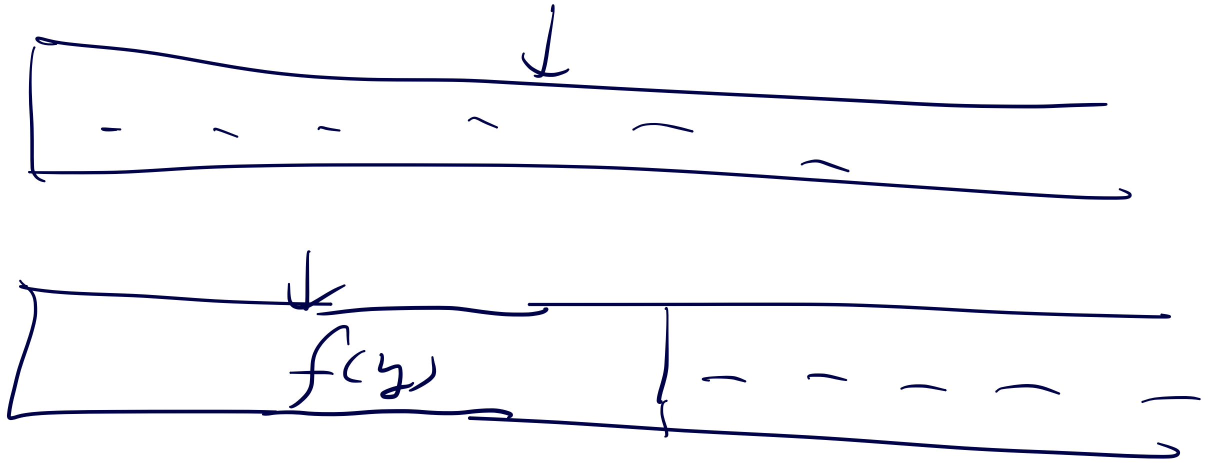
n^{100}

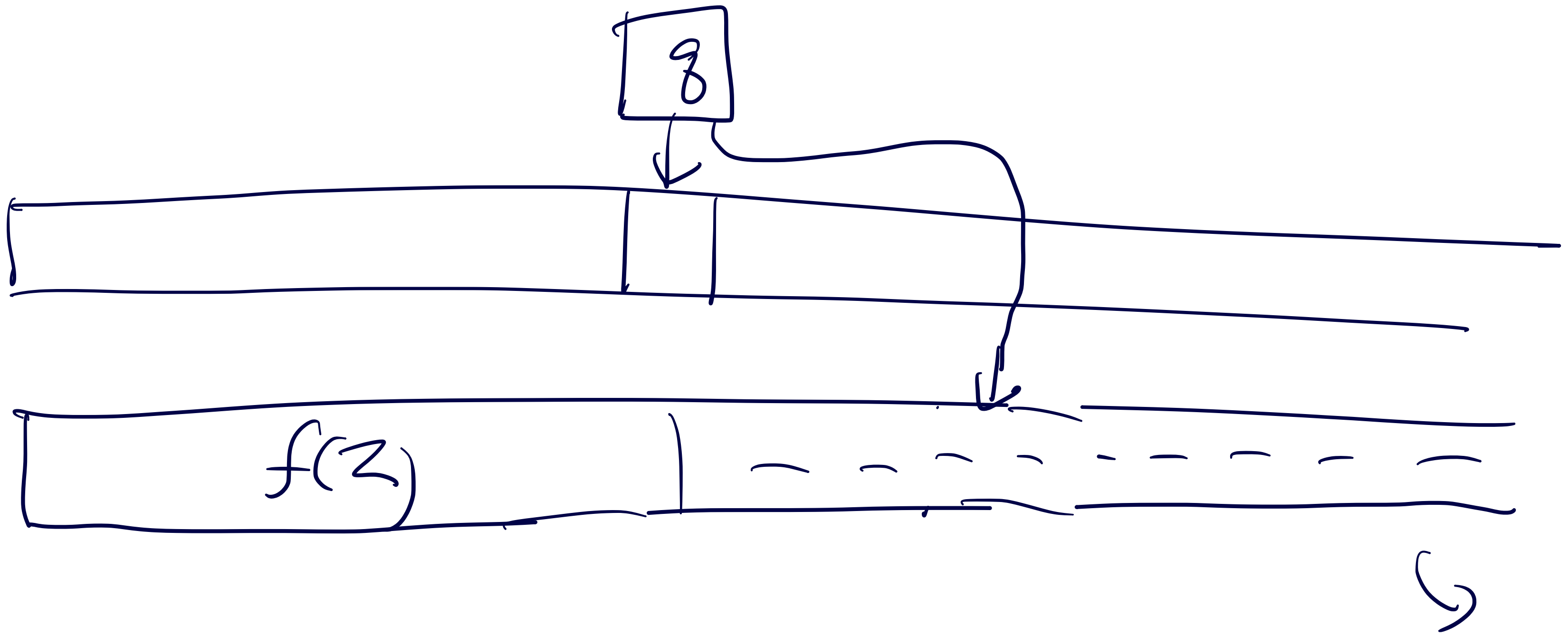
$\exists\, p(n)$: a polynomial in n, $T_A(n) \le p(n)$.

Oracle machines

- An oracle machine is just like a Turing machine, except
  - *Oracle invocation*
  - *Oracle spoke*
  - Notation: $M^f(x)$

$Q \ni q_{invoke}, \; q_{spoke}$





$M^f(x)$

$$\delta : Q \times \Sigma^2 \to Q \times \Sigma^2 \times \{-1, 0, 1\}^2$$

Turing reduction

- A problem Π is *Turing-reducible* to Π', if there exists an oracle machine M such that for every function f that solves Π', it holds that M^f solves Π

- Meaning?

$\Pi \leq_T \Pi'$

$\downarrow$ Halting

$\downarrow$ Some other problem.

f solves Π'

$\rightarrow$ M^f solves Π

N solves Π'

M^f solves Π

$\rightarrow$ We can construct M' which solves Π

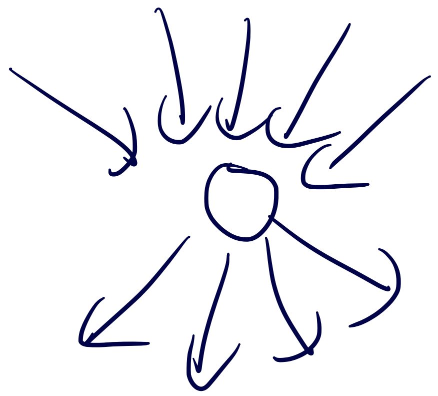
$f(y) = N(y)$

M'

Non-uniform model of computation

- *Non-uniform* model
  - No *uniformity* in algorithms for handling different size inputs
  - Circuits
  - Machines with advice

Boolean circuits



- A boolean circuit is a directed acyclic graph with labels on the vertices
  - Internal vertices (gates): in-degree and out-degree at least 1, labeled by a boolean operation ($\wedge, \vee, \neg$)
  - Sources: labeled by distinct natural numbers — without any incoming edges
  - Sinks: labeled by distinct, consecutive natural numbers — without any outgoing edges.

Boolean circuits

x_1 x_2 x_3 x_4

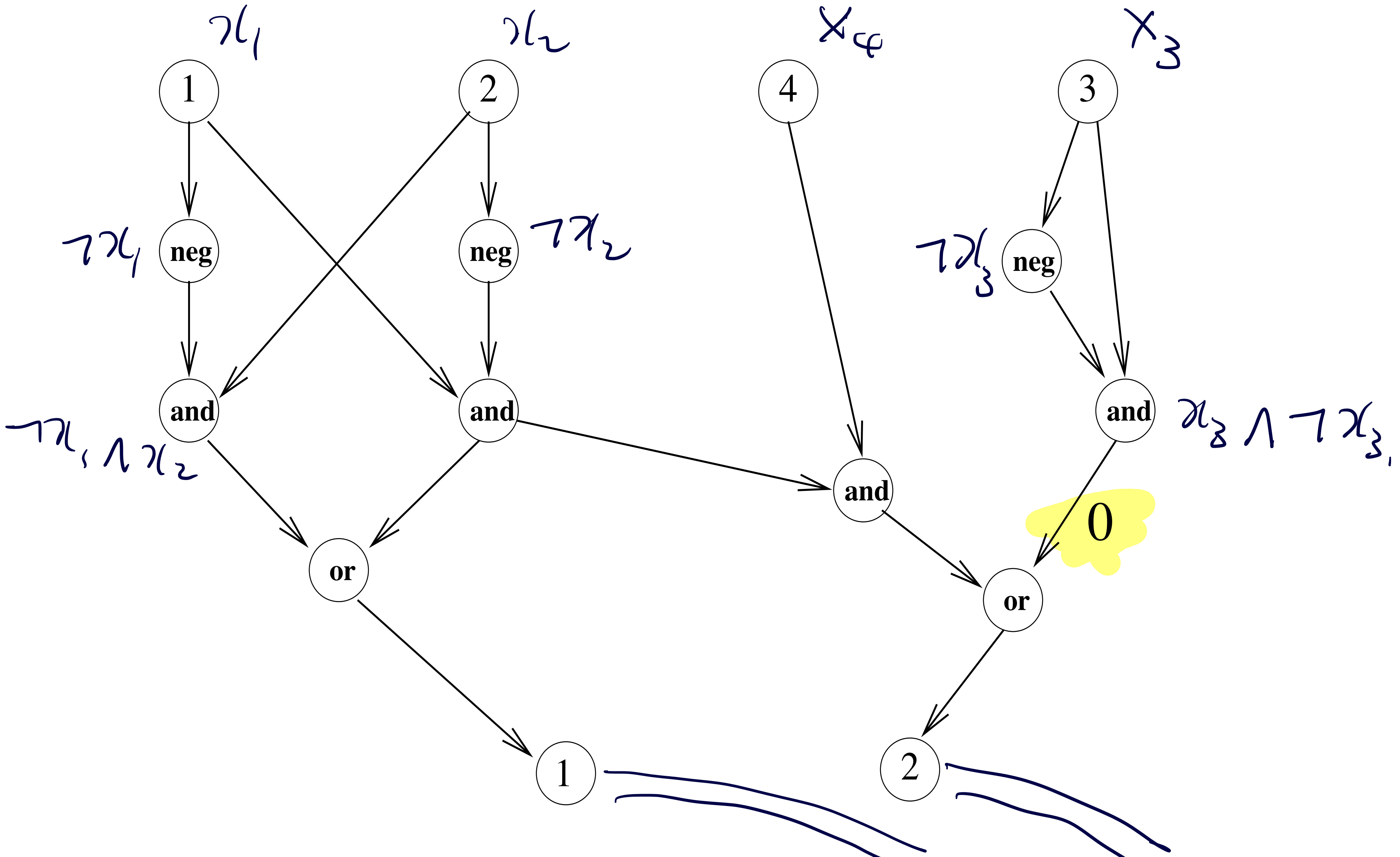
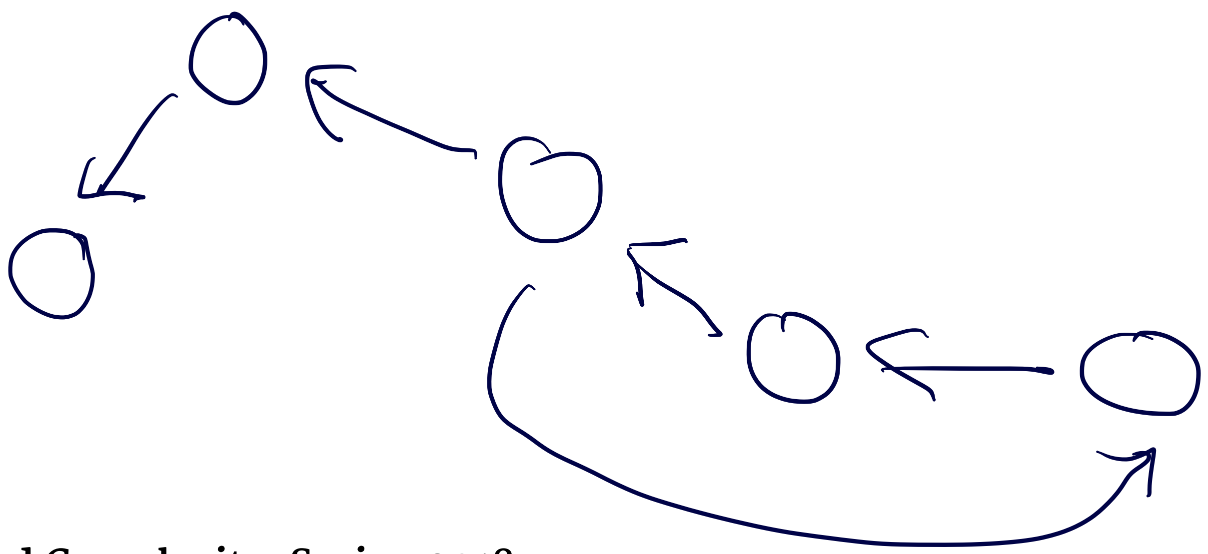


Figure 1.3: A circuit computing $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2, x_1 \wedge \neg x_2 \wedge x_4)$.

Boolean circuits

- C: a boolean circuit with n input labels and m output labels
  - C induces a function $f_C : \{0,1\}^n \to \{0,1\}^m$
- $\exists$ polytime algorithm for evaluating a circuit C on input x

Only problematic situation: $\exists$ some unevaluated vertices but all of them are not ready to be evaluated.



Circuit model

- $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$: a family of circuits
- $\mathcal{C}$ *computes* $f : \{0,1\}^* \to \{0,1\}^*$, if C_n computes $f|_{x\in\{0,1\}^n}$ for each $n \in \mathbb{N}$
  - Or, $C_{|x|}(x) = f(x)$ for all $x \in \{0,1\}^*$
- Why is this called *non-uniform*?

Circuit complexity

- *Size* of a circuit: total number of edges
- $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ has *size complexity* $s : \mathbb{N} \to \mathbb{N}$ if for every n, the size of C_n is $s(n)$
- $s_f : \mathbb{N} \to \mathbb{N}$: circuit complexity of $f : \{0,1\}^* \to \{0,1\}^*$
  - $s_f(n)$ is the size of the smallest circuit that computes $f_n = f|_{x\in\{0,1\}^n}$

Circuit model

- Some facts
  - For any f, the circuit complexity s_f is well-defined. $s_f(n)$ is at most exponential in n
  - A family of circuits is *uniform*, if there exists a Turing machine which generates the circuit family
  - If a function f is computed by an algorithm of time complexity t, then it has circuit complexity at most $\mathbf{poly}(t)$.

$\{C_n\}$ is <u>uniform</u> if $\exists \underbrace{\text{TM}}_{\text{polytime}}$ M s.t. $M(1^n) = \langle C_n \rangle$.

Prop) If $\{C_n\}$ is uniform, and if $\{C_n\}$ computes f, then $\exists$ a TM which computes f.

$N(x)$

$M(1^{|x|}) \to \langle C_n \rangle$

$\mathcal{E}(\langle C_n \rangle, x) = y$

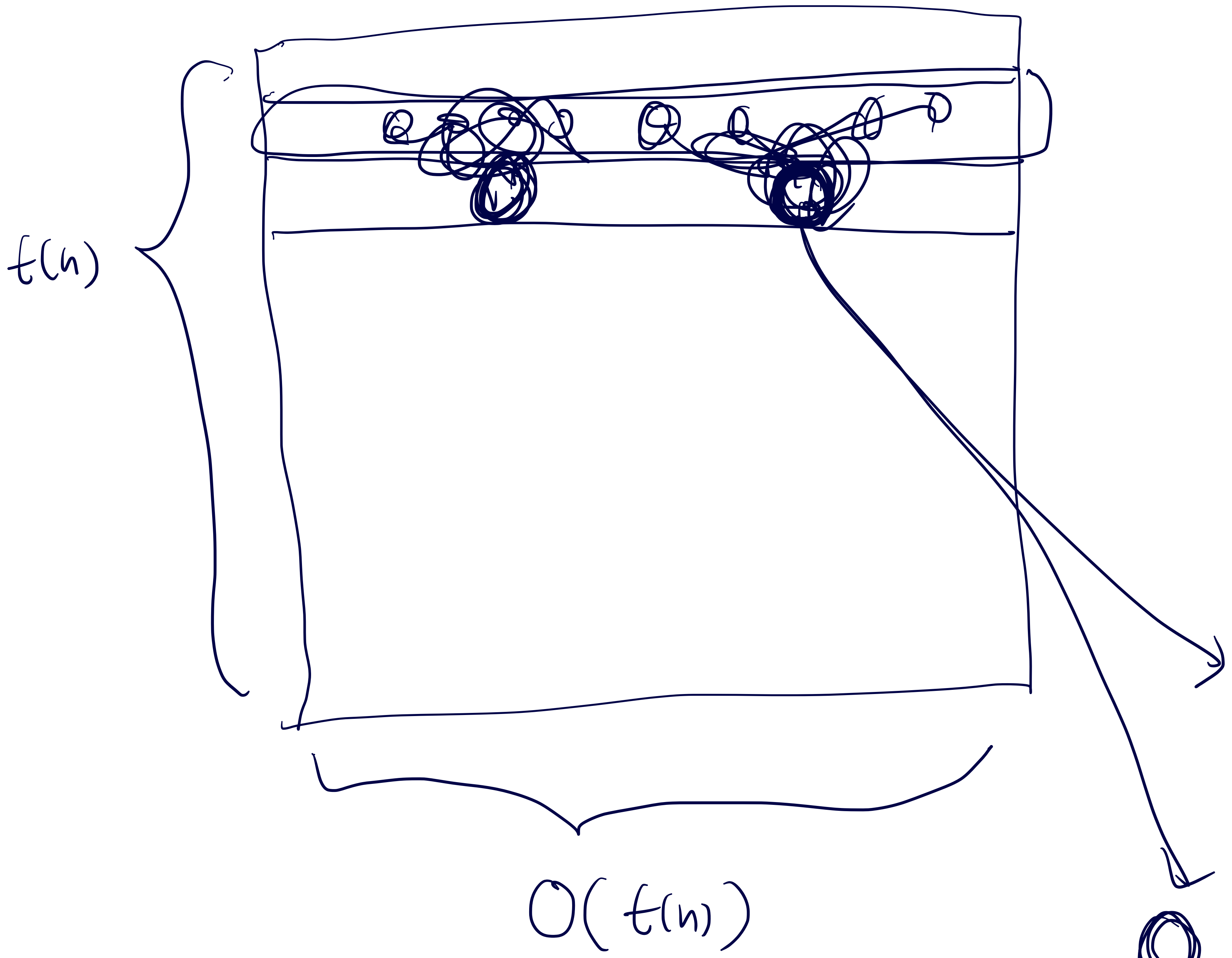
Example) $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$

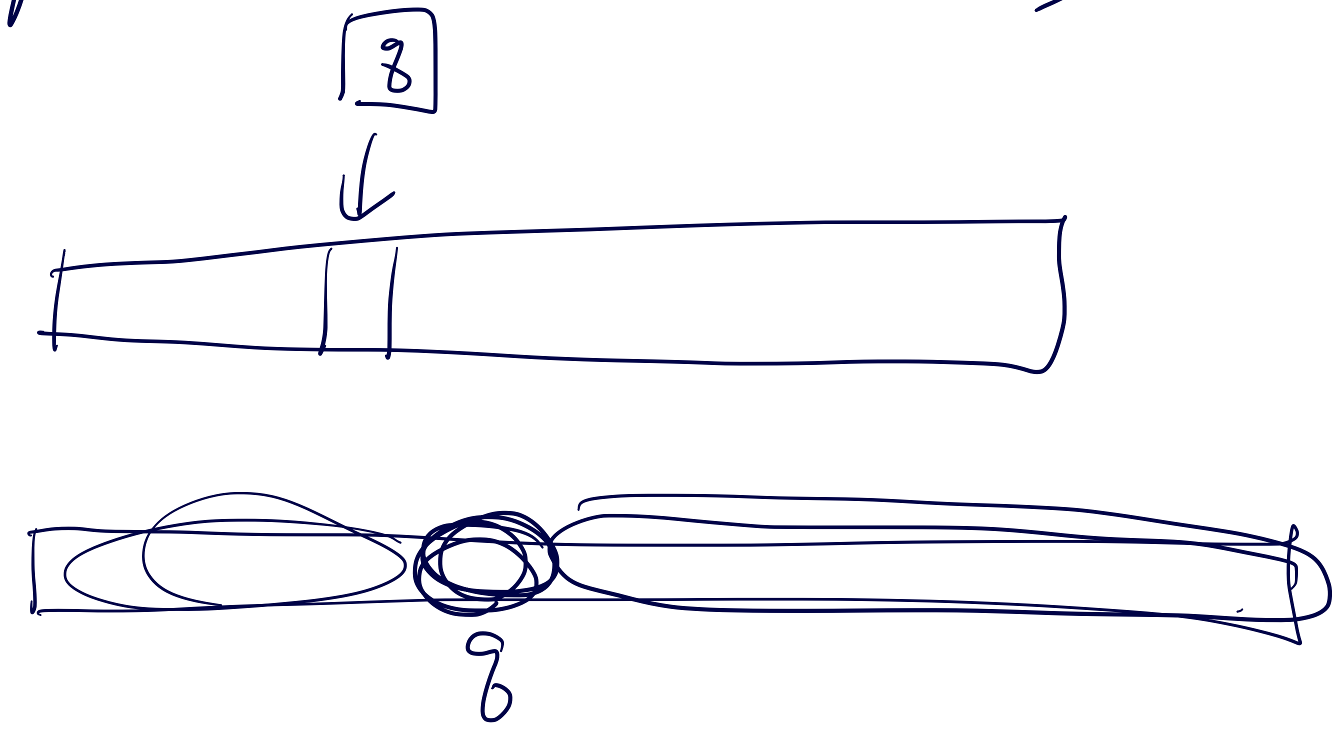
x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
	⋮		
1	1	1	1

$$f = (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee$$

$$(x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$

f can be computed by a TM within $t(n)$ steps $\qquad t(n) \geq n$.





n

is a function of some const no. of previous bits

$O = s \cdot O(t(n)^2)$

Machines that take advice

- An algorithm A computes $f : \{0,1\}^* \to \{0,1\}^*$ using advice of length $l : \mathbb{N} \to \mathbb{N}$, if $\exists (a_n)_{n \in \mathbb{N}}$ such that
  - For every $x \in \{0,1\}^*$, $A(a_{|x|}, x) = f(x)$
  - For every $n \in \mathbb{N}$, $|a_n| = l(n)$
- $(a_n)_{n \in \mathbb{N}}$ is the *advice sequence*

Circuits and advices

- Any function having circuit complexity s can be computed using advice of length $O(s \log s)$
  - A graph with v vertices and e edges can be described by a string of length $2e \log_2 v$

$2\log_2 v$: the no. of bits required to write two vertices

$A(a_n, x) = f(x)$.

$A(y, x)$ $|y| = \ell(n)$, $|x| = n$

$\tilde{C}_n(y,x) := A(y,x)$ $\tilde{C}_n$: circuit s.t. $\tilde{C}_n(a_n, x) = f(x)$

$C_n(x) := \tilde{C}_n(a_n, x)$