Quantum Algorithms
Lecture 3
Boolean circuits II

Zhejiang University

Basic algorithms. Depth, space and width.

Polynomial-time uniform

F_n are computed by polynomial-size circuits C_n with the following property: there exists a TM that for each positive integer n runs in time $poly(n)$ and constructs the circuit C_n.

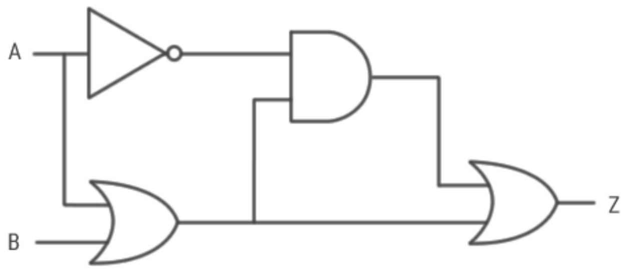
Such C_n - polynomial-time uniform.

Uniform sequences of circuits – effectively constructed.

Depth

Depth in hardware

Depth is important parameter together with size. Depth is the time that is needed to carry out all assignments in the circuit, if we can do more than one in parallel.



Example: size=4, depth=3.

Depth

More formally, the depth of a Boolean circuit is the maximum number of gates on any path from the input to the output. The depth is $\leq d$ if and only if one can arrange the gates into d layers, so that the input bits of any gate at layer j come from the layers $1, \ldots, j-1$.

Bounded fan-in

Fan-in - number of input bits.

We assume that all gates have bounded fan-in.

This is always the case when circuits are built over a finite basis. Unbounded fan-in can occur, for example, if one uses OR gates with an arbitrary number of inputs.

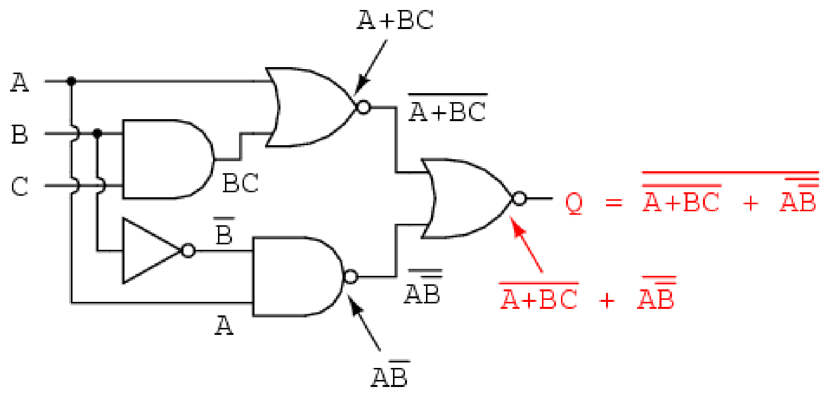
Bounded fan-out

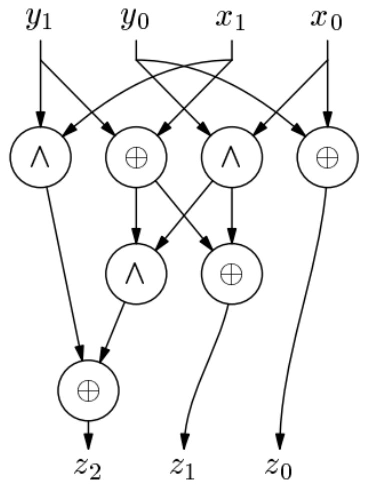
Fan-out - the number of times an input or an auxiliary variable is used.

This restriction is needed to allow conversion of Boolean circuits into quantum circuits without extra cost.

Trivial gates and formula

If it is necessary to use the variable more times, one may insert additional "trivial gates" (identity transformations) into the circuit, at the cost of some increase in size and depth. Note that a formula is a circuit in which all auxiliary variables have fan-out 1, whereas the fan-out of the input variables is unbounded.


A+BC
A
B
C
$\overline{A+BC}$
BC
$\overline{B}$
$Q = \overline{\overline{A+BC} + \overline{A\overline{B}}}$
$\overline{A\overline{B}}$
$\overline{A+BC} + \overline{A\overline{B}}$
A
$A\overline{B}$



Interesting properties of depth

Let C be an $O(logn)$-depth circuit which computes some function $f: B^n \to B^m$. After eliminating all extra variables and gates in C (those which are not connected to the output), we get a circuit of size $poly(n+m)$.

Interesting properties of depth

Let C be a circuit (over some basis B) which computes a function $f\colon B^n \to B$. C can be converted into an equivalent (i.e., computing the same function) formula C' of the same depth over the same basis.

Interesting properties of depth

Balanced formula.

Let C be a formula of size L over the basis {NOT, OR, AND} (with fan-in ≤ 2). It can be converted into an equivalent formula of depth $O(logL)$ over the same basis.

Therefore, it does not matter whether we define the formula complexity of a function in terms of size or in terms of depth. This is not true for the circuit complexity.

Interesting properties of depth

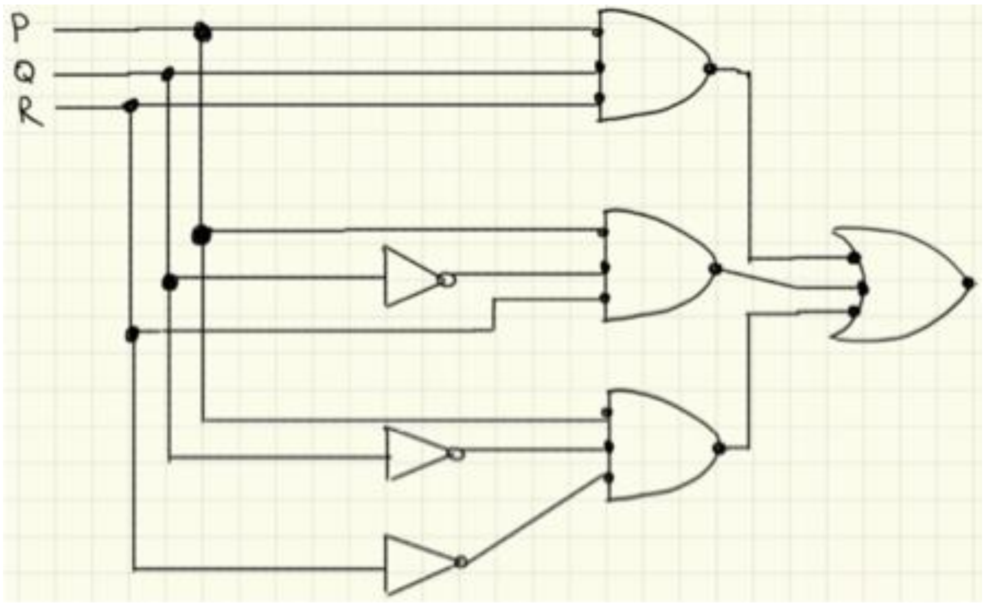
Any function can be computed by a circuit of depth ≤ 3 with gates of type NOT, AND, and OR, if we allow AND- and OR-gates with arbitrary fan-in and fan-out.

Note that we can replace AND- / OR- gate with fan-in/fan-out of n with sequence of $log n$ AND- / OR- gates.

Interesting properties of depth

Any function can be computed by a circuit of depth ≤ 3 with gates of type NOT, AND, and OR, if we allow AND- and OR-gates with arbitrary fan-in and fan-out.

For example, we can consider DNF function.



Interesting properties of depth

The function PARITY is the sum of n bits modulo 2.

Suppose it is computed by a circuit of depth 3 containing NOT-gates, AND-gates and OR-gates with arbitrary fan-in. The size of the circuit is exponential (at least c^n for some $c > 1$ and for all n).

Basic algorithms

Comparison

Construct a circuit of size $O(n)$ and depth $O(logn)$ that tells whether two n-bit integers are equal and if they are not, which one is greater.

There are three possible results of the comparison of two numbers x and y: $x > y$, $x = y$, or $x < y$. We construct a circuit which yields a two-bit answer encoding these three possibilities.

Comparison

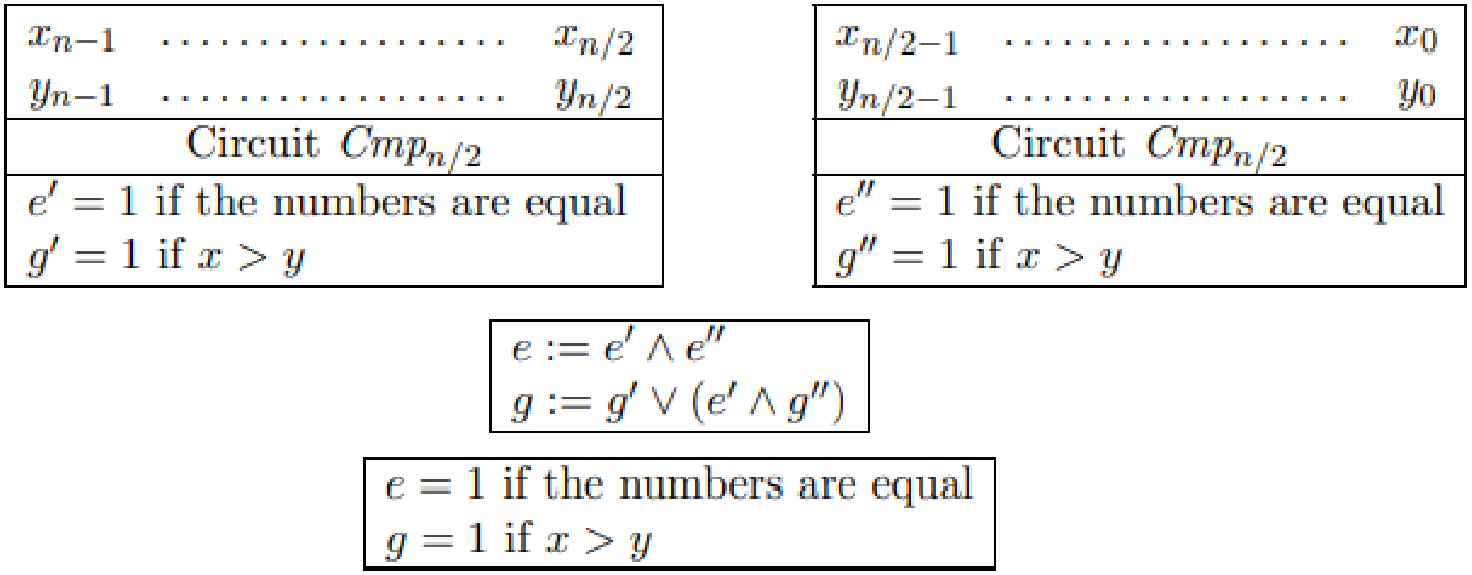

x_{n-1} .... $x_{n/2}$
y_{n-1} .... $y_{n/2}$
Circuit $Cmp_{n/2}$
$e' = 1$ if the numbers are equal
$g' = 1$ if $x > y$
$x_{n/2-1}$ .... x_0
$y_{n/2-1}$ .... y_0
Circuit $Cmp_{n/2}$
$e'' = 1$ if the numbers are equal
$g'' = 1$ if $x > y$
$e := e' \wedge e''$
$g := g' \vee (e' \wedge g'')$
$e = 1$ if the numbers are equal
$g = 1$ if $x > y$

Fig. S2.1. Circuit Cmp_n for comparison of n-bit numbers (size of circuit $= O(n)$, depth $= O(\log n)$).

Comparison

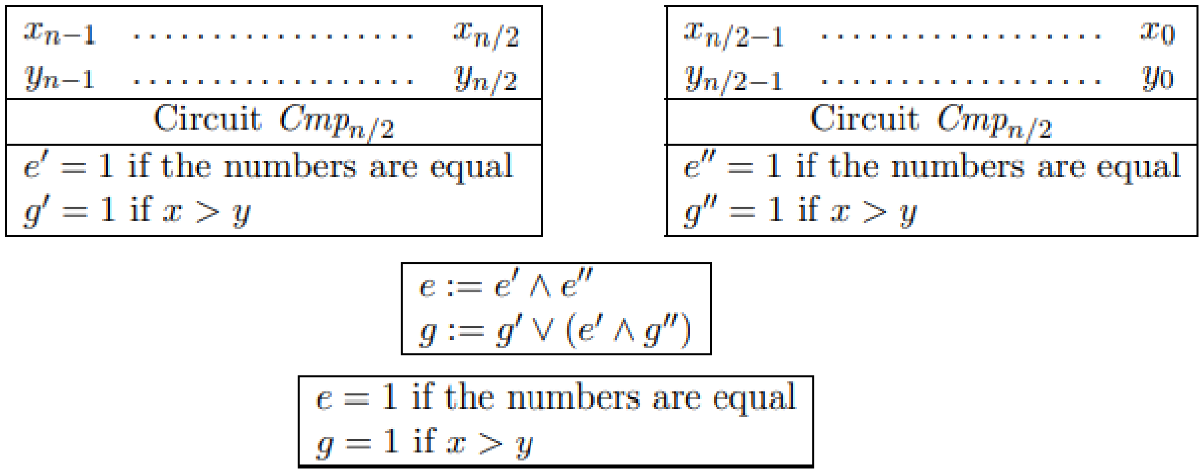
A circuit for the comparison of n-bit numbers is constructed recursively. We compare the first $n/2$ (high) bits and the last $n/2$ (low) bits separately, and then combine the results. (This "divide and conquer" method will be used for the solution of many other problems.)

Comparison

We estimate the size L_n and depth d_n of this circuit. It is easy to see that

$$L_n = 2L_{n/2} + 3,\ d_n = d_{n/2} + 2.$$

Therefore, $L_n = O(n)$ and $d_n = O(logn)$.



x_{n-1} .... $x_{n/2}$
y_{n-1} .... $y_{n/2}$
Circuit $Cmp_{n/2}$
$e' = 1$ if the numbers are equal
$g' = 1$ if $x > y$
$x_{n/2-1}$ .... x_0
$y_{n/2-1}$ .... y_0
Circuit $Cmp_{n/2}$
$e'' = 1$ if the numbers are equal
$g'' = 1$ if $x > y$
$e := e' \wedge e''$
$g := g' \vee (e' \wedge g'')$
$e = 1$ if the numbers are equal
$g = 1$ if $x > y$

Fig. S2.1. Circuit Cmp_n for comparison of n-bit numbers (size of circuit $= O(n)$, depth $= O(\log n)$).

Comparison

A bit harder solution.

The inequality $x > y$ holds if and only if the number $x + (2^n - 1 - y)$ is greater than $2^n - 1$, which can be checked by looking at the n-th bit of this number. Note that $2^n - 1 - y$ is computed very easily (by negating all bits of y), so the comparison can be reduced to addition.

Access by index

Let $n = 2^l$. Construct circuit of size $O(n)$ and depth $O(logn)$.

Given an n-bit string $x = x_0 \cdots x_{n-1}$ (a table) and an l-bit number j (an index), find x_j.

Search

Let $n = 2^l$. Construct circuit of size $O(n)$ and depth $O(logn)$.

Evaluate the disjunction $y = x_0 \vee \cdots \vee x_{n-1}$, and if it equals 1, find the smallest j such that $x_j = 1$.

Parallelizing computation

A finite-state automaton is a device with an input alphabet A', an output alphabet A'', a set of states Q and a transition function $D: Q \times A' \to Q \times A''$. It is initially set to some state $q_0 \in Q$. Then it receives input symbols $x_0, \ldots, x_{m-1} \in A'$ and changes its state from q_0 to q_1 to q_2, etc., according to the rule

$$(q_{j+1}, y_j) = D(q_j, x_j).$$

Such a device is a part of a Turing machine.

Parallelizing computation

The iterated application of D defines a function D^m: $(q_0, x_0, \dots, x_{m-1}) \to (q_m, y_0, \dots, y_{m-1})$. We may assume without loss of generality that $Q = B^r$, $A' = B^{r'}$, $A'' = B^{r''}$; then D: $B^{r+r'} \to B^{r+r''}$ whereas D^m: $B^{r+r'm} \to B^{r+r''m}$.

Parallelizing computation

The work of automaton:

$$\begin{tikzcd}
 & x_{m-1} \arrow[d] & & x_1 \arrow[d] & x_0 \arrow[d] & \\
q_m & \boxed{D} \arrow[l] \arrow[d] & \cdots \arrow[l, "q_{m-1}"'] & \boxed{D} \arrow[l, "q_2"'] \arrow[d] & \boxed{D} \arrow[l, "q_1"'] \arrow[d] & q_0 \arrow[l] \\
 & y_{m-1} & & y_1 & y_0 &
\end{tikzcd}$$

Parallelization of iteration

Let the integers r, r', r'' and m be fixed; set $k = r + r' + r''$. Construct a circuit of size $\exp(O(k))m$ and depth $O(k \log m)$ that receives a transition function D: $B^{r+r'} \to B^{r+r''}$ (as a value table), an initial state q_0 and input symbols $x_0, \dots, x_{m-1}$, and produces the output $(q_m, y_0, \dots, y_{m-1}) = D^m(q_0, x_0, \dots, x_{m-1})$.

Addition

A circuit of size $O(n)$ and depth $O(logn)$ that adds two n-bit integers.

$$q_0 := 0, \qquad q_{j+1} := \begin{cases} 0 & \text{if } x_j + y_j + q_j < 2 \\ 1 & \text{if } x_j + y_j + q_j \geq 2 \end{cases} \qquad (j = 0, \ldots, n-1),$$

$$z_j := x_j \oplus y_j \oplus q_j, \qquad z_n := q_n,$$

where $q_0, \ldots, q_n$ are the carry bits.

Iterated addition

A circuit of size $O(nm)$ and depth $O(logn + logm)$ for the addition of m n-digit numbers.

Multiplication

A circuit of size $O(nm)$ and depth $O(logn + logm)$ for the multiplication of an n-digit number by an m-digit number.

Division

Compute the inverse of a real number $x = \overline{1.x_1x_2\ldots} = 1 + \sum_{j=1}^{\infty} 2^{-j}x_j$ with precision 2^{-n}. By definition, this requires to find a number z such that $|z - x^{-1}| \leq 2^{-n}$. For this to be possible, x must be known with precision 2^{-n} or better; let us assume that x is represented by an $n+1$-digit number x' such that $|x' - x| \leq 2^{-(n+1)}$. Construct an $O(n^2 \log n)$-size, $O((log n)^2)$-depth circuit for the solution of this problem.

Division

Divide two integers with remainder: $(a, b) \to (\lfloor a/b \rfloor, (a\, mod\, b))$, where $0 \le a < 2^k b$ and $0 < b < 2^n$. In this case, construct a circuit of size $O(nk + k^2 \log k)$ and depth $O(logn + (logk)^2)$.

Division

Size:

$O(n^2 \log n)$ vs $O(nk \ + \ k^2 \log k)$

Depth:

$O((logn)^2)$ vs $O(logn \ + \ (logk)^2)$

Majority

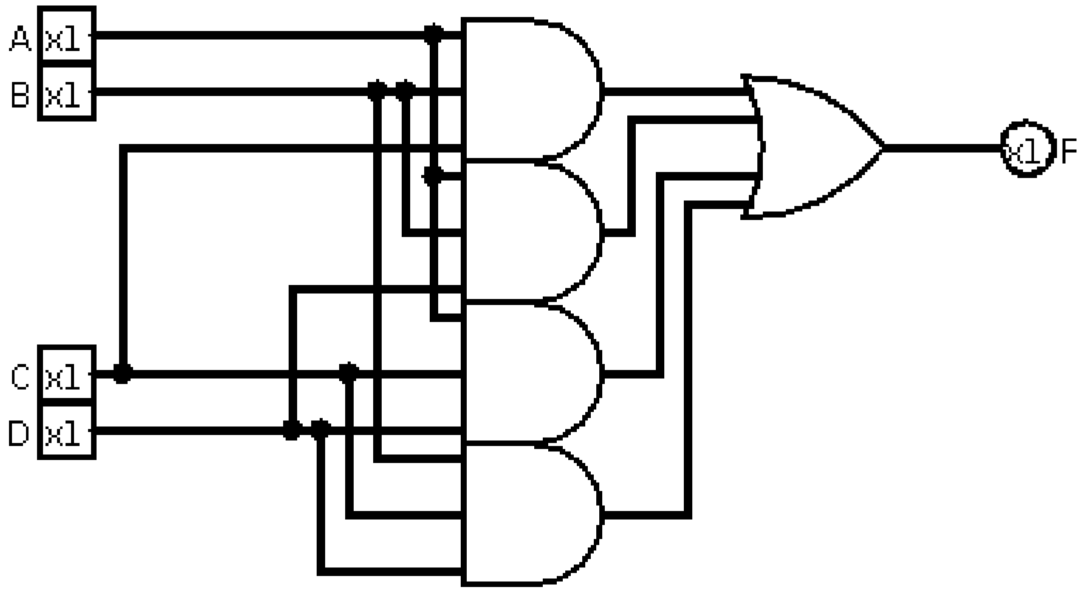
The majority function $MAJ: B^n \rightarrow B$ equals 1 for strings in which the number of 1s is greater than the number of 0s, and equals 0 elsewhere. Construct a circuit of size $O(n)$ and depth $O(logn)$ that computes the majority function.

Majority

To compute the function MAJ we count 1s among the inputs, i.e., compute the sum of the inputs. Then we compare the result with $n/2$.

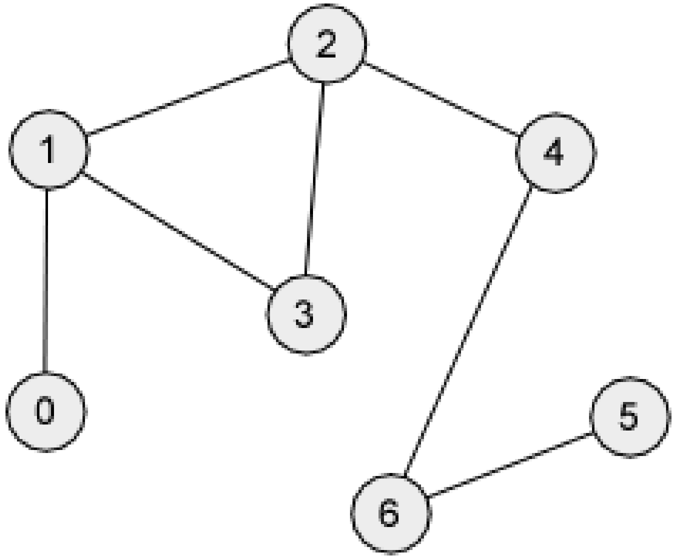
Majority

More naïve solution can look like check of all combinations where more than half of inputs is equal to 1. So we check all possible combinations of $n/2+1$ input values with AND operation, and then pass to OR operation.



Connecting path

A circuit of size $O(n^3 logn)$ and depth $O((logn)^2)$ that checks whether two fixed vertices of an undirected graph are connected by a path. The graph has n vertices, labeled $1, \dots, n$; there are $m = n(n-1)/2$ input variables x_{ij} (where $i < j$) indicating whether there is an edge between i and j.



Space and width

Depth vs space

In the solution to the above problems parallel algorithms were provided, which were described by circuits of polylogarithmic depth. If the circuit size is not taken into account, then uniform computation with poly-logarithmic depth is equivalent to computation with poly-logarithmic space.

Very limited space

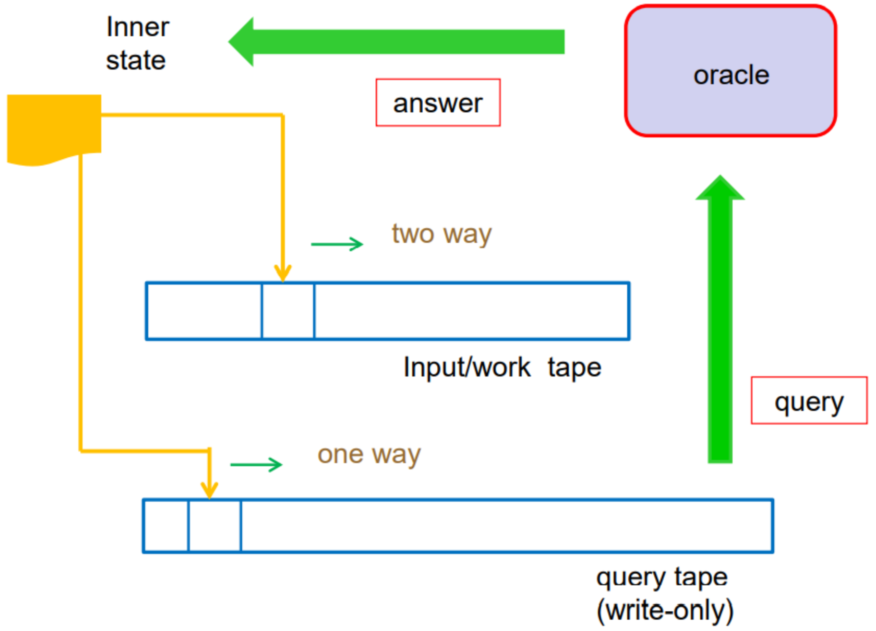
We are going to study computation with very limited space — so small that the input string would not fit into it. So, let us assume that the input string $x = x_0 \cdots x_{n-1}$ is stored in a supplementary read-only memory, and the Turing machine can access bits of x by their numbers.

Very limited space

We may think of the input string as a function $X\colon j \to x_j$ computed by some external agent, called “oracle”. The length of an "oracle query" j is included into the machine work space, but the length of x is not. This way we can access all input bits with space $O(logn)$ (but no smaller).

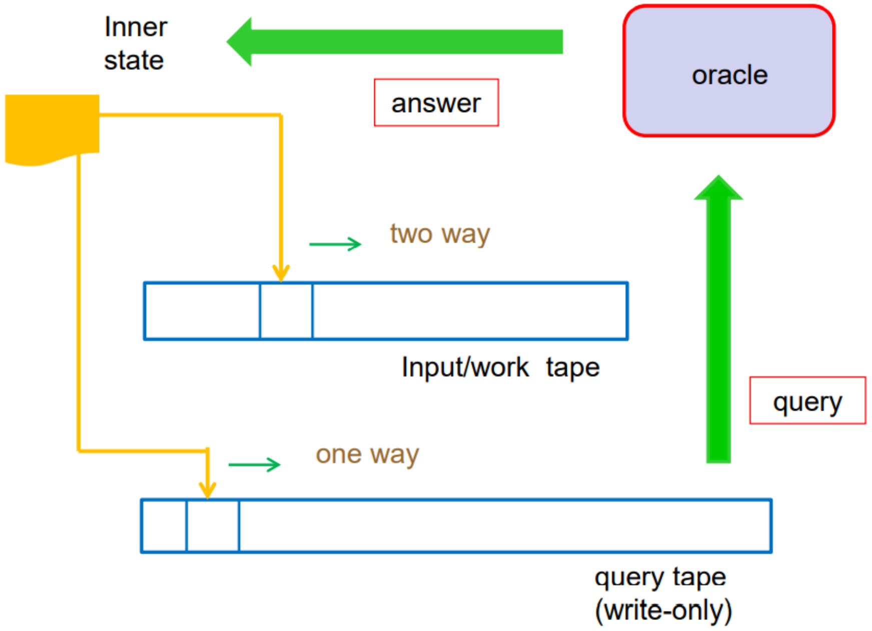
TM with oracle

Let $X: A^* \to A^*$ be a partial function. A Turing machine with oracle X is an ordinary TM with a supplementary tape, in which it can write a string z and have the value of $X(z)$ available for inspection at the next computation step.



TM input access

In our case $X(z) = x_j$, where z is the binary representation of j $(0 \le j \le n-1)$ by $\lceil \log_2 n \rceil$ digits; otherwise $X(z)$ is undefined.



Oracle TM computation

The computation of a function $f\colon B^n \to B^m$ by such a machine is defined as follows. Let x ($|x| = n$) be the input. We write n and another number k $(0 \leq k < m)$ on the machine tape and run the machine. We allow it to query bits of x.

Oracle TM computation

When the computation is complete, the first cell of the tape must contain the k-th bit of $f(x)$. Note that if the work space is limited by s, then $n \leq 2^s$ and $\mathrm{m} \leq 2^s$. The computation time is also bounded: the machine either stops within $\exp(O(s))$ steps, or enters a cycle and runs forever.

Simulation of small depth

Can be done in small space.

There exists a Turing machine that evaluates the output variables of a circuit of depth d over a fixed basis, using work space $O(d + logm)$ (where m is the number of the output variables). The input to the machine consists of a description of the circuit and the values of its input variables.

Small space is parallelizable

Let M be a TM. For each choice of n, m, s, let $f_{n,m,s} : B^n \to B^m$ be the function computed by the machine M with space s. There exists a family of $\exp(O(s))$-size, $O(s^2)$-depth circuits $C_{n,m,s}$ which compute the functions $f_{n,m,s}$.

These circuits can be constructed by a TM with space $O(s)$.

Computation space

We discuss the space restriction in terms of TMs.

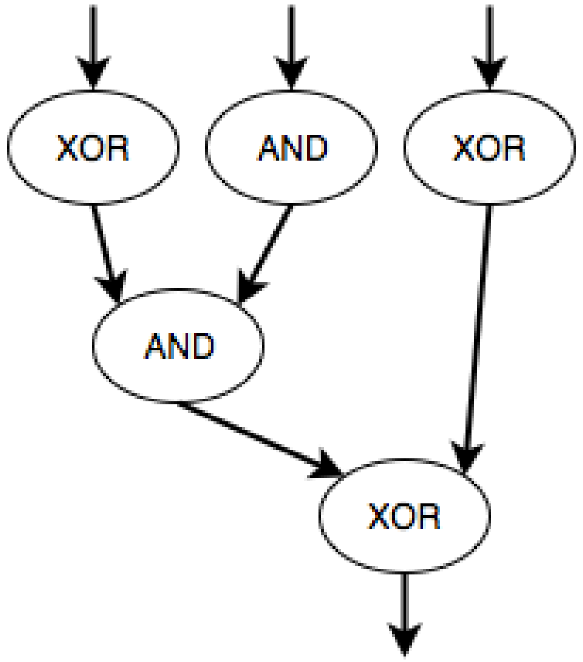
The circuit analog of the computation space - the circuit width.

Circuit width

Let C be a circuit whose gates are arranged into d layers. The width of C is the maximum amount of information carried between the layers, not including the input variables. More formally, for each $l = 1, \dots, d$ we define w_l to be the number of auxiliary variables from layers $1, \dots, l$ that are output variables or connected to some variables from layers $l + 1, \dots, d$ (i.e., used in the right-hand side of the corresponding assignments). Then the width of C is $w = \max\{w_1, \dots, w_d\}$.

Circuit width

Informally – circuit width is the biggest width (on one of the layers), that is the biggest number of gates on one layer.



Circuit width

The width is rather meaningless parameter, unless we put some other restrictions on the circuit. To characterize computation with limited space (e.g., the class PSPACE), one has to use either Turing machines, or some class of circuits with regular structure.

Circuit width reduction

Let C be a formula of depth d that computes a Boolean function $f(x_1, \dots, x_n)$. It is possible to construct a circuit of size $\exp(O(d))$ and width $O(1)$ that computes the same function.

Circuit width

There is a Boolean function f which cannot be computed by any circuit of size $O(n)$ and width $o(n)$.

https://arxiv.org/abs/1811.01347

Thank you for your attention!