CISC-102
Winter 2020
Week 4

Functions

We have already seen functions in this course. For example:

$$x^2 - 4x + 3$$

We could also write this function as an equation:

$$y = x^2 - 4x + 3$$

In this example you can think of plugging in a (Real) value for x and you will get a distinct value for y. So functions can be viewed as a *mapping* or a *transformation* or even some kind of *machine or algorithm* that takes an input an produces a distinct output.

Underlying every function are two sets (the two sets can be the same).

Let A and B these two sets. We define a function f from A into B as a mapping from every element of A to one element of B. This can be written as:

$$f : A \rightarrow B$$

Vocabulary

Suppose f is a function from the set A to the set B. Then we say that A is the *domain* of f and B is the *codomain* of f. (Synonyms for codomain are: *target set* and *range*)

Notation

Let f denote a function from A to B, then we write:

$$f : A \to B$$

which is pronounced "f is a function from A to B",

or "f maps A into B".

If $a \in A$, and $b \in B$ we can write:

$$f(a) = b$$

to denote that the function f maps the element a to b.

More Vocabulary

We can say that b is the *image* of a under f.

More notation.

A function can be expressed by a formula (written as an equation, as illustrated by the following example:

$$f(x) = x^2 \text{ for } x \in \mathbb{R}$$

In this example f is the function and x is the variable.

Sometimes we can express the image of a variable (the *independent variable*) by a *dependent variable* as follows:

$$y = x^2$$

Some Common Functions

Section 3.4 of Schaum’s notes describes some common functions that you may already be familiar with. Please read through this section. We will discuss modular arithmetic in more details when we look at the properties of the integers and integer arithmetic.

Sequences and indexed classes of sets

We discussed indexed sets and the generalized union and intersection operators. For example:

Let A_i denote the set $\{x : x \in \mathbb{Z}, x \geq i\}$, for all $i \in \mathbb{N}$.

$$\bigcup_{i \in \mathbb{N}} A_i = \mathbb{N}$$

$$\bigcap_{i \in \mathbb{N}} A_i = \emptyset$$

These indexed sets are defined by using an ***indexing function.***

Let I be any nonempty set, and let S be a collection of sets. (Or a set of sets.) An indexing function from I to S :

$$f : I \rightarrow S$$

maps indexes to sets in the collection. That is, for any $i \in I$ we denote an image $f(i)$ by A_i.

A *sequence* can be defined as a function from the natural numbers $\mathbb{N}$ into some set A. The notation a_n is used to denote the image of the number n.

As a concrete example consider a function f on the natural numbers defined as

$$f(n) = 2n$$

an equivalent sequence definition would be

$$a_n = 2n.$$

Suppose we want to denote the sum of the first k values of this function or sequence.

We could use "Sigma" notation as follows:

$$\sum_{i=1}^{k} f(i) = f(1) + f(2) + \cdots + f(k) = 2 + 4 + 6 + \cdots + 2k$$

or alternately

$$\sum_{i=1}^{k} a_i = a_1 + a_2 + \cdots + a_k = 2 + 4 + 6 + \cdots + 2k$$

Recursively Defined Functions

Recall the factorial function, $n!$. We can define n! and (n+1)! using these explicit iterative formulae:

$n!$ $= 1 \times 2 \times 3 \times \cdots \times n$
$(n+1)!$ $= 1 \times 2 \times 3 \times \cdots \times n \times (n+1)$

Notice how $(n+1)! = n! \times (n+1)$. This is a recursive definition of the factorial function. More formally we have the following definition.

The Factorial function is defined for non-negative integers, that is {0, 1, 2, 3, …} as follows:

(i) If $n = 0$ then $n! = 1$ (Base)
(ii) If $n > 0$ then $n! = n \times (n-1)!$ (Recursive definition)

Definition: (from SN) A function is said to be recursively defined if it has the following two properties:

i) There must be base values that are given and where the function does not refer to itself.
ii) Each time the function does refer to itself the referred function argument must be closer to the base than the referring function argument.

In the factorial definition (n-1) is closer to 0, than n is.

We can use a recursive definition for the handshake problem.

Suppose that S is a set consisting of *n* elements, $n \geq 2$.
Q. How many two element subsets are there of the set S?

We need to come up with a base statement and a recursive definition.

The recursive definition is based on the observation, a set of *n* elements has *n*-1 more two element subsets than a set of *n-1* elements.

Let f be a function with domain {2,3,4, …} and range $\mathbb{N}$, such that:

i) f(2) = 1 (1 two element subset}
ii) $f(n) = f(n-1) + n-1$, for $n \in \mathbb{N}, n \geq 3$.

To see why this counts handshakes, suppose we know how many there are with *n-1* people, the "new" n^{th} person adds n-1 to the count by shaking hands with the n-1 others.

(This same reasoning leads to the sum $\sum_{i=1}^{n} i$)

We can now use mathematical induction to prove that $f(n) = n(n-1)/2$, for all $n \in \mathbb{N}, n \geq 2$.

Note: the function is defined for all n, starting at n = 2, even though the recursive calls start at n = 3. The base case in the proof is at n = 2, and not n = 3.

The function recursively defined as:
f(2) = 1,
f(n) = f(n-1) + n-1, for $n \in \mathbb{N}, n \geq 3$.
has the closed form expression f(n) = n(n-1)/2, for all natural numbers n, n ≥ 2.

We prove this using mathematical induction.

Base: f(2) = 1 = 2(2-1)/2.
Induction Hypothesis: f(k) = k(k-1)/2 for a fixed natural number k, k ≥ 2.
Induction Step: f(k+1) = f(k) + k
= k(k-1)/2 + k
= $(k^2 - k + 2k)/2$
= $(k^2 + k)/2$
= (k+1)(k)/2
Therefore by the principle of mathematical induction we conclude that f(n) = n(n-1)/2 for all natural numbers n, n ≥ 2. □

We can use a recursive definition for the number of values that can be stored in a binary string. The recursive definition is based on the observation that an n bit binary number stores twice as many values as an (n-1) bit binary number.

Let f be a function on the the Natural numbers such that:

i) $f(1) = 2$ (2 values can be stored in one bit)
ii) $f(n) = f(n-1) \times 2$, for $n \geq 2$.

We can show using mathematical induction that the closed form for the recursive function is 2^n.

Let P(n) be the proposition that $f(n) = 2^n$, where f(n) is recursively defined as:
i) $f(1) = 2$
ii) $f(n) = 2f(n-1)$, for $n \geq 2$.

Theorem: $f(n) = 2^n$ for all natural numbers n.
Proof:
Base: $f(1) = 2 = 2^1$
Induction Hyp.: $f(k) = 2^k$, for a fixed value $k \in \mathbb{N}, k \geq 1$.
Induction Step: $\mathbf{f(k+1) = 2}\ f(k)$

$$= 2 \times 2^k$$
$$= 2^{k+1}.$$

Therefore by the principle of mathematical induction we conclude that P(n) is true for all natural numbers n. □

The recursive function:

i) $f(1) = 2$.
ii) $f(n) = 2\ f(n-1)\ n \geq 2$.

Can be viewed as a recursive definition of 2^n.

Can you guess what the following recursive function defines?

i) $g(1) = 5$.
ii) $g(n) = 5\ g(n-1)\ n \geq 2$.

How about this one?

i) $h(1) = x$.
ii) $h(n) = (x)h(n-1)\ n \geq 2$.

Consider a function recursively defined as:

$g(1) = 1, g(n) = g(n-1) + 2n-1, n \geq 2.$

What is the value of $g(2)$, $g(3)$, $g(4)$?
Using the values of $g(2)$, $g(3)$, $g(4)$, can you guess the value of $g(n)$?

Let g(n) be a function recursively defined as
$g(1) = 1$, $g(n) = g(n-1) + 2n-1$, $n \geq 2$.
Then $g(n) = n^2$, for all natural numbers n.

Base: $g(1) = 1 = 1^2$

Induction Hypothesis: $g(k) = k^2$ for some fixed $k \in \mathbb{N}$, $k \geq 1$.

Induction Step:
$$\begin{aligned} g(k+1) &= g(k) + 2(k+1) - 1 \\ &= k^2 + 2k + 1 \\ &= (k+1)^2 \end{aligned}$$

Therefore by the principle of mathematical induction we have shown that $g(n) = n^2$ for all natural numbers n. □

(The recursive definition just says that $n^2 = (n-1)^2 + 2n - 1$. This can be confirmed by doing a bit of algebra.)

Injective(one-to-one), Surjective(onto), Bijective(one-to-one and onto) functions.

A function f: A → B is a *one-to-one* function if for every

a ∈ A there is a distinct image in B. A one-to-one function is also called an *injective function* or

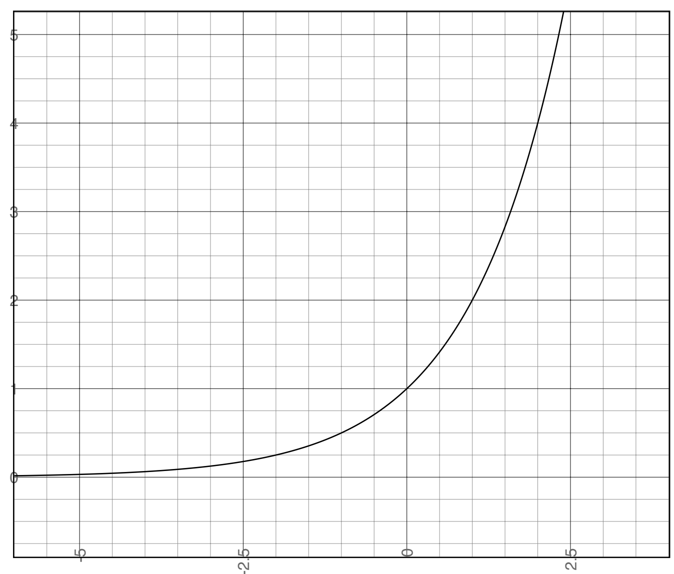
an *injection*. Another way to say this is $f(a_1) \neq f(a_2)$ if $a_1 \neq a_2$

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and $f(x) = 2^x$.

$f(x) = 2^x$ is one-to-one because

there is a distinct image for every

$x \in \mathbb{R}$, that is if $2^x = 2^y$ then $x = y$.

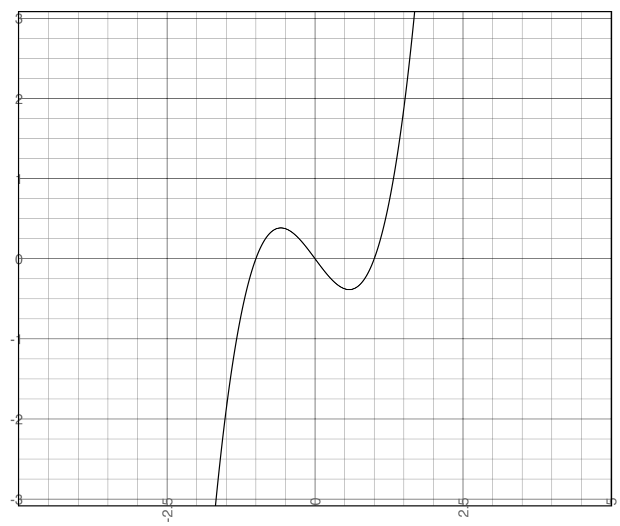


A function f: A → B is an *onto* function if

every $b \in$ B is an image. An onto function is also called a *surjective function* or a *surjection*.

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and $f(x) = x^3 - x$.

$f(x) = x^3 - x$ is onto because the pre-image of any real number y is the solution set of the cubic polynomial equation $x^3 - x - y = 0$ and every cubic polynomial with real coefficients has at least one real root.



Both one-to-one, and onto.

Note: $f(x) = x^3 - x = x(x^2 - 1)$ is **not** one-to-one because $f(x) = 0$ for $x = -1, x = +1, x = 0$

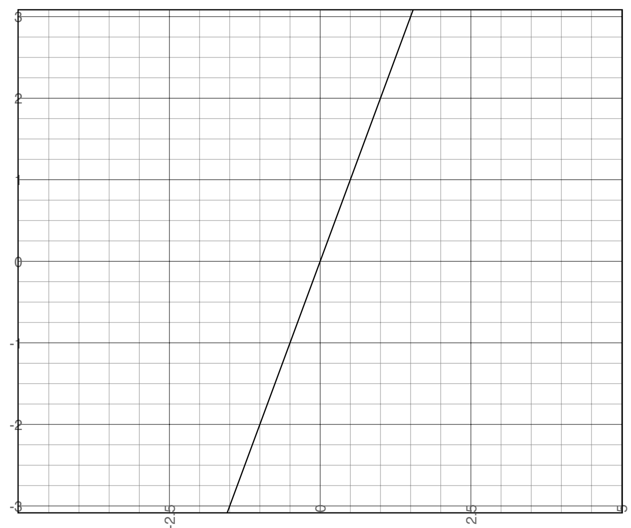
Note: $f(x) = 2^x$ is **not** onto because $2^x > 0$ for all $x \in \mathbb{R}$.

A function that is both one-to-one and onto is called a ***bijective function*** or a ***bijection***.

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and $f(x) = 2x$

$f(x) = 2x$ is one-to-one because we get a distinct image for every pre-image.

$f(x) = 2x$ is onto because every $y \in \mathbb{R}$ is an image. So $f(x) = 2x$ is a bijection.

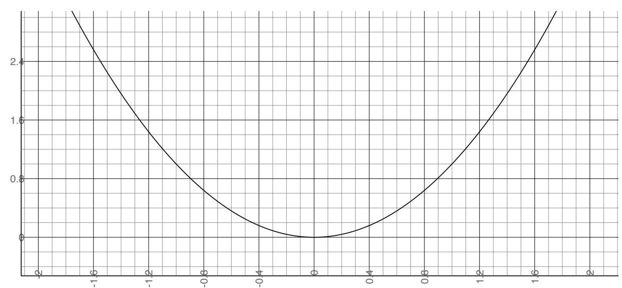


Bijective functions are also called *__invertible__* functions. That is suppose that f is a bijective function on the set A. Then f^{-1} denotes the inverse of the function f , meaning that whenever $f(x) = y$ we have $f^{-1}(y) = x$.

In our previous example we saw that function $f(x) = 2x$ is a bijective function. In this case we can define

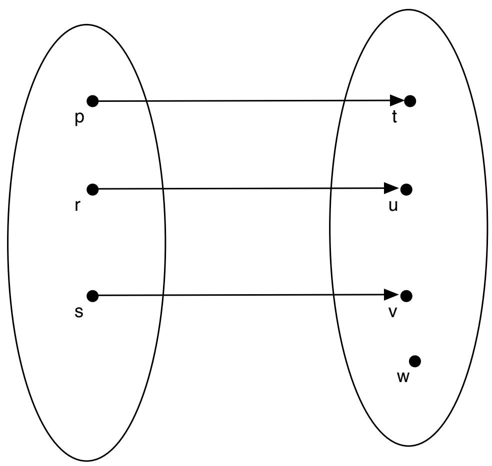
$f^{-1}(x) = x/2$, so we get $f^{-1}(2x) = x$.

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and $f(x) = x^2$

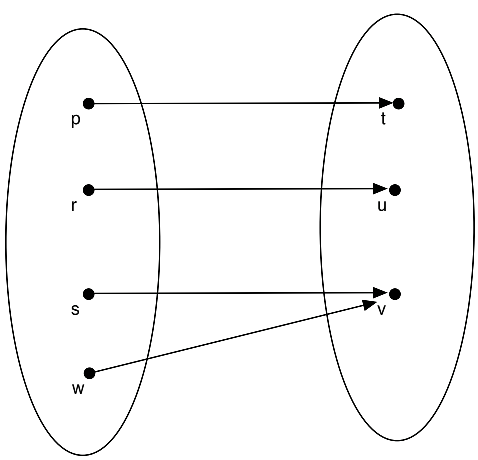


Observe that $f(x) = x^2$ is a function because every $x \in \mathbb{R}$ has a distinct image. However, $f(x) = x^2$ is neither one-to-one (because $f(x) = f(-x)$) or onto ($f(x) \geq 0$).

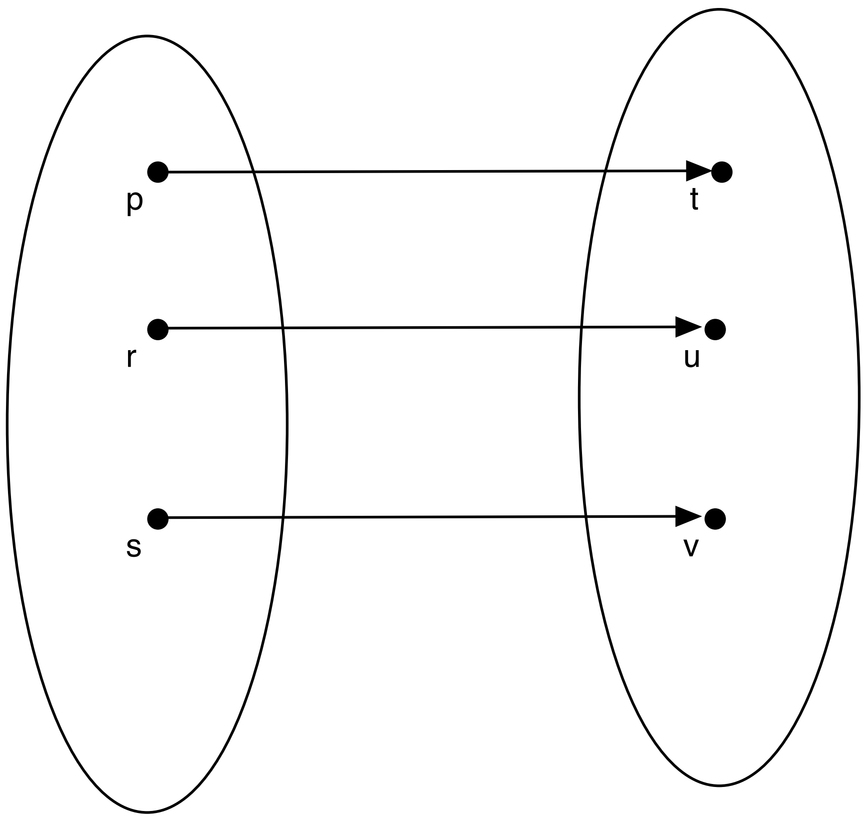
Pictorial examples:



Illustrates a function that is one-to-one, but not onto.



Illustrates a function that is onto, but not one-to-one.


p
t
r
u
s
v

Determine whether the following functions are one-to-one, onto, both, or neither.

$f : \mathbb{Z} \to \mathbb{Z}$, $f(x) = x + 5$

$f : \mathbb{N} \to \mathbb{N}$, $f(x) = x + 5$

$f : \mathbb{Z} \to \mathbb{Z}$, $f(x) = x^2 + 5$

$f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2 + 5$

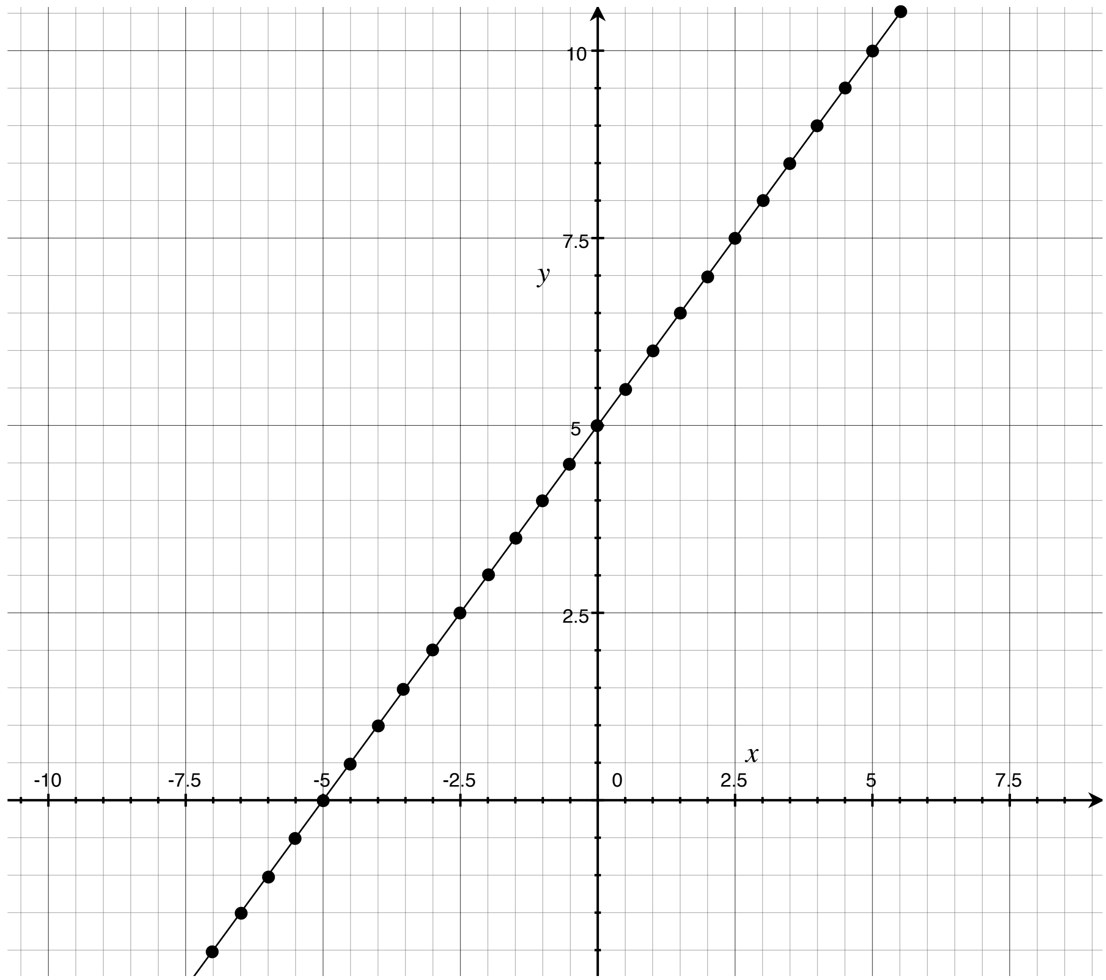
$f : \mathbb{N} \to \mathbb{N}$, $f(1) = 1$, $f(x) = f(x-1) \times x$ for $x > 1$

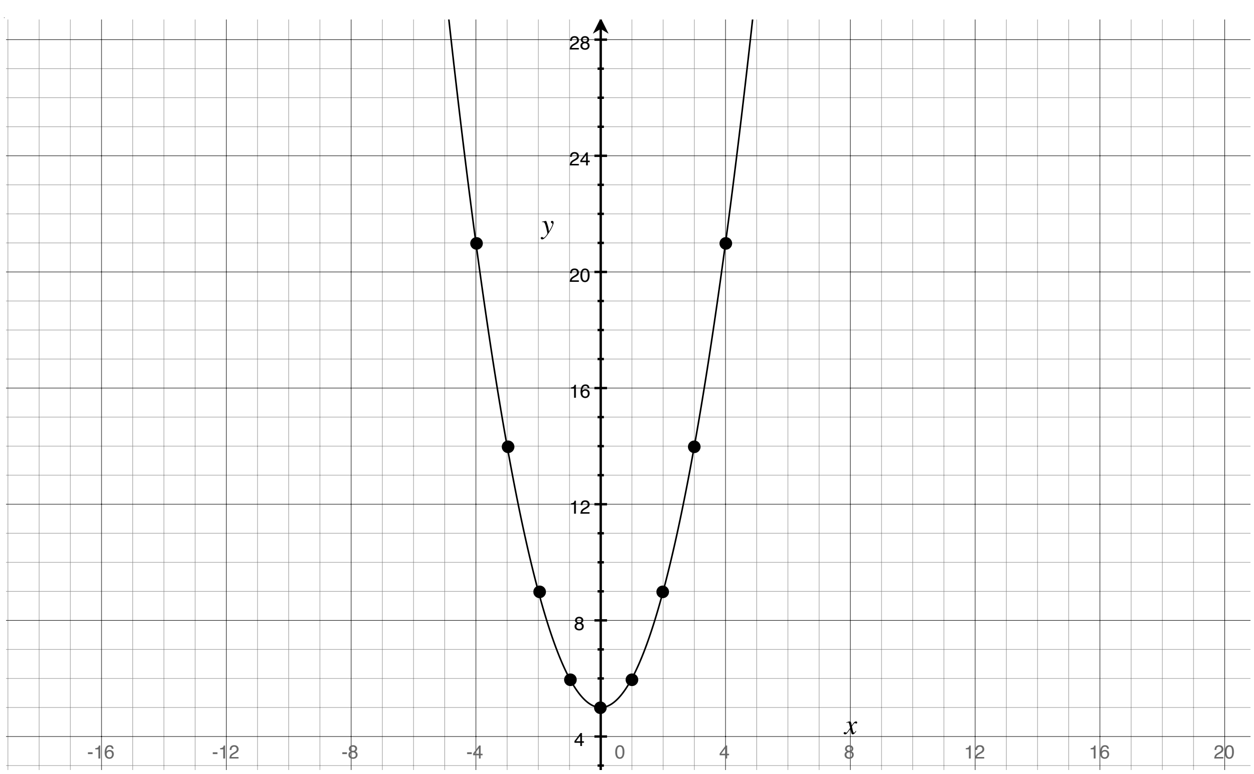
Let S denote the set of binary strings of length n, and let $T = \{x : x \in (\mathbb{N} \cup \{0\}),\ 0 \leq x \leq 2^n\}$

f: $S \to T$, f(s) = numeric value of binary string s.

Let S denote the set of binary strings of length n, and let $T = \{x : x \in \mathbb{Z},\ 0 \leq x \leq 2^n - 1\}$

f: $S \to T$, f(s) = numeric value of binary string s.


10
7.5
y
5
2.5
x
-10
-7.5
-5
-2.5
0
2.5
5
7.5


28
24
y
20
16
12
8
4
x
-16
-12
-8
-4
0
4
8
12
16
20