class sage.rings.semirings.non_negative_integer_semiring.NonNegativeIntegerSemiring
    Bases: sage.sets.non_negative_integers.NonNegativeIntegers

A class for the semiring of the non negative integers

This parent inherits from the infinite enumerated set of non negative integers and endows it with its natural
semiring structure.

EXAMPLES:

    sage: NonNegativeIntegerSemiring()
    Non negative integer semiring

For convenience, NN is a shortcut for NonNegativeIntegerSemiring():

    sage: NN == NonNegativeIntegerSemiring()
    True
    sage: NN.category()
    Category of facade infinite enumerated commutative semirings

Here is a piece of the Cayley graph for the multiplicative structure:

    sage: G = NN.cayley_graph(elements=range(9), generators=[0,1,2,3,5,7])
    sage: G
    Looped multi-digraph on 9 vertices
    sage: G.plot()
    Graphics object consisting of 48 graphics primitives

This is the Hasse diagram of the divisibility order on NN.

    sage: Poset(NN.cayley_graph(elements=[1..12], generators=[2,3,5,7,11])).show()

Note: as for NonNegativeIntegers, NN is currently just a “facade” parent; namely its elements are plain
Sage Integers with Integer Ring as parent:

    sage: x = NN(15); type(x)
    <type 'sage.rings.integer.Integer'>
    sage: x.parent()
    Integer Ring
    sage: x+3
    18

    additive_semigroup_generators()
    Returns the additive semigroup generators of self.
EXAMPLES:

```
sage: NN.additive_semigroup_generators()
Family (0, 1)
```
TROPICAL SEMIRINGS

AUTHORS:

• Travis Scrimshaw (2013-04-28) - Initial version

```
class sage.rings.semirings.tropical_semiring.TropicalSemiring(base, use_min=True):
    Bases: sage.structure.parent.Parent, sage.structure.unique_representation.UniqueRepresentation

The tropical semiring.
Given an ordered additive semigroup \( R \), we define the tropical semiring \( T = R \cup \{+\infty\} \) by defining tropical addition and multiplication as follows:

\[
a \oplus b = \min(a, b), \quad a \otimes b = a + b.
\]

In particular, note that there are no (tropical) additive inverses (except for \( \infty \)), and every element in \( R \) has a (tropical) multiplicative inverse.

There is an alternative definition where we define \( T = R \cup \{-\infty\} \) and alter tropical addition to be defined by

\[
a \oplus b = \max(a, b).
\]

To use the max definition, set the argument `use_min = False`.

**Warning:** `zero()` and `one()` refer to the tropical additive and multiplicative identities respectively. These are **not** the same as calling `T(0)` and `T(1)` respectively as these are **not** the tropical additive and multiplicative identities respectively.

Specifically do not use `sum(...)` as this converts 0 to 0 as a tropical element, which is not the same as `zero()`. Instead use the `sum` method of the tropical semiring:

```
sage: T = TropicalSemiring(QQ)
sage: sum([T(1), T(2)])  # This is wrong
0
sage: T.sum([T(1), T(2)])  # This is correct
1
```

Be careful about using code that has not been checked for tropical safety.

**INPUT:**

• `base` – the base ordered additive semigroup \( R \)
• use_min – (default: True) if True, then the semiring uses \( a \oplus b = \min(a, b) \); otherwise uses \( a \oplus b = \max(a, b) \)

EXEMPLARY:

```
sage: T = TropicalSemiring(QQ)
sage: elt = T(2); elt
2
```

Recall that tropical addition is the minimum of two elements:

```
sage: T(3) + T(5)
3
```

Tropical multiplication is the addition of two elements:

```
sage: T(2) * T(3)
5
sage: T(0) * T(-2)
-2
```

We can also do tropical division and arbitrary tropical exponentiation:

```
sage: T(2) / T(1)
1
sage: T(2)^(3/7)
-6/7
```

Note that “zero” and “one” are the additive and multiplicative identities of the tropical semiring. In general, they are not the elements 0 and 1 of \( R \), respectively, even if such elements exist (e.g., for \( R = \mathbb{Z} \)), but instead the (tropical) additive and multiplicative identities \( +\infty \) and 0 respectively:

```
sage: T.zero() + T(3) == T(3)
True
sage: T.one() * T(3) == T(3)
True
sage: T.zero() == T(0)
False
sage: T.one() == T(1)
False
```

**Element**

alias of *TropicalSemiringElement*

**additive_identity()**

Return the (tropical) additive identity element \( +\infty \).

EXEMPLARY:

```
sage: T = TropicalSemiring(QQ)
sage: T.zero() +infinity
```

gens()

Return the generators of `self`.

EXEMPLARY:
sage: T = TropicalSemiring(QQ)
sage: T.gens()
(1, +infinity)

infinity()
Return the (tropical) additive identity element +∞.

EXAMPLES:

sage: T = TropicalSemiring(QQ)
sage: T.zero()
+infinity

multiplicative_identity()
Return the (tropical) multiplicative identity element 0.

EXAMPLES:

sage: T = TropicalSemiring(QQ)
sage: T.one()
0

one()
Return the (tropical) multiplicative identity element 0.

EXAMPLES:

sage: T = TropicalSemiring(QQ)
sage: T.one()
0

zero()
Return the (tropical) additive identity element +∞.

EXAMPLES:

sage: T = TropicalSemiring(QQ)
sage: T.zero()
+infinity

class sage.rings.semirings.tropical_semiring.TropicalSemiringElement

Bases: sage.structure.element.Element

An element in the tropical semiring over an ordered additive semigroup \( R \). Either in \( R \) or +∞. The operators +, · are defined as the tropical operators ⊕, ⊙ respectively.

lift()
Return the value of self lifted to the base.

EXAMPLES:

sage: T = TropicalSemiring(QQ)
sage: elt = T(2)
sage: elt.lift() 2
sage: elt.lift().parent() is QQ
True
sage: T.additive_identity().lift().parent()
The Infinity Ring
multiplicative_order()

Return the multiplicative order of self.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.multiplicative_identity().multiplicative_order()
1
sage: T.additive_identity().multiplicative_order()
+Infinity
```

class sage.rings.semirings.tropical_semiring.TropicalToTropical

Map from the tropical semiring to itself (possibly with different bases). Used in coercion.
CHAPTER THREE

INDICES AND TABLES

- Index
- Module Index
- Search Page
r

sage.rings.semirings.non_negative_integer_semiring, 1
sage.rings.semirings.tropical_semiring, 3
INDEX

A
additive_identity() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 4
additive_semigroup_generators() (sage.rings.semirings.non_negative_integer_semiring.NonNegativeIntegerSemiring method), 1

E
Element (sage.rings.semirings.tropical_semiring.TropicalSemiring attribute), 4

G
gens() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 4

I
infinity() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 5

L
lift() (sage.rings.semirings.tropical_semiring.TropicalSemiringElement method), 5

M
multiplicative_identity() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 5
multiplicative_order() (sage.rings.semirings.tropical_semiring.TropicalSemiringElement method), 5

N
NonNegativeIntegerSemiring (class in sage.rings.semirings.non_negative_integer_semiring), 1

O
one() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 5

S
sage.rings.semirings.non_negative_integer_semiring (module), 1
sage.rings.semirings.tropical_semiring (module), 3

T
TropicalSemiring (class in sage.rings.semirings.tropical_semiring), 3
TropicalSemiringElement (class in sage.rings.semirings.tropical_semiring), 5
TropicalToTropical (class in sage.rings.semirings.tropical_semiring), 6

Z
zero() (sage.rings.semirings.tropical_semiring.TropicalSemiring method), 5