PyRival

Release 0.2.0.post60+g7f54a87

Jul 09, 2023

Contents:

CHAPTER 1

Infinite Recursion

Infinite recursion can be achieved by using the `pyrival.misc.bootstrap` decorator

To use it you will need to make a few modifications to the recursive function:

- Change all `return` to `yield`
- Add `yield` before recursive function calls

For example the following code

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(10))    # prints 3628800
print(factorial(1000)) # exceeds recursion limit
```

will be changed to the following

```
import pyrival.misc

@pyrival.misc.bootstrap
def factorial(n):
    if n == 0:
        yield 1
    else:
        yield n * (yield factorial(n - 1))

print(factorial(10))     # prints 3628800
print(factorial(1000)    # prints 402387...000000
print(factorial(10000)) # prints 284625...000000
```

CHAPTER 2

API Reference

2.1 pyrival.algebra

2.1.1 pyrival.algebra.chinese_remainder

pyrival.algebra.chinese_remainder.**chinese_remainder**(*a*, *p*)
returns x s.t. x = a[i] (mod p[i]) where p[i] is prime for all i

pyrival.algebra.chinese_remainder.**composite_crt**(*b*, *m*)
returns x s.t. x = b[i] (mod m[i]) for all i

pyrival.algebra.chinese_remainder.**extended_gcd**(*a*, *b*)
returns gcd(a, b), s, r s.t. a * s + b * r == gcd(a, b)

pyrival.algebra.chinese_remainder.**gcd**(*x*, *y*)
greatest common divisor of x and y

2.1.2 pyrival.algebra.discrete_log

pyrival.algebra.discrete_log.**discrete_log**(*a*, *b*, *mod*)
Returns smallest x > 0 s.t. pow(a, x, mod) == b or None if no such x exists. Note: works even if a and mod are not coprime.

2.1.3 pyrival.algebra.factors

pyrival.algebra.factors.**all_factors**(*n*)
returns a sorted list of all distinct factors of n

pyrival.algebra.factors.**distinct_factors**(*n*)
returns a list of all distinct factors of n

pyrival.algebra.factors.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.algebra.factors.**memodict**(*f*)
memoization decorator for a function taking a single argument

pyrival.algebra.factors.**pollard_rho**(*n*)
returns a random factor of n

pyrival.algebra.factors.**prime_factors**()
x.__getitem__(y) <==> x[y]

2.1.4 pyrival.algebra.fft

pyrival.algebra.fft.**fft**(*P*)

pyrival.algebra.fft.**fft_conv**(*P*, *Q*)

pyrival.algebra.fft.**ifft**(*P*)

2.1.5 pyrival.algebra.fst

pyrival.algebra.fst.**fst**(*a*, *oplus=<built-in function and_>*, *inv=False*)

pyrival.algebra.fst.**fst_conv**(*a*, *b*)

2.1.6 pyrival.algebra.gcd

pyrival.algebra.gcd.**extended_gcd**(*a*, *b*)
returns gcd(a, b), s, r s.t. a * s + b * r == gcd(a, b)

pyrival.algebra.gcd.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.algebra.gcd.**gcdm**(**args*)

pyrival.algebra.gcd.**lcm**(*a*, *b*)

pyrival.algebra.gcd.**lcmm**(**args*)

2.1.7 pyrival.algebra.is_prime

pyrival.algebra.is_prime.**is_prime**(*n*)
returns True if n is prime else False

2.1.8 pyrival.algebra.mod_sqrt

pyrival.algebra.mod_sqrt.**mod_sqrt**(*a*, *p*)
returns x s.t. x**2 == a (mod p)

2.1.9 pyrival.algebra.modinv

pyrival.algebra.modinv.**extended_gcd**(*a*, *b*)
returns gcd(a, b), s, r s.t. a * s + b * r == gcd(a, b)

pyrival.algebra.modinv.**modinv**(*a*, *m*)
returns the modular inverse of a w.r.t. to m, works when a and m are coprime

2.1.10 pyrival.algebra.ntt

pyrival.algebra.ntt.**intt**(*P*)

pyrival.algebra.ntt.**ntt**(*P*)

pyrival.algebra.ntt.**ntt_conv**(*P*, *Q*)

2.1.11 pyrival.algebra.phi

pyrival.algebra.phi.**phi**(*n*)
returns phi(x) for all x <= n

2.1.12 pyrival.algebra.primitive_root

pyrival.algebra.primitive_root.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.algebra.primitive_root.**ilog**(*n*)
returns the smallest a, b s.t. a**b = n for integer a, b

pyrival.algebra.primitive_root.**memodict**(*f*)
memoization decorator for a function taking a single argument

pyrival.algebra.primitive_root.**pollard_rho**(*n*)
returns a random factor of n

pyrival.algebra.primitive_root.**prime_factors**()
x.__getitem__(y) <==> x[y]

pyrival.algebra.primitive_root.**primitive_root**(*p*)
returns a primitive root of p

2.1.13 pyrival.algebra.sieve

pyrival.algebra.sieve.**prime_list**(*n*)
returns a list of primes <= n

pyrival.algebra.sieve.**prime_sieve**(*n*)
returns a sieve of primes >= 5 and < n

2.2 pyrival.combinatorics

2.2.1 pyrival.combinatorics.combinatorics

pyrival.combinatorics.combinatorics.**bell**(*n*)

pyrival.combinatorics.combinatorics.**catalan**(*n*)

pyrival.combinatorics.combinatorics.**derangements**(*n*)

pyrival.combinatorics.combinatorics.**euler**(*n*, *k*)

pyrival.combinatorics.combinatorics.**memoize**(*f*)
memoization decorator for a function taking one or more arguments

pyrival.combinatorics.combinatorics.**multinomial**(*k*)

pyrival.combinatorics.combinatorics.**nCr**(*n*, *r*)

pyrival.combinatorics.combinatorics.**stirling_2**(*n*, *k*)

2.2.2 pyrival.combinatorics.nCr_mod

pyrival.combinatorics.nCr_mod.**make_nCr_mod**(*max_n=200000*, *mod=1000000007*)

2.2.3 pyrival.combinatorics.partitions

pyrival.combinatorics.partitions.**memoize**(*f*)
memoization decorator for a function taking one or more arguments

2.3 pyrival.data_structures

2.3.1 pyrival.data_structures.BitArray

class pyrival.data_structures.BitArray.**BitArray**(*size*)
Bases: object

implements bitarray using bytearray

2.3.2 pyrival.data_structures.CFraction

pyrival.data_structures.CFraction.**CFrac2Frac**(*cfrac*)

pyrival.data_structures.CFraction.**CFraction**(*frac*)

2.3.3 pyrival.data_structures.DisjointSetUnion

class pyrival.data_structures.DisjointSetUnion.**DisjointSetUnion**(*n*)
Bases: object

find(*a*)

set_size(*a*)

union(*a*, *b*)

class pyrival.data_structures.DisjointSetUnion.**UnionFind**(*n*)
Bases: object

find(*a*)

union(*a*, *b*)

2.3.4 pyrival.data_structures.FenwickTree

class pyrival.data_structures.FenwickTree.**FenwickTree**(*x*)
Bases: object

findkth(*k*)
Find largest idx such that sum(bit[:idx]) <= k

query(*end*)
calc sum(bit[:end])

update(*idx*, *x*)
updates bit[idx] += x

2.3.5 pyrival.data_structures.Fraction

class pyrival.data_structures.Fraction.**Fraction**(*num=0*, *den=1*)
Bases: object

pyrival.data_structures.Fraction.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.data_structures.Fraction.**limit_denominator**(*frac*, *max_den=1000000*)

2.3.6 pyrival.data_structures.Heap

class pyrival.data_structures.Heap.**Heap**(*iterable=None*, *reverse=False*)
Bases: object

peek()

pop()

poppush(*item*)

push(*item*)

pushpop(*item*)

replace(*item*)

class pyrival.data_structures.Heap.**OrderHeap**(*iterable=None*, *key=<function OrderHeap.<lambda>>*, *reverse=False*)
Bases: *pyrival.data_structures.Heap.Heap*

peek()

pop()

poppush(*item*)

push(*item*)

pushpop(*item*)

replace(*item*)

class pyrival.data_structures.Heap.**RemovalHeap**(*iterable=None*, *reverse=False*)
Bases: *pyrival.data_structures.Heap.Heap*

peek()

pop()

poppush(*item*)

push(*item*)

pushpop(*item*)

remove(*item*)

replace(*item*)

sweep()

class pyrival.data_structures.Heap.**XHeap**(*iterable=None*, *key=<function XHeap.<lambda>>*, *reverse=False*)

Bases: *pyrival.data_structures.Heap.Heap*

peek()

pop()

poppush(*item*)

push(*item*)

pushpop(*item*)

remove(*item*)

replace(*item*)

sweep()

2.3.7 pyrival.data_structures.LazySegmentTree

class pyrival.data_structures.LazySegmentTree.**LazySegmentTree**(*data*, *default=0*, *func=<built-in function max>*)

Bases: object

add(*start*, *stop*, *value*)
lazily add value to [start, stop)

query(*start*, *stop*, *default=0*)
func of data[start, stop)

2.3.8 pyrival.data_structures.LinkedList

class pyrival.data_structures.LinkedList.**LinkedList**(*iterable=None*)

Bases: object

after(*node*)

append(*value*)

appendleft(*value*)

before(*node*)

get_node(*index*)

insert(*index*, *value*)

insert_after(*node*, *value*)

insert_between(*node*, *left_node*, *right_node*)

merge_left(*other*)

merge_right(*other*)

pop(*node=None*)

to_list()

class pyrival.data_structures.LinkedList.**Node**(*value*)
Bases: object

2.3.9 pyrival.data_structures.Node

class pyrival.data_structures.Node.**Node**(*value*)
Bases: object

2.3.10 pyrival.data_structures.PersistentSegTree

pyrival.data_structures.PersistentSegTree.**create**(*n*)
create a persistant segment tree of size n

pyrival.data_structures.PersistentSegTree.**minimum**(*ind*, *l*, *r*, *n*)
find mimimum of set[l:r] for segment tree ind, of size n

pyrival.data_structures.PersistentSegTree.**setter**(*ind*, *i*, *val*, *n*)
set set[i] = val for segment tree ind, of size n

2.3.11 pyrival.data_structures.RangeQuery

class pyrival.data_structures.RangeQuery.**RangeQuery**(*data*, *func=<built-in function min>*)
Bases: object

query(*start*, *stop*)
func of data[start, stop)

2.3.12 pyrival.data_structures.SegmentTree

class pyrival.data_structures.SegmentTree.**SegmentTree**(*data*, *default=0*, *func=<built-in function max>*)
Bases: object

query(*start*, *stop*)
func of data[start, stop)

2.3.13 pyrival.data_structures.SortedList

The "sorted list" data-structure, with amortized O(n^(1/3)) cost per insert and pop.

Example:

A = SortedList() A.insert(30) A.insert(50) A.insert(20) A.insert(30) A.insert(30)

print(A) # prints [20, 30, 30, 30, 50]

print(A.lower_bound(30), A.upper_bound(30)) # prints 1 4

print(A[-1]) # prints 50 print(A.pop(1)) # prints 30

print(A) # prints [20, 30, 30, 50] print(A.count(30)) # prints 2

class pyrival.data_structures.SortedList.**FenwickTree**(*x*)
Bases: object

find_kth(*k*)
Find largest idx such that sum(bit[:idx]) <= k

update(*idx*, *x*)
updates bit[idx] += x

class pyrival.data_structures.SortedList.**SortedList**(*iterable=()*)
Bases: object

block_size = 700

count(*x*)

insert(*x*)

lower_bound(*x*)

pop(*k=-1*)

upper_bound(*x*)

2.3.14 pyrival.data_structures.Treap

class pyrival.data_structures.Treap.**TreapHashMap**(*data=None*)
Bases: *pyrival.data_structures.Treap.TreapMultiSet*

add(*key*)

discard(*key*)

get(*key*, *default=None*)

remove(*key*)

class pyrival.data_structures.Treap.**TreapHashSet**(*data=None*)
Bases: *pyrival.data_structures.Treap.TreapMultiSet*

add(*key*)

discard(*key*)

remove(*key*)

class pyrival.data_structures.Treap.**TreapMultiSet**(*data=None*)
Bases: object

add(*key*)

ceiling(*key*)

discard(*key*)

floor(*key*)

higher(*key*)

lower(*key*)

max()

min()

remove(*key*)

root = 0

size = 0

class pyrival.data_structures.Treap.**TreapSet**(*data=None*)
Bases: *pyrival.data_structures.Treap.TreapMultiSet*

add(*key*)

pyrival.data_structures.Treap.**treap_builder**(*sorted_data*)
Build a treap in O(n) time using sorted data

pyrival.data_structures.Treap.**treap_ceiling**(*root*, *key*)

pyrival.data_structures.Treap.**treap_create_node**(*key*)

pyrival.data_structures.Treap.**treap_erase**(*root*, *key*)

pyrival.data_structures.Treap.**treap_floor**(*root*, *key*)

pyrival.data_structures.Treap.**treap_higher**(*root*, *key*)

pyrival.data_structures.Treap.**treap_insert**(*root*, *key*)

pyrival.data_structures.Treap.**treap_insert_unique**(*root*, *key*)

pyrival.data_structures.Treap.**treap_lower**(*root*, *key*)

pyrival.data_structures.Treap.**treap_max**(*root*)

pyrival.data_structures.Treap.**treap_merge**(*left*, *right*)

pyrival.data_structures.Treap.**treap_min**(*root*)

pyrival.data_structures.Treap.**treap_split**(*root*, *key*)

2.3.15 pyrival.data_structures.Trie

class pyrival.data_structures.Trie.**Trie**(**words*)
Bases: object

add(*word*)

2.3.16 pyrival.data_structures.TwoSat

class pyrival.data_structures.TwoSat.**TwoSat**(*n*)
Bases: object

either(*x*, *y*)
either x or y must be True

set(*x*)
x must be True

solve()

pyrival.data_structures.TwoSat.**find_SCC**(*graph*)

2.3.17 pyrival.data_structures.convex_hull_trick

pyrival.data_structures.convex_hull_trick.**convex_hull_trick**(*K*, *M*, *integer=True*)
Given lines on the form y = K[i] * x + M[i] this function returns intervals, such that on each interval the convex hull is made up of a single line. Input:

K: list of the slopes M: list of the constants (value at x = 0) interger: boolean for turning on / off integer mode. Integer mode is exact, it

works by effectively flooring the seperators of the intervals.

Return: hull_i: on interval j, line i = hull_i[j] is >= all other lines hull_x: interval j and j + 1 is separated by x = hull_x[j], (hull_x[j] is the last x in interval j)

pyrival.data_structures.convex_hull_trick.**max_query**(*x*, *K*, *M*, *hull_i*, *hull_x*)
Find maximum value at x in O(log n) time

2.3.18 pyrival.data_structures.tree_repr

pyrival.data_structures.tree_repr.**tree_repr**(*tree*)

2.4 pyrival.geometry

2.4.1 pyrival.geometry.convex_hull

pyrival.geometry.convex_hull.**convex_hull**(*points*)

pyrival.geometry.convex_hull.**remove_middle**(*a*, *b*, *c*)

2.4.2 pyrival.geometry.lines

pyrival.geometry.lines.**collinear**(*p1*, *p2*, *p3*)

pyrival.geometry.lines.**dist**(*p1*, *p2*)

pyrival.geometry.lines.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.geometry.lines.**get_2dline**(*p1*, *p2*)

pyrival.geometry.lines.**get_line**(*p1*, *p2*)

pyrival.geometry.lines.**intersect**(*l1*, *l2*)

pyrival.geometry.lines.**is_parallel**(*l1*, *l2*)

pyrival.geometry.lines.**is_same**(*l1*, *l2*)

pyrival.geometry.lines.**rotate**(*p*, *theta*, *origin=(0, 0)*)

2.4.3 pyrival.geometry.polygons

pyrival.geometry.polygons.**area**(**p*)

pyrival.geometry.polygons.**circumcircle_radius**(*a*, *b*, *c*)

pyrival.geometry.polygons.**dist**(*p1*, *p2*)

pyrival.geometry.polygons.**incircle_radius**(*a*, *b*, *c*)

pyrival.geometry.polygons.**is_in_circle**(*p*, *c*, *r*)

pyrival.geometry.polygons.**perimeter**(**p*)

2.4.4 pyrival.geometry.vectors

pyrival.geometry.vectors.**angle**(*oa*, *ob*)

pyrival.geometry.vectors.**closest_point**(*p*, *a*, *b*, *segment=False*)

pyrival.geometry.vectors.**cross2d**(*v1*, *v2*)

pyrival.geometry.vectors.**cross3d**(*v1*, *v2*)

pyrival.geometry.vectors.**dot**(*v1*, *v2*)

pyrival.geometry.vectors.**norm_sq**(*v*)

pyrival.geometry.vectors.**scale**(*v*, *s*)

pyrival.geometry.vectors.**to_vec**(*p1*, *p2*)

pyrival.geometry.vectors.**translate**(*p*, *v*)

2.5 pyrival.graphs

2.5.1 pyrival.graphs.bellman_ford

pyrival.graphs.bellman_ford.**bellman_ford**(*n*, *edges*, *start*)

2.5.2 pyrival.graphs.bfs

pyrival.graphs.bfs.**bfs**(*graph*, *start=0*)

pyrival.graphs.bfs.**layers**(*graph*, *start=0*)

2.5.3 pyrival.graphs.components

pyrival.graphs.components.**connected_components**(*n*, *graph*)

2.5.4 pyrival.graphs.cycle_finding

pyrival.graphs.cycle_finding.**cycle_finding**(*f*, *x0*)

2.5.5 pyrival.graphs.dfs

pyrival.graphs.dfs.**dfs**(*graph*, *start=0*)

2.5.6 pyrival.graphs.dijkstra

pyrival.graphs.dijkstra.**dijkstra**(*graph*, *start*)

Uses Dijkstra's algortihm to find the shortest path from node start to all other nodes in a directed weighted graph.

2.5.7 pyrival.graphs.dinic

class pyrival.graphs.dinic.**Dinic**(*n*)

Bases: object

add_edge(*a*, *b*, *c*, *rcap=0*)

calc(*s*, *t*)

dfs(*v*, *t*, *f*)

2.5.8 pyrival.graphs.euler_walk

pyrival.graphs.euler_walk.**euler_walk**(*n*, *adj*)

2.5.9 pyrival.graphs.find_path

pyrival.graphs.find_path.**find_path**(*start*, *end*, *parents*)

Constructs a path between two vertices, given the parents of all vertices.

2.5.10 pyrival.graphs.floyd_warshall

pyrival.graphs.floyd_warshall.**floyd_warshall**(*n*, *edges*)

2.5.11 pyrival.graphs.hopcroft_karp

Produces a maximum cardinality matching of a bipartite graph

Example:

0—0

1—1

/ /

/ 2 2

/

/

/

3

>>>> n = 4 >>>> m = 3 >>>> graph = [[0, 1], [1, 2], [1], [2]] >>>> match1, match2 = hopcroft_karp(graph, n, m) >>>> match1 [0, 1, -1, 2] >>>> match2 [0, 1, 3]

Meaning 0—0

1—1

2 2

/

/

/

3

pyrival.graphs.hopcroft_karp.**hopcroft_karp**(*graph*, *n*, *m*)
Maximum bipartite matching using Hopcroft-Karp algorithm, running in O(|E| sqrt(|V|))

2.5.12 pyrival.graphs.is_bipartite

pyrival.graphs.is_bipartite.**is_bipartite**(*graph*)

2.5.13 pyrival.graphs.kruskal

class pyrival.graphs.kruskal.**UnionFind**(*n*)
Bases: object

find(*a*)

merge(*a*, *b*)

pyrival.graphs.kruskal.**kruskal**(*n*, *U*, *V*, *W*)

2.5.14 pyrival.graphs.lca

class pyrival.graphs.lca.**LCA**(*root*, *graph*)
Bases: object

class pyrival.graphs.lca.**RangeQuery**(*data*, *func=<built-in function min>*)
Bases: object

query(*begin*, *end*)

2.5.15 pyrival.graphs.maximum_matching

pyrival.graphs.maximum_matching.**maximum_matching**(*edges*, *mod=1073750017*)
Returns the maximum cardinality matching of any simple graph (undirected, unweighted, no self-loops) Uses a randomized algorithm to compute the rank of the Tutte matrix The rank of the Tutte matrix is equal to twice the size of the maximum matching with high probability The probability for error is not more than n/mod

Complexity: O(n ^ 3) worst case, O(n * |matching_size|) on average

Parameters

- **edges** – a list of edges, assume nodes can be anything numbered from 0 to max number in edges

- **mod** – optional, a large random prime

Returns the maximum cardinality matching of the graph

2.5.16 pyrival.graphs.prim

pyrival.graphs.prim.**prim**(*n*, *adj*)

2.5.17 pyrival.graphs.scc

Given a directed graph, find_SCC returns a list of lists containing the strongly connected components in topological order.

Note that this implementation can be also be used to check if a directed graph is a DAG, and in that case it can be used to find the topological ordering of the nodes.

pyrival.graphs.scc.**find_SCC**(*graph*)

2.5.18 pyrival.graphs.toposort

pyrival.graphs.toposort.**kahn**(*graph*)

pyrival.graphs.toposort.**toposort**(*graph*)

2.6 pyrival.linear_algebra

2.6.1 pyrival.linear_algebra.matrix

pyrival.linear_algebra.matrix.**eye**(*m*)
returns an indentity matrix of order m

pyrival.linear_algebra.matrix.**mat_add**(**mat*)

pyrival.linear_algebra.matrix.**mat_inv**(*A*)

pyrival.linear_algebra.matrix.**mat_mul**(*A*, *B*)

pyrival.linear_algebra.matrix.**mat_pow**(*mat*, *power*)
returns mat**power

pyrival.linear_algebra.matrix.**mat_sub**(*A*, *B*)

pyrival.linear_algebra.matrix.**minor**(*mat*, *i*, *j*)

pyrival.linear_algebra.matrix.**transpose**(*mat*)

pyrival.linear_algebra.matrix.**vec_mul**(*mat*, *vec*)

2.6.2 pyrival.linear_algebra.max_xor

Maximizes xor of values in a list (works with big integers)

Example: >>>> A = [10**20, 3, 6, 4] >>>> I = max_xor(A) >>>> xor = 0 >>>> for i in I: .... xor ^= A[i] .... >>>> xor 10000000000000000007

pyrival.linear_algebra.max_xor.**max_xor**(*A*)
Input: List A of non-negative integers Output: I such that xor(A[i] for i in I) is maximized

2.6.3 pyrival.linear_algebra.multivariable_crt

pyrival.linear_algebra.multivariable_crt.**extended_gcd**(*a*, *b*)
returns gcd(a, b), s, r s.t. a * s + b * r == gcd(a, b)

pyrival.linear_algebra.multivariable_crt.**gcd**(*x*, *y*)
greatest common divisor of x and y

pyrival.linear_algebra.multivariable_crt.**is_sol**(*A*, *x*, *b*, *m*)
checks if Ax = b mod m

pyrival.linear_algebra.multivariable_crt.**mat_mul**(*A*, *B*)

pyrival.linear_algebra.multivariable_crt.**mat_sub**(*A*, *B*)

pyrival.linear_algebra.multivariable_crt.**mcrt**(*A*, *b*, *m*)
returns x s.t. Ax = b mod m

pyrival.linear_algebra.multivariable_crt.**modinv**(*a*, *m*)
returns the modular inverse of a w.r.t. to m

pyrival.linear_algebra.multivariable_crt.**pivot**(*A*, *m*)
returns the pivot of A and m

2.7 pyrival.misc

2.7.1 pyrival.misc.FastIO

class pyrival.misc.FastIO.**FastIO**(*file*)
Bases: io.IOBase

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

newlines = 0

read()

readline()
Read and return a line from the stream.

If size is specified, at most size bytes will be read.

The line terminator is always b'n' for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

class pyrival.misc.FastIO.**IOWrapper**(*file*)
Bases: io.IOBase

pyrival.misc.FastIO.**input**()

pyrival.misc.FastIO.**str**(*x=b"*)

2.7.2 pyrival.misc.Random

2.7.3 pyrival.misc.alphabeta

class pyrival.misc.alphabeta.**AlphaBetaNode**(*value=None*, *children=None*)
Bases: object

pyrival.misc.alphabeta.**alphabeta**(*node*, *depth*, *alpha=-inf*, *beta=inf*, *maximizingPlayer=True*)

2.7.4 pyrival.misc.as_integer_ratio

pyrival.misc.as_integer_ratio.**as_integer_ratio**(*x*, *prec=53*)

2.7.5 pyrival.misc.bit_hacks

pyrival.misc.bit_hacks.**least_bit**(*x*)

pyrival.misc.bit_hacks.**next_mask**(*x*)

pyrival.misc.bit_hacks.**subset_masks**(*m*)

pyrival.misc.bit_hacks.**sum_of_subsets**(*K*, *D*)

2.7.6 pyrival.misc.bootstrap

pyrival.misc.bootstrap.**bootstrap**(*f*, *stack=[]*)

2.7.7 pyrival.misc.cumsum2d

pyrival.misc.cumsum2d.**cumsum2d**(*A*)

2.7.8 pyrival.misc.lis

pyrival.misc.lis.**lis**(*nums*, *cmp=<function <lambda>>*)

2.7.9 pyrival.misc.memoize

pyrival.misc.memoize.**memodict**(*f*)
Memoization decorator for a function taking a single argument.

pyrival.misc.memoize.**memoize**(*f*)
Memoization decorator for a function taking one or more arguments.

2.7.10 pyrival.misc.mod

2.7.11 pyrival.misc.order_statistic

pyrival.misc.order_statistic.**order_statistic**(*a*, *k*)
returns the k-th (0 <= k < len(a)) largest element of a

2.7.12 pyrival.misc.ordersort

pyrival.misc.ordersort.**bucketsort**(*order*, *seq*)

pyrival.misc.ordersort.**long_ordersort**(*order*, *seq*)

pyrival.misc.ordersort.**multikey_ordersort**(*order*, **seqs*, *sort=<function ordersort>*)

pyrival.misc.ordersort.**ordersort**(*order*, *seq*, *reverse=False*)

2.7.13 pyrival.misc.ostream

class pyrival.misc.ostream.**ostream**
Bases: object

2.7.14 pyrival.misc.py3k

Python 3 compatibility tools.

2.7.15 pyrival.misc.readnumbers

pyrival.misc.readnumbers.**readnumbers**(*zero=0*)

2.7.16 pyrival.misc.split

pyrival.misc.split.**split**(*b*)

2.8 pyrival.numerical

2.8.1 pyrival.numerical.berlekamp_massey

pyrival.numerical.berlekamp_massey.**berlekamp_massey**(*s*)

pyrival.numerical.berlekamp_massey.**linear_rec**(*S*, *tr*, *k*)

2.8.2 pyrival.numerical.hill_climbing

pyrival.numerical.hill_climbing.**hill_climbing**(*func*, *x_0*, *y_0*, *cmp=<built-in function min>*)

2.8.3 pyrival.numerical.integrate

pyrival.numerical.integrate.**fast_quad**(*func*, *a*, *b*, *eps=1e-06*)

pyrival.numerical.integrate.**quad**(*func*, *a*, *b*, *n=1000*)

pyrival.numerical.integrate.**rec**(*func*, *a*, *b*, *eps*, *S*)

pyrival.numerical.integrate.**simpson**(*func*, *a*, *b*)

2.8.4 pyrival.numerical.interpolate

pyrival.numerical.interpolate.**interpolate**(*points*)

2.8.5 pyrival.numerical.iroot

pyrival.numerical.iroot.**iroot**(*n*, *k=2*)

2.8.6 pyrival.numerical.polynomial

pyrival.numerical.polynomial.**diff**(*a*)

pyrival.numerical.polynomial.**divroot**(*a*, *x0*)

pyrival.numerical.polynomial.**poly**(*a*, *x*)

2.8.7 pyrival.numerical.search

pyrival.numerical.search.**binary_search**(*func*, *lo*, *hi*, *abs_prec=1e-07*)
Locate the first value x s.t. func(x) = True within [lo, hi]

pyrival.numerical.search.**discrete_binary_search**(*func*, *lo*, *hi*)
Locate the first value x s.t. func(x) = True within [lo, hi]

pyrival.numerical.search.**discrete_ternary_search**(*func*, *lo*, *hi*)
Find the first maximum of unimodal function func() within [lo, hi]

pyrival.numerical.search.**fractional_binary_search**(*func*, *lo=(0, 1)*, *hi=(1, 0)*, *limit=1000000*)

pyrival.numerical.search.**golden_section_search**(*a*, *b*, *func*, *abs_prec=1e-07*)

pyrival.numerical.search.**ternary_search**(*func*, *lo*, *hi*, *abs_prec=1e-07*)
Find maximum of unimodal function func() within [lo, hi]

2.9 pyrival.strings

2.9.1 pyrival.strings.LCSubstr

pyrival.strings.LCSubstr.**LCSubstr**(*a*, *b*)

2.9.2 pyrival.strings.LPSubstr

pyrival.strings.LPSubstr.**LPSubstr**(*s*)

2.9.3 pyrival.strings.hashing

class pyrival.strings.hashing.**Hashing**(*s*, *mod=2147483647*, *base1=1092183611*, *base2=1711395481*)
Bases: object

get_hashes(*length*)

hashed(*start*, *stop*)

2.9.4 pyrival.strings.kmp

pyrival.strings.kmp.**match**(*s*, *pat*)

pyrival.strings.kmp.**partial**(*s*)

pyrival.strings.kmp.**string_find**(*s*, *pat*)

2.9.5 pyrival.strings.lcs

pyrival.strings.lcs.**lcs**(*a*, *b*)

pyrival.strings.lcs.**lps**(*s*)

2.9.6 pyrival.strings.min_rotation

pyrival.strings.min_rotation.**least_rotation**(*s*)

2.9.7 pyrival.strings.suffix_array

Calculates the suffix array and LCP array in O(n) time

Example: >>>> S = 'cabbage' >>>> SA = SAIS([ord(c) for c in S]) >>>> LCP = KASAI(S, SA) >>>> SA [1, 4, 3, 2, 0, 6, 5] >>>> LCP [1, 0, 1, 0, 0, 0]

pyrival.strings.suffix_array.**KASAI**(*A*, *SA*)
Calculates LCP array in O(n) time Input: String A and its suffix array SA

pyrival.strings.suffix_array.**SAIS**(*A*)
Calculates suffix array in O(len(A) + max(A)) Input: Int list A with A[i] >= 0 for all i

2.10 pyrival.tools

2.10.1 pyrival.tools.interactive_runner

class pyrival.tools.interactive_runner.**PrefixedStream**(*stream*, *prefix*)
Bases: io.IOBase

close()
Flush and close the IO object.

This method has no effect if the file is already closed.

write(*b*)

pyrival.tools.interactive_runner.**async_main**(*argv=None*)

pyrival.tools.interactive_runner.**main**(*argv=None*)

pyrival.tools.interactive_runner.**show_exit_code**(*process*, *prefix*)

pyrival.tools.interactive_runner.**tee**(*in_stream*, *out_streams*)

2.10.2 pyrival.tools.stress_tester

pyrival.tools.stress_tester.**cmd2func**(*args*)

pyrival.tools.stress_tester.**func2judge**(*sol*)

pyrival.tools.stress_tester.**stress_tester**(*tests*, *solution*, *judge=None*, *catch_all=False*)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

D

E

F

G

N

O

P

Q

R

S

T

U

V

W

X