Pudełka 3D

Wydanie 1.1

Dr. Marcin Kuropatwiński i Robert Kolke

29 paź 2020

Contents

CHAPTER 1

Podstawowe informacje

1.1 Licencja

Program jest udostępniany na licencji GNU General Public License https://www.gnu.org/licenses/licenses.pl.html

1.2 Założenia działania programu

Program bierze dowolną liczbę pudełek które mogą się przecinać i zwraca pudełka nie przecinające się mające jako unię unię pudełek na wejściu

Przecinanie - pudełka przecinają się jeżeli tworzące je interwały mają niepustą cześć wspólną dla osi x, y i z.

Rozbicie - jeśli dwa pudełka się przecinają, zostaje zwrócona niekoniecznie taka sama ilość pudełek, które unię mają równą pudełkom przecinającym się, lecz się nie przecinają.

1.3 Układ współrzędnych kartezjańskich

Jest to układ współrzędnych oparty na trzech płaszczyznach: x, y oraz z. Więcej informacji https://en.wikipedia.org/wiki/Cartesian_coordinate_system

1.4 interwały

Interwał to odcinek o dwóch atrybutach granicznych.

1.5 pudełka

Pudełka to obiekty przedstawione przez 3 interwały, po jednym na każdą z płaszczyzn. Można je sobie wyobrazić jako zwykłe prostopadłościany.

1.6 drzewo

Drzewo korzystające z 3-wymiarowych węzłów.

CHAPTER 2

Szczegóły techniczne

2.1 Złożoność obliczeniowa

Złożoność obliczeniowa algorytmu wynosi: O(nlog^3*(n)), gdzie n to liczba pudełek wyjściowych.

2.2 Wersja języka programowania

Python wersja 3.8.5 - https://www.python.org

2.3 Biblioteki

- rtree - https://pypi.org/project/Rtree/
- portion - https://pypi.org/project/portion/
- math - (wbudowana w język programowania)

2.4 Środowisko programistyczne

Pycharm wersja community - https://www.jetbrains.com/pycharm/

CHAPTER 3

Lista modułów

3.1 boxes3D

class boxes3D.**boxStack**

Klasy bazowe: `object`

Klasa przechowująca dane

stosu pudełek z wejścia programu

append(*added*)

Funkcja append stosu

Parametry **added** – element do dodania do stosu

extend(*added*)

Funkcje extend stosu

Parametry **added** – element, który posłuży do rozszerzenia stosu

get_stack()

Funkcja get stosu

Zwraca Obecny stos

Typ zwracany *boxStack*

pop()

Funkcje pop stosu

Zwraca stos pomniejszony o ostatni element

Typ zwracany *boxStack*

set_stack(*new_stack*)

Funkcja set stosu

Parametry **new_stack** – aktualizowanie stanu stosu

class boxes3D.**tree**
Klasy bazowe: object

Klasa przechowująca instrukcje drzewa

get_tree()
Funkcje get drzewa

Zwraca drzewo rtree na którym algorytm główny zamieszcza pudełka

Typ zwracany rtree

set_tree(*new_tree*)
Funkcje set drzewa

Parametry **new_tree** – nowe drzewo

3.2 cut_box

class cut_box.**box3D**(*interval_x*, *interval_y*, *interval_z*)
Klasy bazowe: object

Klasa przechowująca instrukcje dot. pudełek

static factory(*x1*, *y1*, *z1*, *x2*, *y2*, *z2*)
metoda statyczna tworząca pudełko na podstawie interwałów

podanych w kolejności wszystkie lower, potem wszystkie upper

Parametry

- **x1** – wartość lower dla interwału na osi x
- **y1** – wartość lower dla interwału na osi y
- **z1** – wartość lower dla interwału na osi z
- **x2** – wartość upper dla interwału na osi x
- **y2** – wartość upper dla interwału na osi y
- **z2** – wartość upper dla interwału na osi z

Zwraca nowy obiekt pudełko

Typ zwracany *box3D*

get_interval_x()
Funkcje get dla jednego z interwałów

Zwraca interwał pudełka wskazujący położenie na osi x

Typ zwracany portion.Interval

get_interval_y()
Funkcje get dla jednego z interwałów

Zwraca interwał pudełka wskazujący położenie na osi y

Typ zwracany portion.Interval

get_interval_z()
Funkcje get dla jednego z interwałów

Zwraca interwał pudełka wskazujący położenie na osi z

Typ zwracany portion.Interval

class cut_box.**myInterval**(**intervals*)

Klasy bazowe: `portion.interval.Interval`

box_cut(*box1*)

Funkcja przycinająca pudełko

Parametry **box1** – pudełko przed przycięciem

Zwraca pudełko po przycięciu

Typ zwracany *box3D*

box_cut_execute(*interval*)

Funkcja która przycina 1 interwał

Parametry **interval** – interwał do przycięcia

Zwraca interwał przycięty o epsilon z obu wartości granicznych

Typ zwracany *myInterval*

box_uncut(*box1*)

Funkcja cofająca przycięcie pudełka

Parametry **box1** – pudełko, w którym chcemy cofnąć przycięcie interwałów o epsilon

Zwraca pudełko z cofniętym przycięciem interwałów

Typ zwracany *box3D*

box_uncut_execute(*interval*)

Funkcja która cofa przycięcie 1 interwału

Parametry **interval** – interwał przycięty o epsilon, produkt funkcji box_cut_execute

Zwraca interwał w stanie takim samym jak przed przycięciem

Typ zwracany *myInterval*

eps = 1e-07

Klasa dziedzicząca z funkcji interwał

Parametry **eps** – liczba potrzebna do przycięcia pudełek,

bez tego są liczone jako przecinające się nawet jak tylko nachodzą na siebie granicami

property get_lower_eps

Funkcja get dla wartości lower

Zwraca wartość lower_eps(ilon)

Typ zwracany complex

property get_lower_meps

Funkcja get dla wartości lower

Zwraca wartość lower_meps(ilon)

Typ zwracany complex

property get_upper_eps

Funkcja get dla wartości upper

Zwraca wartość upper_eps(ilon)

Typ zwracany complex

property get_upper_meps
Funkcja set dla wartości upper

Zwraca wartość upper_meps(ilon)

Typ zwracany complex

property lower_eps
wartość lower interwału + eps(ilon)

Typ zwracany complex

Type return

property lower_meps
wartość lower interwału - eps(ilon)

Typ zwracany complex

Type return

property upper_eps
wartość upper interwału + eps(ilon)

Typ zwracany complex

Type return

property upper_meps
wartość upper interwału - eps(ilon)

Typ zwracany complex

Type return

cut_box.**my_closed**(*lower*, *upper*)
Funkcja tworząca interwał zastępująca oryginalny closed

Parametry

- **lower** – wartość lower dla nowo powstającego interwału
- **upper** – wartość upper dla nowo powstającego interwału

Zwraca nowy interwał o dodanych właściwościach

Typ zwracany *myInterval*

3.3 mainalgo

class mainalgo.**algorithm**
Klasy bazowe: object

Główna klasa programu.

static algorytm(*Q*, *tree*)
Funkcja statyczna, w wyniku której wszystkie przecinające się pudełka ze stosu zostają rozbite i wstawione do drzewa :param Q: stos pudełek

Parametry **tree** – puste drzewo z indeksowaniem w trzech wymiarach

Zwraca drzewo rtree zawierające pudełka

Typ zwracany tree.tree

begin(*box1*, *box2*)
Funkcja obraca pudełka zmieniając kolejność interwałów

Parametry

- **box1** – pudełko ze stosu
- **box2** – pudełko z drzewa

Zwraca lista table, która zawiera pudełka po rozbiciach

Typ zwracany list

3.4 signatures_setup

class signatures_setup.**signatures**
Klasy bazowe: object

get_signature(*interval1*, *interval2*)
Funkcja nadająca sygnatury parom interwałów

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca sygnaturę zależnie od stosunku porównywanych interwałów

Typ zwracany string

get_signatures_triple(*box1*, *box2*)
Funkcja zwracająca listę z trzema sygnaturami interwałów

Parametry

- **box1** – pierwsze z pudełek, na podstawie których program dobiera zestaw sygnatur
- **box2** – drugie z pudełek, na podstawie których program dobiera zestaw sygnatur

Zwraca listę sygnatur stosunku położenia interwałów dla pudełek wprowadzonych

Typ zwracany list

ie(*interval1*, *interval2*)
Funkcja sprawdzająca, jaką sygnaturę nadać danej parze interwałów

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku equal, inaczej False

Typ zwracany bool

ii12(*interval1*, *interval2*)
Funkcja sprawdzająca, jaką sygnaturę nadać danej parze interwałów

Parametry

- **interval1** – interwał pierwszy do porównania
- **interval2** – interwał drugi do porównania

Zwraca True jeśli interwały są w stosunku in oraz drugi interwał jest ma przedział zaczynający się wyżej

Typ zwracany bool

ii21(*interval1*, *interval2*)
Funkcja sprawdzająca, jaką sygnaturę nadać danej parze interwałów

Parametry

- **interval1** – interwał pierwszy do porównania
- **interval2** – interwał drugi do porównania

Zwraca True jeśli interwały są w stosunku in oraz pierwszy interwał jest ma przedział zaczynający się wyżej

Typ zwracany bool

io12(*interval1*, *interval2*)
Funkcja sprawdzająca, jaką sygnaturę nadać danej parze interwałów

Parametry

- **interval1** – interwał pierwszy do porównania
- **interval2** – interwał drugi do porównania

Zwraca True jeśli interwały są w stosunku out oraz drugi interwał jest ma przedział zaczynający się wyżej

Typ zwracany bool

io21(*interval1*, *interval2*)
Funkcja sprawdzająca, jaką sygnaturę nadać danej parze interwałów

Parametry

- **interval1** – interwał pierwszy do porównania
- **interval2** – interwał drugi do porównania

Zwraca True jeśli interwały są w stosunku out oraz pierwszy interwał jest ma przedział zaczynający się wyżej

Typ zwracany bool

is_equal(*interval1*, *interval2*)
Funkcja sprawdzająca czy interwały są w stosunku equal

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku equal, inaczej False

Typ zwracany bool

is_half_out(*int1*, *int2*)
Funkcja sprawdzająca czy interwały są w stosunku half out

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku half out, inaczej False

Typ zwracany bool

is_in(*interval1*, *interval2*)
Funkcja sprawdzająca czy interwały są w stosunku in

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku in, inaczej False

Typ zwracany bool

is_out(*interval1*, *interval2*)
Funkcja sprawdzająca czy interwały są w stosunku out

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku out, inaczej False

Typ zwracany bool

is_separate(*int1*, *int2*)
Funkcja sprawdzająca czy interwały są w stosunku separate

Parametry

- **interval1** – pierwszy z interwałów do porównania
- **interval2** – drugi z interwałów do porównania

Zwraca True jeśli pudełka są w stosunku separate, inaczej False

Typ zwracany bool

my_sort(*inputlist*)
Funkcja sortująca sygnatury interwałów

Parametry **inputlist** – lista

Zwraca

permute(*sortin*, *permutation*)
Funkcja przeprowadzająca operację permutacji

Parametry

- **sortin** – lista zmiennych do permutacji
- **permutation** – kolejność permutacji

Zwraca lista po permutacji

Typ zwracany list

permute_signatures(*box_tab*, *sort_order*)
Funkcja permutująca interwały z listy

Parametry

- **box_tab** – lista interwałów z pudełka

- **sort_order** – kolejność sortowania w postaci listy np. [3, 1, 2, 0]

Zwraca lista interwałów po permutacji

Typ zwracany list

ret_original_order(*split*, *sorted2in*)
Funkcja przywracająca oryginalny porządek interwałów

Parametry

- **split** – lista posortowanych interwałów
- **sorted2in** – kolejność docelowej permutacji

Zwraca listę pudełek z posortowanymi interwałami

Typ zwracany list

sort_signatures(*box1*, *box2*, *in2sorted*)
Funkcja przygotowująca pudełka do posortowania interwałów

Parametry

- **box1** – pierwsze pudełko z nieposortowanymi interwałami
- **box2** – drugie pudełko z nieposortowanymi interwałami
- **in2sorted** – lista z kolejnością do permutowania interwałów

Zwraca 2 listy interwałów po permutacji według zadanej kolejności

Typ zwracany list, list

3.5 split_intervals

split_intervals.**mylen**(*interval*)
Funkcja licząca długość interwału potrzebna do funkcji is_half_out

Parametry **interval** – interwał do zmierzenia długości

Zwraca długość interwału

Typ zwracany complex

class split_intervals.**split**
Klasy bazowe: object

Klasa dzieląca pudełka

zawiera ona 20 funkcji, z których każda rozbija pudełka

Uwzględniłem możliwość wystąpienia half-out

Parametry **empty** – interwał pusty

empty = ()

iII_I_iII_I_iII_I(*box1*, *box2*)

iII_I_iII_I_oII_I(*box1*, *box2*)

iII_I_iII_I_oI_II(*box1*, *box2*)

iII_I_oII_I_oII_I(*box1*, *box2*)

iII_I_oI_II_oII_I(*box1*, *box2*)

iII_I_oI_II_oI_II(*box1*, *box2*)

iI_II_iII_I_iII_I(*box1*, *box2*)

iI_II_iII_I_oII_I(*box1*, *box2*)

iI_II_iII_I_oI_II(*box1*, *box2*)

iI_II_iI_II_iII_I(*box1*, *box2*)

iI_II_iI_II_iI_II(*box1*, *box2*)

iI_II_iI_II_oII_I(*box1*, *box2*)

iI_II_iI_II_oI_II(*box1*, *box2*)

iI_II_oII_I_oII_I(*box1*, *box2*)

iI_II_oI_II_oII_I(*box1*, *box2*)

iI_II_oI_II_oI_II(*box1*, *box2*)

oII_I_oII_I_oII_I(*box1*, *box2*)

oI_II_oII_I_oII_I(*box1*, *box2*)

oI_II_oI_II_oII_I(*box1*, *box2*)

oI_II_oI_II_oI_II(*box1*, *box2*)

split(*idx_sign*, *tri_sign*, *tri_sign_i*)

Funkcja dobierająca właściwą funkcje rozbijającą na bazie trójki interwałów

Parametry

- **idx_sign** – lista sygnatur interwałów
- **tri_sign** – lista interwałów pierwszego pudełka
- **tri_sign_i** – lista interwałów drugiego pudełka

Zwraca listę pudełek powstałych w wyniku rozbicia pudełek wejściowych

- genindex
- modindex
- search

Dokumentacja pdf: „https://github.com/KolRobOsk/Praktyki/blob/Final/docs/pudeka3d.pdf”

Indeks modułów Pythona

b

c

Indeks

L

M

O

P

R

S

T

U