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 Praktyki.boxes3D

3.1.1 rola w programie

Moduł posiada 2 klasy: * Tree - drzewo przechowujące pudełka po zakończeniu działania programu. * BoxStack - stos pudełek przechowujący pudełka nie wprowadzone jeszcze do drzewa.

3.1.2 funkcje

nazwa(typy argumentów:argumenty)

- Tree.get_tree() - funkcja zwracająca drzewo dla obiektu rtree
- Tree.set_tree(rtree:new_tree) - funkcja ustawiająca nową postać drzewa
- BoxStack.get_stack() - zwraca stos dla obiektu BoxStack
- BoxStack.set_stack(BoxStack:new_stack) - ustawia nową postać stosu
- BoxStack.append(Box3D:added) - dodaje element do stosu pudełek (wstawiając cały obiekt added na koniec)
- BoxStack.extend(list:added) - rozszerza stos pudełek (wstawiając wszystkie elementy Box3D z listy na koniec stosu)
- BoxStack.pop() - usuwa ostatni element stosu pudełek

3.2 Praktyki.cut_box

3.2.1 rola w programie

Moduł składa się z 2 klas: Box3D oraz myInterval oraz funkcji my_closed. Przy pomocy klasy Box3D tworzone są pudełka. Klasa myInterval dziedziczy z klasy Portion.Interval. Jej zadaniem jest przycinanie (odejmowanie bardzo małych liczb, czyli epsilonów od granicy interwałów) pudełek, aby program nie rozpoznawał pudełek jako przecinające się gdy tylko stykają się ze sobą. my_closed to funkcja tworząca interwał, który potem będzie można przyciąć.

3.2.2 funkcje

nazwa(typy argumentów:argumenty)

- my_closed(lower, upper) - funkcja tworząca interwał z atrybutami lower - dolna granica, upper - górna granica
- box3D.get_interval_x() - funkcja zwracająca interwał pudełka na płaszczyźnie x
- box3D.get_interval_y() - funkcja zwracająca interwał pudełka na płaszczyźnie y
- box3D.get_interval_z() - funkcja zwracająca interwał pudełka na płaszczyźnie z
- box3D.__contains__(list:num) - funkcja, która sprawdza czy punkt jest wewnątrz pudełka
- box3D.__ror__(list:num) - funkcja, która sprawdza czy punkt jest na granicy pudełek
- box3D.factory(complex:x1, complex:y1, complex:z1, complex:x2, complex:y2, complex:z2) - tworzy pudełko na bazie podanych wartości granicznych poszczególnych interwałów (x1 to dolna granica interwału x, zaś y2 to górna granica interwału y)
- myInterval.upper_eps() - atrybut o wartości górnej interwału przypisanego do obiektu + eps (epsilon, bardzo mała liczba)
- myInterval.upper_meps() - atrybut o wartości górnej interwału przypisanego do obiektu - eps (epsilon, bardzo mała liczba)
- myInterval.lower_eps() - atrybut o wartości dolnej interwału przypisanego do obiektu + eps (epsilon, bardzo mała liczba)
- myInterval.lower_meps() - atrybut o wartości dolnej interwału przypisanego do obiektu - eps (epsilon, bardzo mała liczba)
- myInterval.get_upper_eps() - funkcja zwracająca atrybut upper_eps
- myInterval.get_upper_meps() -funkcja zwracająca atrybut upper_meps
- myInterval.get_lower_eps() - funkcja zwracająca atrybut lower_eps
- myInterval.get_lower_meps() - funkcja zwracająca atrybut lower_eps
- myInterval.upper_eps() - funkcja ustawiająca atrybut upper_eps na podstawie atrybutu upper + eps
- myInterval.upper_meps() - funkcja ustawiająca atrybut upper_eps na podstawie atrybutu upper + eps
- myInterval.lower_eps() - funkcja ustawiająca atrybut upper_eps na podstawie atrybutu lower + meps
- myInterval.lower_meps() -funkcja ustawiająca atrybut upper_eps na podstawie atrybutu lower + meps
- myInterval.box_cut(box3D: box1) - funkcja przycina pudełko tj. uruchamia dla każdego interwału (x, y, z) funkcję box_cut_execute
- myInterval.box_uncut(box3D: box2) - funkcja cofa działanie funkcji box_cut dla każdego interwału (x, y, z) przy pomocy funkcji box_uncut_execute

- myInterval.box_cut_execute(myInterval:interval) - funkcja zwraca dla podanego interwału interwał o wartościach granicznych lower + eps, upper - eps
- myInterval.box_uncut_execute(myInterval:interval) - funkcja zwraca dla podanego interwału interwał o wartościach granicznych lower - eps, upper + eps

3.3 Praktyki.signatures_setup

3.3.1 rola w programie

Moduł posiada klasę signatures, dzięki której możliwe jest przeprowadzanie operacji, takich jak permutacja czy sortowanie na sygnaturach.

3.3.2 funkcje

nazwa(typy argumentów:argumenty)

- signatures.io12(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli interwały są w stosunku out oraz drugi interwał ma wyższą wartość górną lub są w stosunku half out i drugi interwał ma większy przedział niż unia interwałów w innym przypadku zwraca False
- signatures.io21(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli interwały są w stosunku out oraz pierwszy interwał ma wyższą wartość górną lub są w stosunku half out i pierwszy interwał ma większy przedział niż unia interwałów, w innym przypadku zwraca False
- signatures.ii12(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli interwały są w stosunki in oraz drugi interwał ma większy przedział od pierwszego, w innym przypadku zwraca False
- signatures.ii21(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli interwały są w stosunki in oraz pierwszy interwał ma większy przedział od drugiego, w innym przypadku zwraca False
- signatures.get_signatures_triple(box3D:box1, box3D:box2) - funkcja zwraca listę sygnatur dla dwóch pudełek na wejściu, korzystając z funkcji get_signature dla każdej pary interwałów
- signatures.sort_signatures(box3D: box1, box3D: box2, list:in2sorted) - funkcja zwraca spermutowane według wzoru liste interwałów dwóch pudełek wejściowych
- signatures.permute_signatures(box_tab, sort_order) - funkcja permutuje interwały z listy
- signatures.permute(list:sortin, list:permutation) - funkcja permutuje listę (sortin) podaną na wejściu według ustalonego wzoru
- signatures.my_sort(list:inputlist) - funkcja zwraca 3 listy: listę posortowanych elementów, wzór permutacji aby dojść do kolejneści posortowanych elementów, oraz wzór permutacji aby cofnąć sortowanie
- signatures.is_in(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli między interwałami występuje stosunek in, w innym przypadku zwraca False
- signatures.is_out(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli między interwałami występuje stosunek out, w innym przypadku zwraca False
- signatures.is_equal(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli między interwałami występuje stosunek equal, w innym przypadku zwraca False
- signatures.is_separate(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli między interwałami występuje stosunek separate, w innym przypadku zwraca False
- signatures.is_half_out(myInterval:interval1, myInterval:interval2) - funkcja zwraca True jeśli między interwałami występuje stosunek half out, w innym przypadku zwraca False

- signatures.ie(myInterval:interval1, myInterval:interval2) - funkcja zwraca True gdy interwały mają równy przedział, w innym przypadku zwraca False
- signatures.get_signature(myInterval:interval1, myInterval:interval2) - funkcja zwraca konkretną sygnaturę dla danej pary interwałów: io12, io21, ii12, ii21, w przypadku stosunku ie zwraca ii12
- signatures.ret_original_order(list:split, list:sorted2in) - funkcja zwraca listę pudełek spermutowanych według wzoru permutacji sorted2in

3.4 Praktyki.split_intervals

3.4.1 rola w programie

Split intervals ma 1 klasę - split, która służy to rozbijania pudełek, poprzez tworzenie nowych o wymaganych cechach oraz funkcję mylen. W klasie znajduje się 20 funkcji rozbijających, o takiej samej anatomii.

3.4.2 funkcje

nazwa(typy argumentów:argumenty)

- my_len(myInterval:interval) - funkcja, która zwraca długość interwału
- [funkcje rozbijające, nazwa:stosunek1_stosunek2_stosunek3, argumenty: box3D:box1, box3D:box2] - funkcja na początku re-deklaruje wszystkie interwały z obu pudełek, następnie tworzy pudełka i wstawia na zwracaną listę. UWAGA - niektóre pudełka zostają pominięte, gdyż interwały w stosunku half out czasem powodują, że interwał jedno lub więcej pudełek wyjściowych nie powinno istnieć według współrzędnych i program przestaje działać.
- split.split(list:idx_sign, list:tri_sign, list:tri_sign_i) - w tej funkcji znajduje się słownik, w którym mieszczą się odniesienia do wszystkich funkcji rozbijających, a na podstawie sygnatur funkcja wybiera, z której funkcji rozbijającej skorzystać.

3.5 Praktyki.mainalgo

3.5.1 rola w programie

Moduł służy do uruchomienia całego algorytmu. Usuwa pliki pozostawione po ostatnim drzewie. Posiada jedną klasę, algorithm.

3.5.2 funkcje

nazwa(typy argumentów:argumenty)

- algorithm.algorytm(BoxStack:Q, rtree:tree) - statyczna funkcja, inicjuje główną pętlę programu, sprawdza czy pudełko rozpatrywane przecina się z którymkolwiek zawartym w drzewie. Jeśli tak, usuwa jedno z przecinających się z drzewa i rozbija je z pudełkiem ze stosu i wstawia na stos. Jeśli nie, wstawia na pudełko na drzewo i usuwa ze stosu.
- algorithm.begin(Box3D:box1, Box3D:box2) - zmienia tworzy tablicę sygnatur dla dwóch pudełek, sortuje je, według ich kolejności permutuje interwały w pudełkach. Następnie pudełka zostają rozbite, a ich interwały permutowane do pierwotnej kolejności.

- genindex
- modindex
- search

Dokumentacja pdf: «https://github.com/KolRobOsk/Praktyki/blob/Praktyki/pudeka3d.pdf»