Przewodnik po bibliotece calkowanie

służącej do liczenia całek dla wielomianów

Maksymilian Siembab

Wstęp

Celem tego dokumentu jest wytłumaczenie jakie funkcje znajdują się w bibliotece oraz jak działają. Biblioteka ta powstała aby móc przekształcać wielomiany zapisane za pomocą łańcuchów znakowych na grafy w postaci list zagnieżdżonych przyjmujących formę s-wyrażeń. W następnych wersjach wzbogaciła się o funkcje liczące całek z przekształconych wielomianów.

Opis zawartości

Funkcje:

- **parse:** przyjmuje jako argument łańcuch znakowy reprezentujący wielomian, zwraca listę składającą się z podzielonych na poszczególne składowe części łańcucha. Dzielenie następuje poprzez wyszczególnienie operatorów takich jak +, -, *, /, ^, (,). Taki podział gwarantuje, że stałe składające się z więcej niż jednej cyfry zostaną rozdzielone poprawnie.

-**preoperation:** przyjmuje jako argument listę składowych wielomianu, zwraca tę samą listę ze wstępnie zgrupowanymi nawiasami. Funkcja ta grupuje wszystkie nawiasy (również te zagnieżdżone) w formie s-wyrażenia [‘(‘, a] gdzie ‘(‘ jest operacją a a tym co znajduje się w nawiasie. ‘(‘ jest więc operatorem jednoargumentowym.

-**operation:** przyjmuje jako argument (najlepiej przetworzoną już przez funkcję preoperation w celu wyodrębnienia nawiasów) listę składowych wielomianu oraz operację względem jakiej będzie tworzyć s-wyrażenia, zwraca graf wypełniony s-wyrażeniami operacji przekazanej jako drugi argument. S-wyrażenia tworzone przez tą funkcję mają postać: [operacja, pr, po] gdzie pr to to co znajduje się przed taką operacją na liście a po oznacza to co znajduje się po takiej operacji.

-**graph:** przyjmuje jako argument wstępnie utworzony graf (w tej bibliotece pewna funkcja przekazuje do funkcji graph wstępny graf będący wynikiem funkcji operation) oraz operacja względem której będą tworzone s-wyrażenia. Funkcja ta przeszukuje graf w poszukiwaniu nieprzetworzonych składowych (muszą tu być poszczególne składowe, nie czyste łańcuchy znakowe) a kiedy takie znajdzie wykonuje na nich funkcję operations z podaniem przekazanego jako drugi argument operatora.

-**delistifier:** przyjmuje jako argument gotowy graf, przegląda go w poszukiwaniu końcowych gałęzi , a następnie „wyjmuje” argumenty z jednoargumentowych list na ich końcach, zwraca tak przetworzony graf. W wyniku przekształceń przez jakie przeszedł graf na jego końcach powstały listy zawierające końcowe argumenty. W celu zapewnienia lepszego działania funkcji dalej przekształcających graf należy je z nich wyciągnąć.

-**make_a_graph:** przyjmuje jako argument listę składowych wielomianu, zwraca prawie gotowy graf. Graf ten jest wykonany zgodnie z kolejnością wykonywania działań. Graf jest prawie gotowy ponieważ nie zostały jeszcze wykonane grafy wewnątrz nawiasów.

-**deep_graph:** przyjmuje jako argument prawie gotowy graf (najlepiej wykonany przy użyciu metody make_a_graph) zwracając graf z gotowymi grafami w nawiasach. Funkcja ta przeszukuje graf w poszukiwaniu nawiasów, na których zawartości wykonuje funkcję make_a_graph i przeszukuje dalej ten utworzony graf w poszukiwaniu nawiasów aż nie znajdzie wszystkich.

-assemble: przyjmuje jako argument gotowy graf, zwraca ciąg znaków reprezentujący ten graf. Funkcja ta zwraca łańcuch znakowy będący poskładanymi wielomianem wcześniej zapisanym w postaci grafu.

-prepareFunction: przyjmuje jako argument ciąg znaków reprezentujący znak, zwraca gotowy graf. Funkcja ta wykonuje wszystkie potrzebne kroki aby zamienić ciąg znaków w graf, wywołując wcześniej opisane funkcje.

-found: przyjmuje jako argument graf oraz szukaną zmienną, zwraca wartość True jeśli w grafie znajduje się podana zmienna.

-extension: przyjmuje graf oraz zmienną, po której całkuje, zwraca zcałkowany graf. Całkuje graf zgodnie z zasadami całkowania dla prostych wielomianów.

-integration: przyjmuje graf oraz zmienne (w postaci listy), po których ten graf będzie całkowany, zwraca zcałkowany graf. Całkuje po zmiennych zgodnie z ich ułożeniem w liście. Funkcja ta wywołuje funkcję extension dla każdej podanej zmiennej. Na koniec dodaje stałą C.

-calculate: przyjmuje graf, zmienną oraz jej wartość, zwraca wartość obliczoną poprzez wykonanie obliczeń zapisanych w grafie.

-integrate: przyjmuje wielomian w postaci ciągu znaków oraz listę zmiennych, zwraca ciąg znaków reprezentujący graf zcałkowany w sposób nieoznaczony. Całkowanie odbywa się po zmiennych i w kolejności ich zapisania w liście.

-integrateFile: przyjmuje ścieżkę do pliku wejściowego, ścieżkę do pliku wyjściowego i wartość bool (wartości te są opcjonalne, domyślnie są ustawione na input="input.txt", output="output.txt", r=False). Otwiera plik podany w ścieżce wejściowej, odczytuje z niego informacje dotyczące wielomianu a następnie całkuje w sposób nieoznaczony a wynik zapisuje w pliku podanym w ścieżce wyjściowej. Kiedy wartość r ustawiona jest na True zwraca też ciąg znaków reprezentujący wynik (zcałkowany wielomian).

-definitiveIntegration: Przyjmuje wielomian w postaci ciągu znaków, dwie wartości oznaczające początek i koniec przedziału całkowania (jako osobne zmienne) oraz zmienną, po której dokonywane jest całkowanie, zwraca wartość całkowania oznaczonego wykonanego na wielomianie. Funkcja ta działa tylko dla wielomianów jednej zmiennej.

-definitiveIntegrationFile: przyjmuje ścieżkę do pliku wejściowego, ścieżkę do pliku wyjściowego i wartość bool (wartości te są opcjonalne, domyślnie są ustawione na input="input.txt", output="output.txt", r=False). Otwiera plik podany w ścieżce wejściowej, odczytuje z niego informacje dotyczące wielomianu a następnie całkuje w sposób oznaczony a wynik zapisuje w pliku podanym w ścieżce wyjściowej. Kiedy wartość r ustawiona jest na True zwraca też wynik (wartość całkowania). Funkcja ta działa tylko dla wielomianów jednej zmiennej.

Instrukcja użytkowania

Aby korzystać z tej biblioteki potrzeba zaimportować tak naprawdę tylko cztery funkcje: integrate, integrateFile, definitiveIntegration, definitiveIntegrationFile, gdyż to one sterują resztą i dostarczają wszystkie potrzebne wartości. To one odpowiadają za dokonywanie kompletnych obliczeń i ich zwracanie/zapisywanie.

Plik do odczytu
Ważnym jest aby poprawnie zapisać wartości w pliku odczytywanym przez funkcje. Przede wszystkim spacje (pojedyncze) służą do rozdzielania poszczególnych argumentów należy więc nie umieszczać ich w wielomianie czy zmiennych tak jak tu:

1+(1/2)*x-2 *x^2+1* x^3

poprawnie będzie:

1+(1/2)*x-2*x^2+1*x^3

Każdy argument z osobna powinien być zwartym blokiem.
Należy wykorzystywać znaki takie jak +, -,* , /, ^, (,), ponieważ program jest przystosowany do ich poprawnego rozdzielania, należy unikać sytuacji gdzie mnożenie zapisane jest w taki sposób: 2x, poprawnie będzie: 2*x.
Poprawna struktura pliku wejściowego dla funkcji definitiveIntegrationFile będzie miała postać:

1+(1/2)*x-2*x^2+1*x^3 3,9 dx

Należy zwrócić uwagę na d poprzedzające zmienną x.

Poprawna struktura pliku wejściowego dla funkcji integrateFile będzie miała postać:

1+(1/2)*x*y-2*x^2+1*x^3*y^2 dxdy

Tutaj symbol d rozdziela zmienne. Brak też wartości początku i końca przedziału całkowania ponieważ jest to całka nieoznaczona.

Argumenty funkcji
Zmienne podawane są jako wartości znakowe np. „x”, nie samo x. To samo aplikuje się do list zmiennych.
Przykładowe wywołanie funkcji integrate:

integrate("1+(1/2)*x-2*x^2*y^3+1*x^3", ["x", "y"])

Należy pamiętać, że ta funkcja zwraca wartości.

Przykładowe wywołanie funkcji definitiveIntegration:

definitiveIntegration("1+(1/2)*x-2*x^2+1*x^3", 3, 9, "x")

Należy pamiętać, że ta funkcja zwraca wartości.