Drzewa

Program zaliczeniowy nr 1 z przedmiotu *Algorytmy i Struktury Danych*

semestr letni 2017/2018

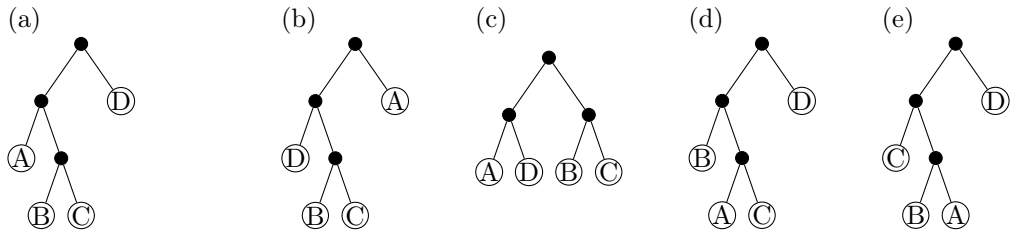
Wstęp

Rozważamy drzewa binarne z liściami etykietowanymi napisami.

Załóżmy, że wierzchołki u i v są synami wierzchołka w, przy czym u jest wierzchołkiem wewnętrznym. Operację zamiany poddrzewa zawieszonego w v z poddrzewem zawieszonym w jednym z synów u nazwiemy *przeszczepem*. Dwa drzewa nazwiemy *podobnymi*, jeśli jedno można otrzymać z drugiego wykonując przeszczep.

Przykład

Rys. 1 ilustruje powyższe definicje.



Rysunek 1: Przykładowe drzewo binarne (a) i wszystkie drzewa do niego podobne (bcde). Drzewa (bc) powstają z (a) wskutek przeszczepów w korzeniu, natomiast drzewa (de) wskutek przeszczepów w lewym synu korzenia.

Zadanie

Należy napisać moduł *ImieNazwisko.py* (najlepiej własne;) zawierający klasy `Leaf`, `BinNode` oraz `BinTree` implementujące odpowiednio liście, wierzchołki wewnętrzne oraz drzewa binarne.

Klasa `Leaf` powinna zawierać następujące metody:

- `__init__(label)` utwórz liść z etykietą `label`
- `is_leaf()` – zwróć `True`
- `label()` – zwróć etykietę

Klasa `BinNode` powinna zawierać następujące metody:

- `__init__(left,right)` utwórz wierzchołek wewnętrzny z synami `left` i `right`
- `is_leaf()` – zwróć `False`
- `son(which)` – zwróć lewego syna, gdy `which='L'`, a prawego, gdy `which='R'`
- `graft(first,second)` – wykonaj przeszczep; przy oznaczeniach z definicji przeszczepu aktualny wierzchołek to w, argumenty wskazują na v (`first`) oraz wybranego syna u (`second`); wartości argumentów jak w metodzie `son`

Klasa `BinTree` powinna zawierać następujące metody:

- `__init__(node)` – utwórz drzewo z wierzchołkiem `node` w korzeniu
- `root()` – zwróć korzeń drzewa
- `similar()` – zwróć wszystkie drzewa podobne do aktualnego
- `height()` – zwróć wysokość aktualnego drzewa
- `max_height()` – używając strategii zachłannej spróbuj serią przeszczepów przekształcić aktualne drzewo w drzewo o maksymalnej wysokości
- `min_height()` – używając strategii zachłannej spróbuj serią przeszczepów przekształcić aktualne drzewo w drzewo o minimalnej wysokości

Metoda `similar` powinna być zaimplementowana za pomocą *generatora* działającego w miejscu, tzn. powinna zwracać *iterator* modyfikujący na bieżąco drzewo, na którym została wywołana, a na koniec przywrócić je do stanu wyjściowego.

Strategia zachłanna w metodach `max_height()` oraz `min_height()` polega na iteracyjnym zastępowaniu aktualnego drzewa optymalnym spośród podobnych do niego, dopóki ta operacja poprawia wysokość.

Przykład

Polecenie

```
> t = BinTree(BinNode(BinNode(Leaf('A'),BinNode(Leaf('B'),Leaf('C'))),Leaf('D')))
```

powinno spowodować utworzenie drzewa z Rys. 1(a), natomiast polecenie

```
> t.root().son('L').draft('L','R')
```

powinno przekształcić je w drzewo z Rys. 1(e). Polecenie

```
> t.max_height()
```

wykonane na wyjściowym drzewie t powinno pozostawić je niezmienione, a polecenie

```
> t.min_height()
```

powinno przekształcić je w drzewo z Rys. 1(c).

Ocena

Rozwiązanie zadania powinno zawierać kod programu z komentarzami oraz raport z analizą problemu: *Czy strategia zachłanna gwarantuje utworzenie optymalnego drzewa?* w metodach `max_height()` oraz `min_height()`.

Ocena rozwiązania:

15 pkt. pełne rozwiązanie

13 pkt. kod bez raportu

10 pkt. kod bez metod `max_height()` i `min_height()`

8 pkt. kod bez metod `max_height()` i `min_height()` oraz ze zwracaniem listy drzew w metodzie `similar()`

5 pkt. kod bez metod `max_height()`, `min_height()` i `similar()`