Dokumentacja projektu semestralnego

Algorytm binarnego drzewa poszukiwań ze wskaźnikiem do rodzica

Autor: Marcin Chamera

Spis treści:

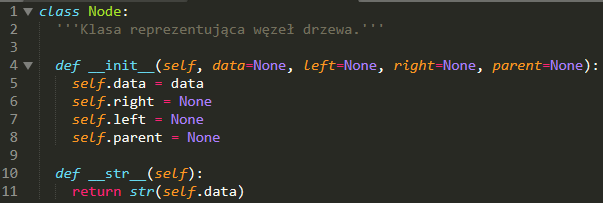
1. Wprowadzenie z opisem teoretycznym algorytmu
2. Interfejs
3. Implementacja
4. Wyniki testów
5. Wprowadzenie z opisem teoretycznym algorytmu

Binarne drzewo poszukiwań (z ang. Binary Search Tree) to specjalny rodzaj drzewa binarnego w którym wartości wszystkich węzłów lewego poddrzewa jakiegokolwiek węzła w drzewie są mniejsze niż wartość tego węzła. Dodatkowo, wartości wszystkich węzłów prawego poddrzewa jakiegokolwiek węzła są większe niż wartość tego węzła. Algorytm BST przedstawiony na laboratoriach został przeze mnie dodatkowo rozszerzony o wskaźnik do rodzica węzła.

1. Interfejs

Program został napisany w celu zaprezentowania działania algorytmu i na wyjściu wyświetla jedynie wyniki metod, jakie zostały wywołane w głównej funkcji programu (\_\_main\_\_).

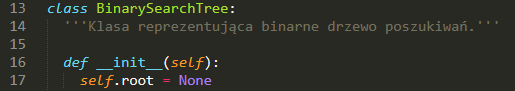
1. Implementacja

Algorytm, który opisuje ta dokumentacja, został zaimplementowany przy pomocy dwóch klas: Node i BinarySearchTree. Na poniższych zrzutach ekranu opisane pola i metody tych klas.

W tej implementacji algorytmu wszystkie mechaniki związane z drzewem zostały przeniesione do klasy BinarySearchTree. Klasa Node służy jedynie do reprezentacji pojedynczego węzła należącego do drzewa. Posiada ona 4 pola:

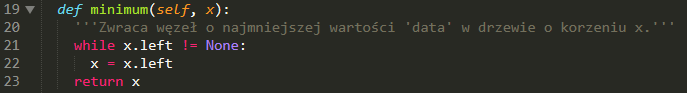
* data – przetrzymuje jakąś informacje zapisaną w węźle (np. liczbę całkowitą, napis etc.)
* right – „wskaźnik” do prawego potomka
* left – „wskaźnik” do lewego potomka
* parent – „wskaźnik” do rodzica

Klasa Node zaimplementowane posiada jedynie metody specjalne \_\_init\_\_, czyli konstruktor oraz \_\_str\_\_, która zamienia klasę na „nieformalny” string.

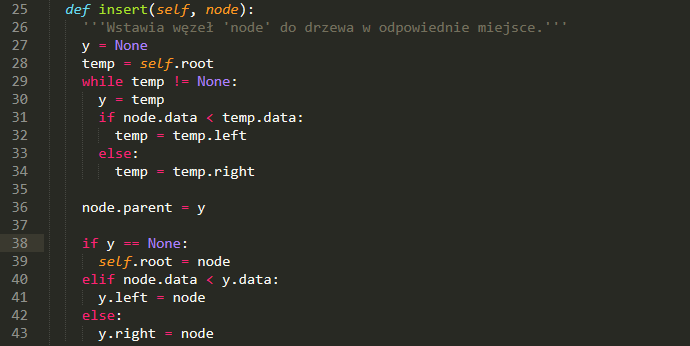


Klasa BinarySearchTree posiada zaimplementowaną jedną metodę specjalną, czyli jej konstruktor \_\_init\_\_. Klasa posiada także jedno pole nazwane root, które reprezentuje korzeń tego drzewa. Wewnątrz konstruktora korzeń zostaje ustawiony na None, to znaczy drzewo zostaje zainicjowane bez żadnego przypisanego do niego węzła.

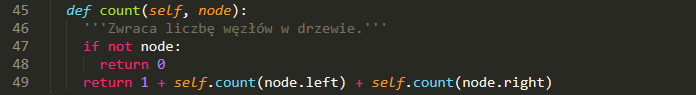
W dalszej części dokumentacji przy podawaniu ilości przekazywanych parametrów do metody nie będę wspominał o parametrze self.



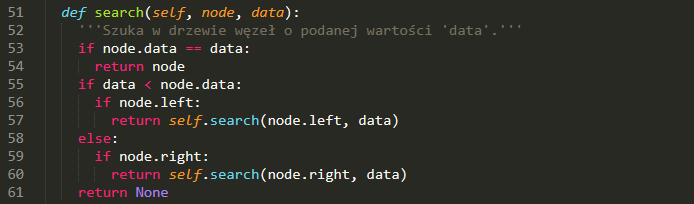
Metoda minimum wyszukuje w drzewie węzeł o najmniejszej wartości. Przyjmuje jeden parametr x, który jest korzeniem wybranego drzewa, w którym szukamy węzła o minimalnej wartości data. W pętli while przechodzimy kolejno do lewych potomków obecnego węzła x, dopóki nie znajdziemy skrajnie lewego potomka w wybranym drzewie. Metoda na koniec zwraca skrajnie lewego potomka.



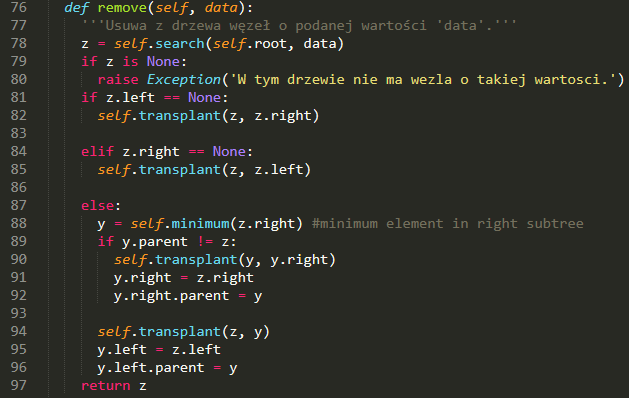
Metoda insert wstawia nowy węzeł do drzewa. W binarnym drzewie poszukiwań wstawiany węzeł nie może zostać umieszczony w dowolnym miejscu ze względu na konieczność zachowania struktury takiego drzewa. W celu wstawienia elementu, najpierw należy znaleźć miejsce, w którym struktura drzewa zostanie zachowana. W tym celu został użyty wskaźnik temp, który kolejno przechodzi przez potomków, zaczynając od korzenia. Ostatni węzeł w iteracji pętli należy uczynić rodzicem nowo wstawianego węzła. W tym celu została użyta zmienna y. W wypadku, gdy drzewo nie ma żadnego węzła, nowy węzeł będzie korzeniem tego drzewa, a wskaźnik jego rodzica nie będzie na nic pokazywał. Dlatego na początku wartość y zostaje ustawiona na None. Na końcu algorytm ustawia nowy węzeł jako dziecko y. Jeśli y jest nullem, nowy węzeł będzie korzeniem drzewa. W innym wypadku następuje sprawdzenie czy wartość w data nowego węzła jest większa czy mniejsza niż wartość w data y i odpowiednio będzie on lewym lub prawym dzieckiem.



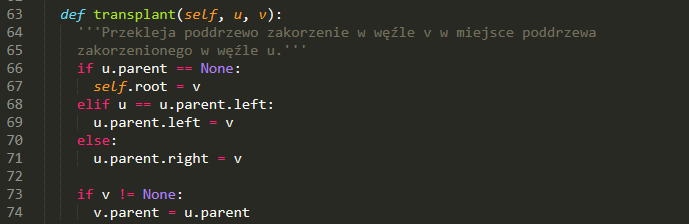
Metoda count zwraca liczbę węzłów w drzewie. Przyjmuje jeden parametr node. Przy pierwszym wywołaniu metody jako parametr node podaje się korzeń drzewa. W wypadku, gdyby korzeń był pusty, metoda zwraca 0. W przeciwnym wypadku, metoda działa rekurencyjnie. Do liczby obecnie zliczonych węzłów dodaje 1 i wywołuje się dla lewego i prawego potomka węzła node. Gdy wszystkie wywołania metody będą już dla nieistniejących węzłów, zwracana jest liczba zliczonych węzłów.



Metoda search szuka w drzewie węzeł o danej wartości. Przyjmuje dwa parametry node i data. Metoda działa rekurencyjnie, gdzie przy pierwszym wywołaniu jako parametr node podawany jest korzeń drzewa. Data to szukana wartość. W przypadku, gdy wartość data obecnie rozpatrywanego węzła jest równa wartości data przekazanej w parametrze, ten węzeł jest zwracany. W przypadku, gdy wartość data obecnie rozpatrywanego węzła jest większa od wartości data przekazanej w parametrze, metoda wywołuje siebie samą z tym samym parametrem data, a jako parametr node przekazywany jest lewy potomek obecnego węzła. W przypadku, gdy wartość data obecnie rozpatrywanego węzła jest mniejsza od wartości data przekazanej w parametrze, metoda wywołuje siebie samą z tym samym parametrem data, a jako parametr node przekazywany jest prawy potomek obecnego węzła. W innym przypadku zwracana jest wartość None.

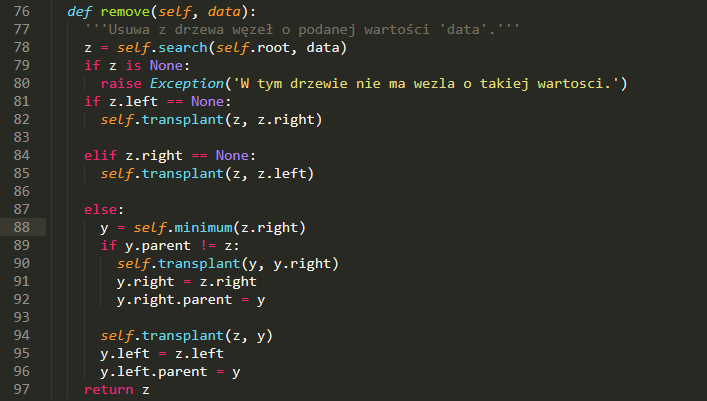


Metoda remove usuwa z drzewa wybrany węzeł. Metoda przyjmuje jeden parametr data, który jest wartością, jaką ma węzeł do usunięcia. Do usunięcia węzła z BST wykorzystana została dodatkowo metoda transplant, która pozwoli zamienić wybrane poddrzewo z innym. Innymi słowy, przeszczepi jedno poddrzewo w miejsce innego.

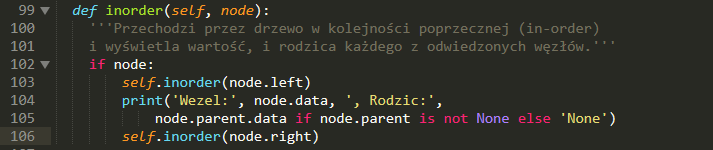


Metoda transplant przyjmuje dwa parametry u i v, gdzie v jest korzeniem drzewa, które będzie przeklejone, a u jest korzeniem drzewa, które zostanie zastąpione. Zadaniem metody transplant jest sprawić, aby węzeł v stał się potomkiem rodzica węzła v, to znaczy jeśli u jest lewym potomkiem, to v zostanie lewym potomkiem rodzica u. Podobnie, jeśli u jest prawym potomkiem, wtedy v zostanie prawym potomkiem rodzica u. Istnieje też możliwość, że u nie ma żadnego rodzica, to znaczy u jest korzeniem drzewa. W tym przypadku, v zostanie korzeniem tego drzewa. Wracając do metody remove. Zakładając, że węzeł do usunięcia jest liściem, łatwo można ten węzeł usunąć, ustawiając rodzica tego węzła na None. Aby usunąć węzeł z tylko jednym potomkiem przeszczepiamy tego potomka do węzła, co nie zakłóci struktury BST. W przypadku, gdy węzeł do usunięcia ma obu potomków, szukamy najmniejszego elementu prawego poddrzewa tego węzła i podmieniamy go z węzłem do usunięcia. Najmniejszy element prawego poddrzewa będzie albo nie miał potomków, albo będzie miał jednego potomka ponieważ jeśli ma lewego potomka, to wtedy nie będzie najmniejszym elementem. Z tego powodu taki węzeł można spokojnie usunąć.

Przypominając jeszcze raz metodę remove:



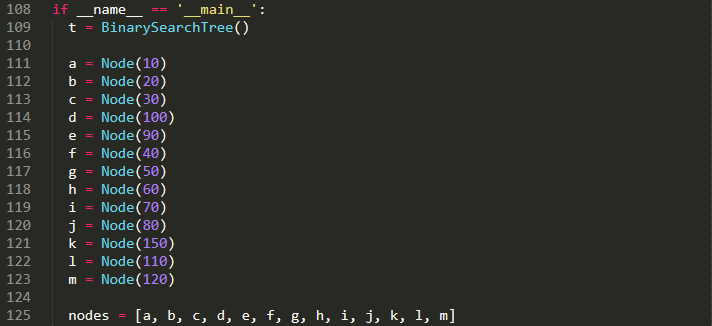
Na początku wyszukiwany jest węzeł o wartości podanej w parametrze. Następnie sprawdzana jest liczba potomków węzła z. Jeśli węzeł z nie ma lewego potomka, wtedy węzeł ten ma albo wyłącznie prawego potomka, albo nie ma ich wcale. W obu tych wypadkach, przy pomocy metody transplant, przeszczepiany jest do niego jego prawy potomek. Podobnie sprawdzane jest, czy węzeł posiada prawego potomka, czy nie i wykonywane jest przeszczepienie jego lewego potomka w miejsce tego węzła. Jeśli żaden z powyższych warunków nie jest spełniony, oznacza to, że węzeł z ma obu potomków. Znajdowany jest wtedy węzeł z minimalną wartością w prawym poddrzewie i przypisywany do zmiennej y. Teraz należy wstawić węzeł y w miejsce z. Najpierw przeklejany jest prawy potomek y w miejsce y, a następnie brane jest prawe poddrzewo węzła z i przeszczepienie go w taki sposób, aby było prawym poddrzewem y. Po tym operacjach, y przeszczepiany jest w miejsce z. Możliwe jest także, że najmniejszy węzeł jest bezpośrednim potomkiem węzła z. W tym wypadku, jedynie y jest przeszczepiany w miejsce z. Po tym, lewy potomek węzła y zostaje lewym potomkiem węzła z.



Metoda inorder przechodzi przez drzewo w kolejności poprzecznej (in-order) i wyświetla wartość data, i wartość data rodzica każdego z odwiedzonych węzłów. Metoda przyjmuje jeden parametr node, dla którego w ciele metody zostanie wyświetlona wartość jego pola data. Najpierw sprawdzany jest warunek, czy taki węzeł istnieje. Jeśli tak, to najpierw wywoływana jest rekurencyjnie ta metoda, gdzie w parametrze przekazywany jest lewy potomek. Następnie wyświetlana jest wartość data węzła node i wartość data rodzica węzła node, ale tylko jeśli rodzic istnieje (czyli jeśli node nie jest korzeniem). Na końcu wywoływana jest rekurencyjnie ta metoda, gdzie w parametrze przekazywany jest prawy potomek.

1. Wyniki testów

Algorytm został poddany testowi poprawności, które zostały umieszczone w bloku instrukcji warunkowej *if\_\_name == ‘\_\_main\_\_’:*. Testy zostały przeprowadzone, wywołując funkcje dla pewnych konkretnych danych przedstawionych na poniższych zrzutach ekranu.



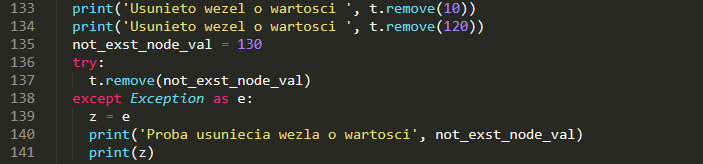
Na początku tworzona jest instancja klasy BinarySearchTree, a po niej paręnaście instancji klasy Node z przykładowymi wartościami. Te instacje klasy Node (węzły) zostały wstawione do listy nodes.



Następnie w pętli węzły wstawiane są do drzewa, po czym wywołana zostaje metoda inorder w celu wyświetlenia aktualnego wyglądu drzewa.



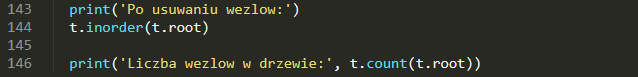
Tak prezentuje się wynik powyższych instrukcji.



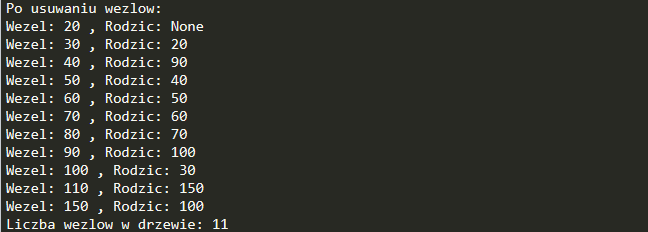
Następnie wywołana zostaje dwa razy metoda remove, a jej wynik zostaje wyświetlony na ekran. Później następuje próba usunięcia węzła o takiej wartości, która nie istnieje w stworzonym drzewie.



Tak prezentuje się wynik powyższych instrukcji.



Na końcu zostaje jeszcze raz wywołana metoda inorder, w celu wyświetlenia zawartości drzewa po usunięciu dwóch węzłów, a następnie zostaje wywołana metoda count.



Tak prezentuje się wynik powyższych instrukcji.