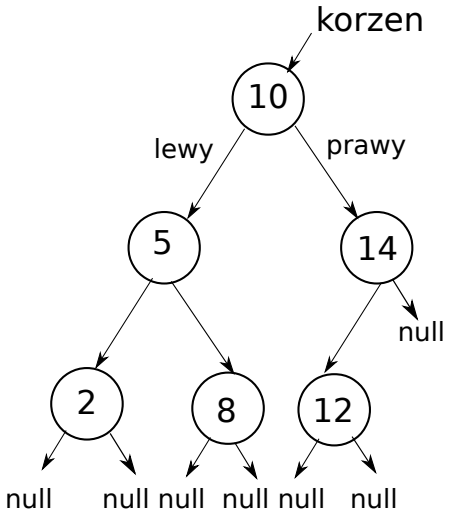
Drzewa BST i AVL

Drzewa poszukiwań binarnych (BST)

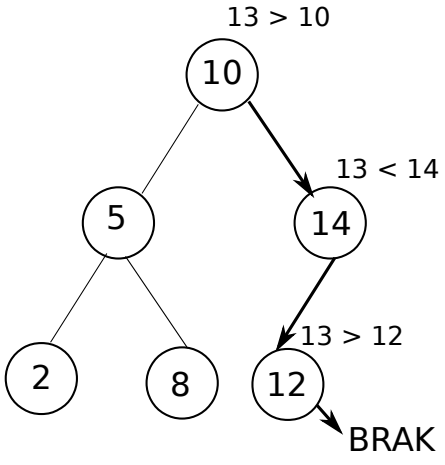
Drzewo BST to dynamiczna struktura danych (w formie drzewa binarnego), która ma tą właściwość, że dla każdego elementu wszystkie elementy w jego prawym poddrzewie są od niego większe, a wszystkie elementy w lewym poddrzewie są od niego mniejsze. Takie uporządkowanie struktury umożliwia w niej wyszukiwanie elementów w sposób analogiczny do wyszukiwania binarnego przy jednoczesnym zachowaniu elastyczności struktury (możliwość szybkiego dodawania lub usuwania elementów).
Drzewo BST jest strukturą, którą implementuje się za pomocą dowiązań (wskaźniki/referencje), a pamięć dla poszczególnych węzłów jest alokowana dynamicznie. Program operujący na drzewie BST potrzebuje jedynie informacji o jego korzeniu (wskaźnik na pierwszy element drzewa). Każdy węzeł poza danymi potrzebuje jeszcze wskazań na lewego i prawego syna. Braki poddrzew oznaczane są symbolem *null*. Poniżej przykład drzewa BST oraz definicji struktury węzła (C++):

```
struct Wezel{
     int liczba;
     Wezel* lewy;
     Wezel* prawy;
};
```

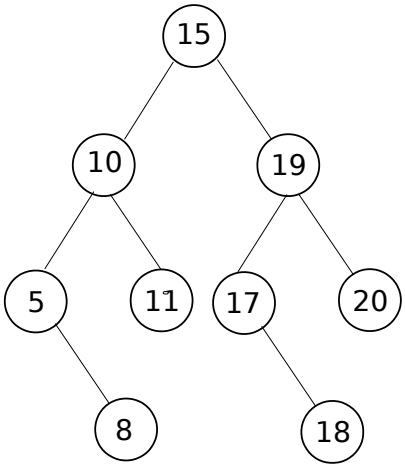


Wyszukiwanie w drzewie BST polega na porównywaniu szukanego elementu do aktualnego węzła i w zależności od tego, który element jest większy następuje przesunięcie do prawego lub lewego poddrzewa. Poszukiwanie jest przerywane, gdy element zostaje znaleziony lub osiągnięty zostanie kres drzewa (*null*). Poniżej przykład wyszukania elementu 13 w drzewie oraz kod (C++):

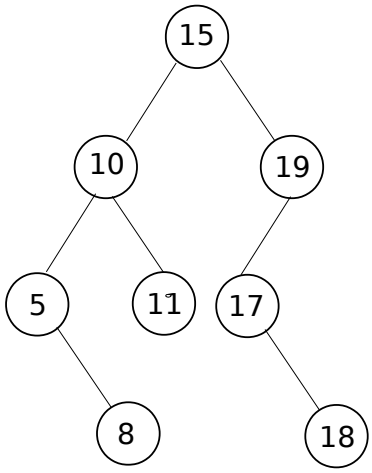
```
bool szukaj(int n, Wezel* korzen){
     while(korzen!=NULL){
          if(korzen->liczba==n)
               return true;
          if(n>korzen->liczba)
               korzen=korzen->prawy;
          else
               korzen=korzen->lewy;
     }
     return false;
}
```



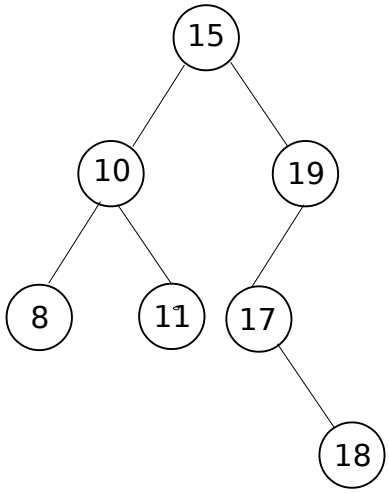
Dodawanie do drzewa BST odbywa się analogicznie jak wyszukiwanie. Najpierw szukamy miejsca, w którym element należy wstawić (faza wyszukiwania), a następnie jest on dodawany w do drzewa (o ile nie występuje już w drzewie). W przykładzie powyżej 13 trafiłaby na prawo od elementu 12, 7 zostałaby umieszczona na lewo od elementu 8, a 100 trafiłoby na prawo od elementu 14. Poniżej przedstawiono drzewo BST powstałe po dodaniu kolejno liczb: 15, 10, 19, 17, 20, 5, 8, 11, 18:



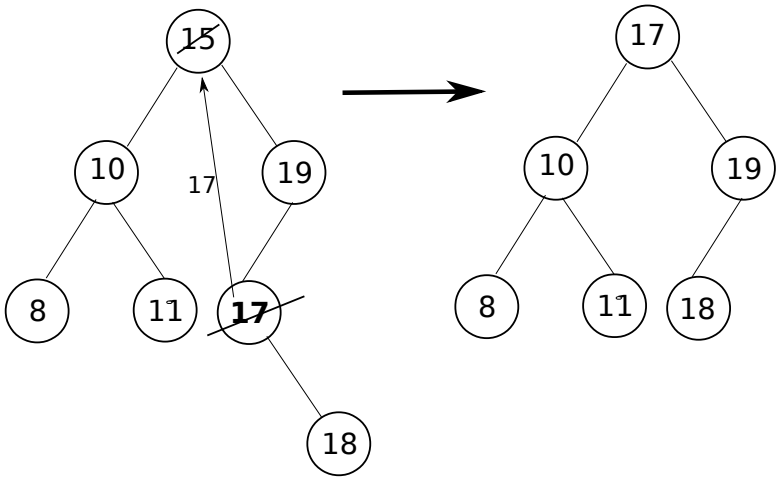
Operacja usuwania w pierwszej fazie przebiega tak jak wyszukiwanie – należy znaleźć element do usunięcia. Jeżeli element występuje w drzewie to należy podjąć odpowiednie kroki. W przypadku, gdy element nie ma synów (jest liściem) to po prostu zwalniamy pamięć przypisujemy *null* w odpowiednim poddrzewie jego ojca. Poniżej drzewo po usunięciu 20:



W drugim przypadku usuwany element na jedno dziecko. Wówczas proces usuwania przypomina usuwanie elementu z listy jednokierunkowej: łączy się ojca usuwanego elementu z jedynym dzieckiem tego elementu. Po usunięciu 5 drzewo wygląda więc następująco:



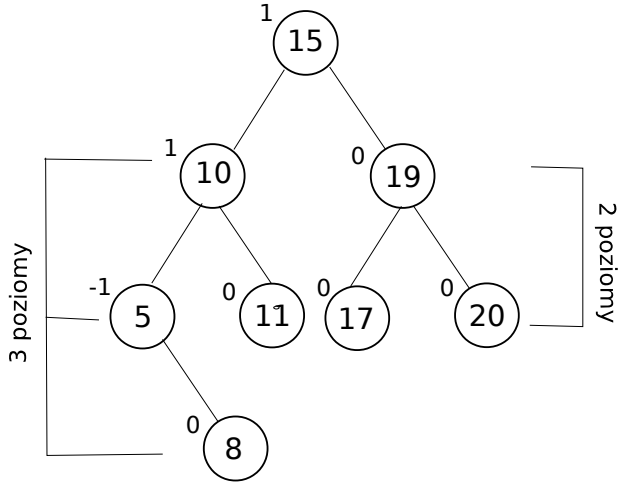
Najtrudniejszy jest trzeci przypadek, gdy usuwany element na dwójkę synów. Wówczas zostąpić go mogą tylko dwa elementy (by dalej zachowany był porządek BST): jego poprzednik i następnik. Są to elementy sąsiednie z usuwanym w hierarchii wszystkich elementów w drzewie. Ich znalezienie nie jest trudne: poprzednik jest największym węzłem w lewym poddrzewie, a następnik najmniejszym węzłem w prawym poddrzewie. W dalszych założeniach przyjmujemy, że usuwany element będzie zastępowany przez swego następnika, a fizycznie z drzewa usuwany jest węzeł następnika (a nie węzeł usuwanego elementu). W przypadku usunięcia z drzewa 15 widzimy, że jego poprzednik to 11, a jego następnik to 17 (hierarchia: 8, 10, 11, 15, 17, 18, 19). W tej sytuacji liczba z następnika (17) zastępuje 15 w korzeniu (nie usuwamy węzła, w którym było 15), a sam węzeł, w którym była 17 jest fizycznie usuwany z drzewa. Zauważmy, że węzeł z następnika może mieć co najwyżej jedno dziecko (czemu?), a więc rozważane są tutaj prostsze przypadki usuwania. Poniżej proces usunięcia 15 z drzewa:



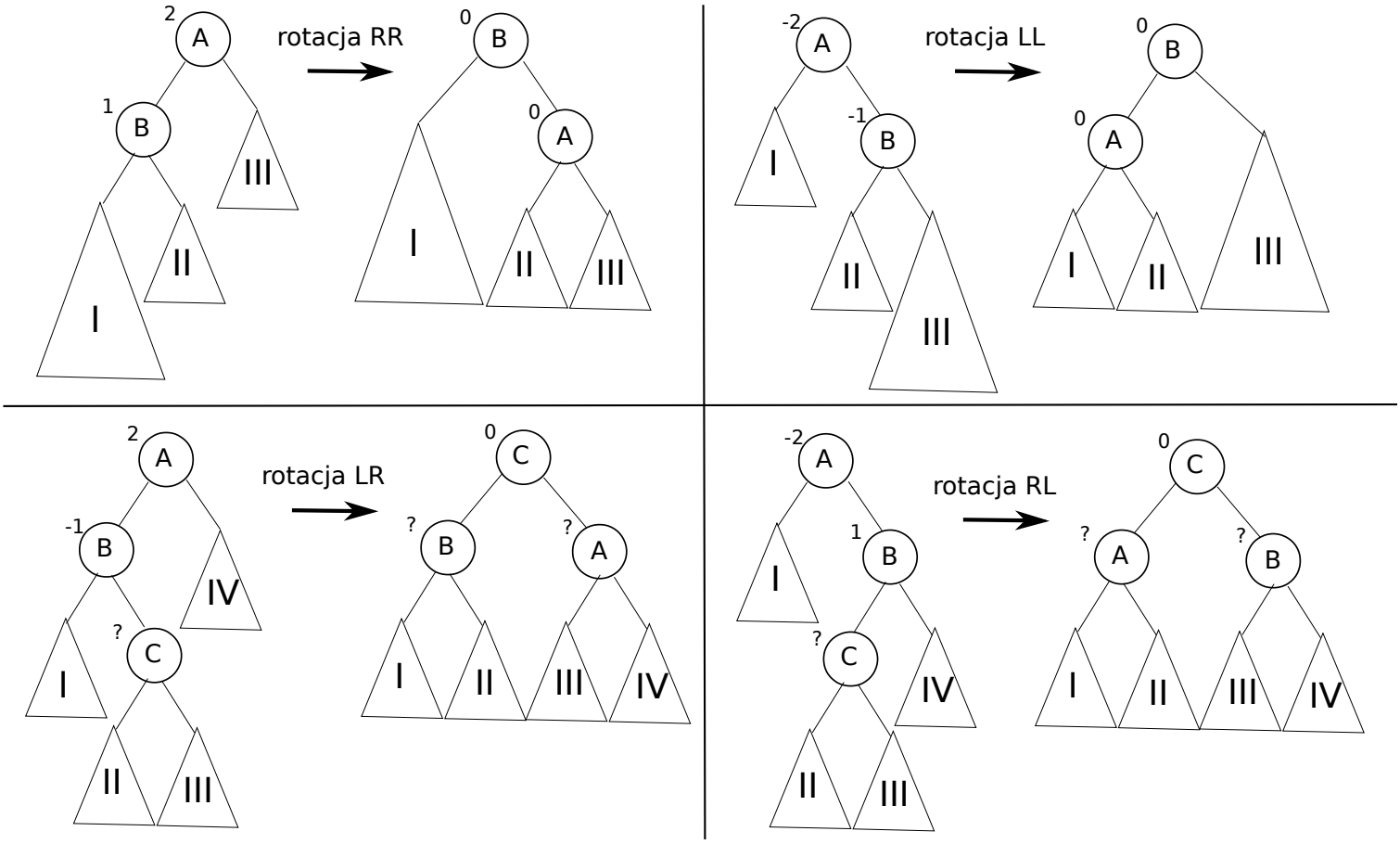
Złożoność wszystkich operacji na drzewie BST jest wprost proporcjonalna do liczby poziomów w drzewie. W przypadku dobrze zrównoważonego drzewa złożoność jest więc logarytmiczna, ale w pewnych przypadkach (np. przy dodaniu do drzewa BST elementów posortowanych) drzewo może zdegenerować się do listy i złożoności podstawowych operacji będą liniowe. Aby zapobiec takiemu przypadkowi stosuje się pewne modyfikacje w drzewie BST, a przykładem jest drzewo AVL.

Drzewo AVL

Drzewo AVL jest odmianą drzewa BST, w której dla każdego węzła zachodzi dodatkowy warunek związany ze strukturą drzewa: różnica pomiędzy liczbą poziomów lewego i prawego poddrzewa wynosi co najwyżej 1. Drzewo spełniające ten warunek w każdym węźle ma gwarantowaną logarytmiczną liczbę poziomów (w zależności od liczby elementów drzewa), co daje logarytmiczną pesymistyczną złożoność podstawowych operacji. Informacja o zrównoważeniu danego poddrzewa jest przechowywana w każdym węźle w postaci wag będących różnicą pomiędzy liczbą poziomów lewego i prawego poddrzewa. W prawidłowym drzewie AVL wagi należą więc do zbioru {-1, 0, 1}. Poniżej przykład drzewa AVL wraz z wagami. Przykładowo waga elementu 15 wynosi 1, ponieważ jego lewe poddrzewo ma 3 poziomy, a prawe poddrzewo ma 2 poziomy (3-2=1).

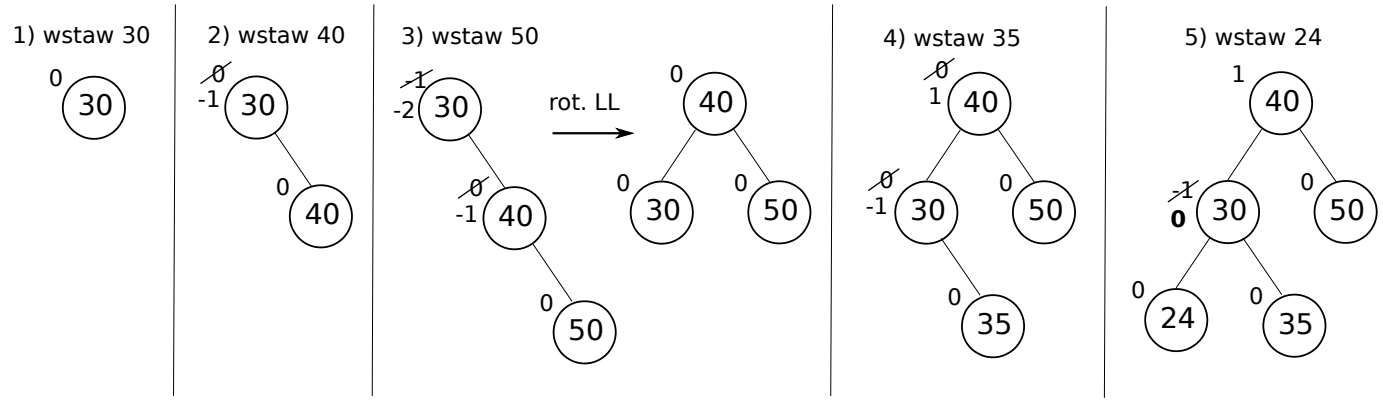


Najważniejszym mechanizmem w drzewie AVL jest utrzymywanie zrównoważenia drzewa po operacjach modyfikacji (wstawianie i usuwanie elementów). Pierwsza faza tych operacji przebiega identycznie jak w drzewie BST. W drugiej fazie, poczynając od miejsca modyfikacji (wstawienia/usunięcia węzła), następuje powrót w górę drzewa (w stronę korzenia) i zmieniane są wagi napotkanych węzłów oraz następują tzw. rotacje (wyważanie drzewa). Po wstawieniu elementu, jeżeli następuje powrót z lewego poddrzewa, to waga aktualnego węzła jest zwiększana o jeden (przybyło z lewej strony). Jeżeli natomiast wracamy z prawego poddrzewa to wagę należy zmniejszyć o jeden (przybyło z prawej strony). W przypadku usunięcia jest dokładnie odwrotnie: wracając z lewego poddrzewa zmniejszamy o jeden wagę aktualnego węzła, a wracając z prawego poddrzewa zwiększamy o jeden wagę. Jeżeli po modyfikacji wagi została ona ustawiona na zero (w przypadku wstawiania), to proces wyważania jest przerywany (oznacza to, że dane poddrzewo po dodaniu elementu się zrównoważyło, ale jego całkowita liczba poziomów się nie zmieniła). Jeżeli po modyfikacji waga została ustawiona na 1 lub -1 (w przypadku usuwania), to także należy przerwać proces wyważania. Jeżeli po modyfikacji waga bieżącego węzła wynosi 2 lub -2, to oznacza konieczność przeprowadzenia rotacji, czyli zmiany struktury drzewa. Istnieją 4 typy rotacji. Poniżej ich schematy:

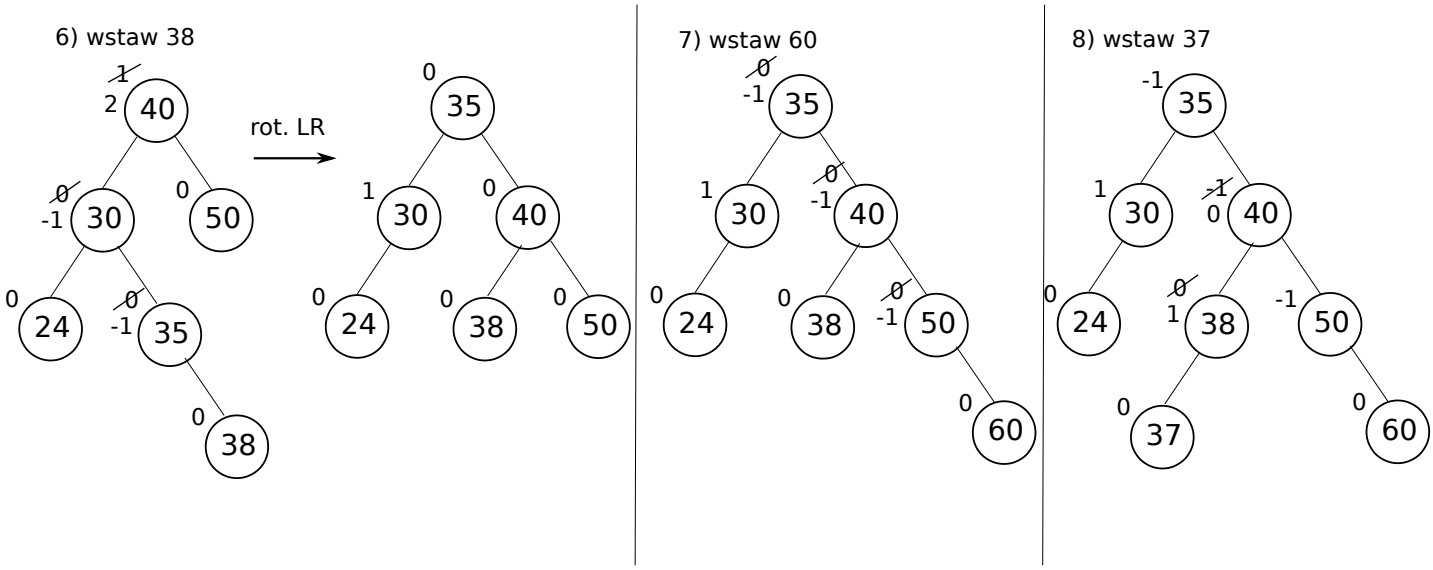


W miejsce liter A, B i C można wstawić dowolne elementy drzewa, natomiast symbole I, II, III i IV oznaczają dowolne poddrzewa (może to być duże poddrzewo, ale również *null*). W wyniku rotacji zmieniają się wagi wybranych elementów drzewa oraz struktura drzewa (zmiana dowiązań). Same poddrzewa (ozn. Liczbami rzymskimi) nie ulegają wewnętrznym modyfikacjom, a zmienia się jedynie ich lokalizacja w drzewie. Dzięki temu, że liczba zmian jest nieduża, to jednak rotacja ma koszt stały. Widoczne jest, że rotacja LL jest odbiciem lustrzanym rotacji RR (to są tzw. pojedyncze rotacje). Podobnie rotacja RL jest odbiciem rotacji LR. W przypadku rotacji podwójnych (LR oraz RL) wagi elementów B i A po rotacji zależą od wagi elementu C przed rotacją (są trzy przypadki do rozpatrzenia). Istnieje jeszcze taki przypadek, gdy waga elementu B wynosi 0 (może do tego dojść po usuwaniu): wówczas wystarczy przeprowadzić pojedynczą rotację i odpowiednio zmienić wagi.

Poniżej przedstawiony został rozbudowany przykład dodania do drzewa AVL kilkunastu liczb oraz usunięcia kilku. Najpierw wstawiamy kolejno 30, 40, 50, 35 i 24.

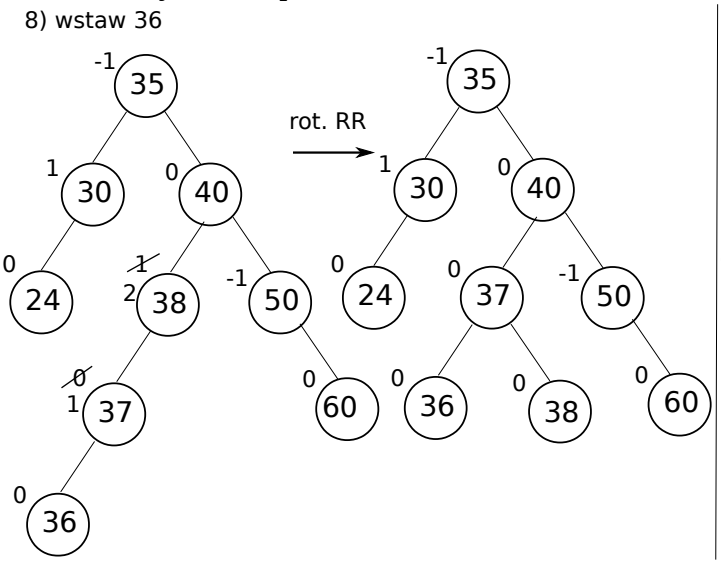


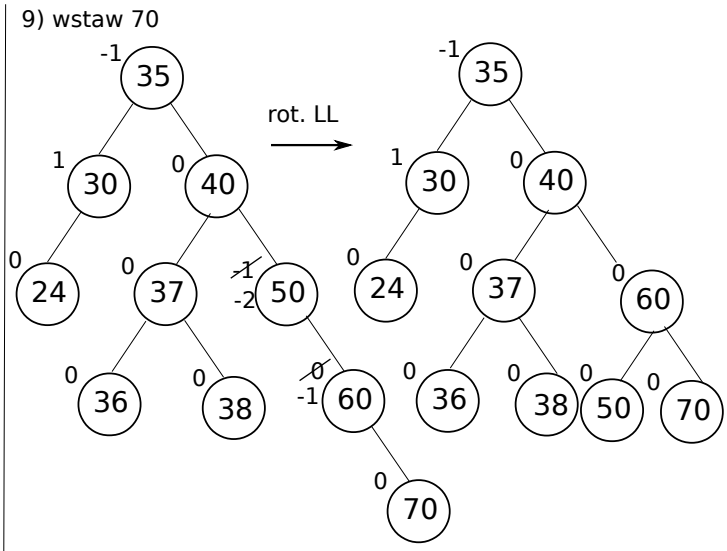
Miejsce wstawienia elementu wyszukiwane jest tak, jak w standardowym drzewie BST. Następnie wracając do korzenia zmieniamy o 1 wagi kolejnych węzłów. Przykładowo po wstawieniu 50 wracamy do węzłów 40 oraz 30 i zmniejszamy im wagi o jeden (ponieważ w obu przypadkach powrót nastąpił z prawego poddrzewa). W przypadku wstawienia 35 najpierw zmiejszamy o jeden wagę w elemencie 30 (powrót z prawej strony), a następnie zwiększamy o jeden wagę w elemencie 40 (powrót z lewej strony). Zauważmy, że po dodaniu 30 potrzebna była rotacja LL. W tym przypadku rolę elementu A pełni 30, a rolę elementu B pełni 40. Poddrzewa I i II są puste, a poddrzewo III to pojedynczy element (50). W przypadku dodania elementu 24 należało zwiększyć wagę elementu 30, co dało w rezultacie 0 (pogrubione). Wyzerowanie wagi po wstawieniu elementu oznacza, że należy przerwać dalsze wyważenie drzewa (stąd w 40 waga nie została zmieniona). Oznacza to, że po wstawieniu elementu potrzebna jest co najwyżej jedna rotacja (po jej wykonaniu korzeń danego poddrzewa będzie miał wagę 0).
Poniżej drzewo po wstawieniu elementów 38, 60 i 37:



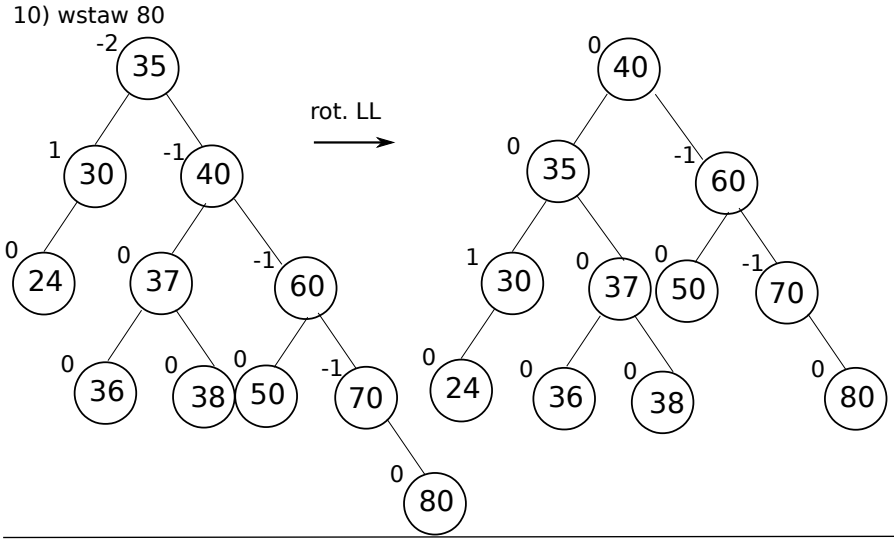
Po wstawieniu 38 następuje modyfikacja wag zgodnie z algorytmem (wracając z prawej strony wagę zmniejszamy o 1, a wracając z lewej strony zwiększamy o jeden). W korzeniu (40) został wykryty brak zrównoważenia (waga 2) i nastąpiła rotacja LR. W tym przypadku A to 40, B to 30, a C to 35. Poddrzewo I to element 24, poddrzewo II jest puste, poddrzewo III to element 38, a poddrzewo IV to element 50. Wstawienie 60 oraz 37 przebiega bez uruchomienia rotacji. Ważne jest jednak, że po wstawieniu 37 wracając do korzenia zmieniamy wagę elementu 40 na zero. W tym momencie ze względu na zerową wagę proces należy przerwać – w korzeniu wagi już nie zmieniamy (co widać również z rysunku).

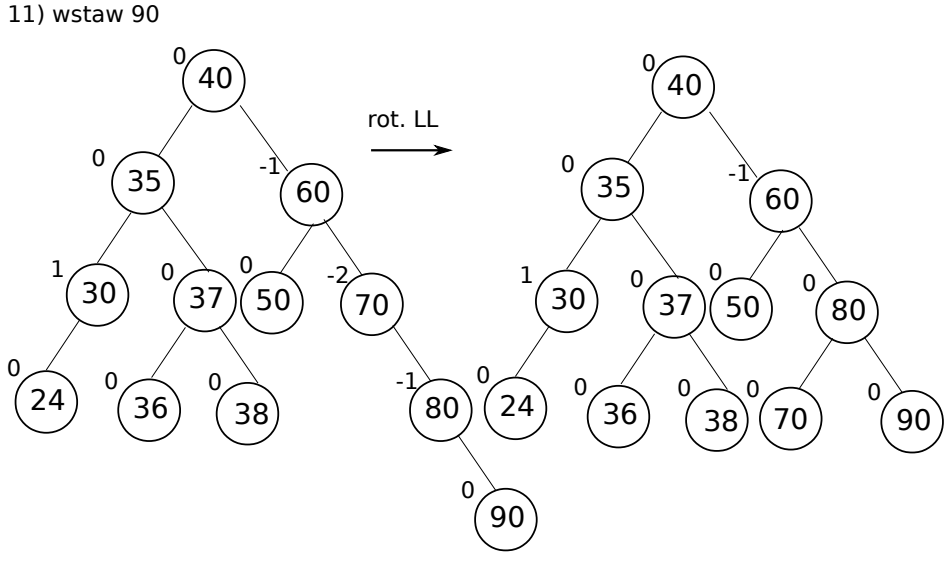
Poniżej drzewo po wstawieniu 36 i 70:

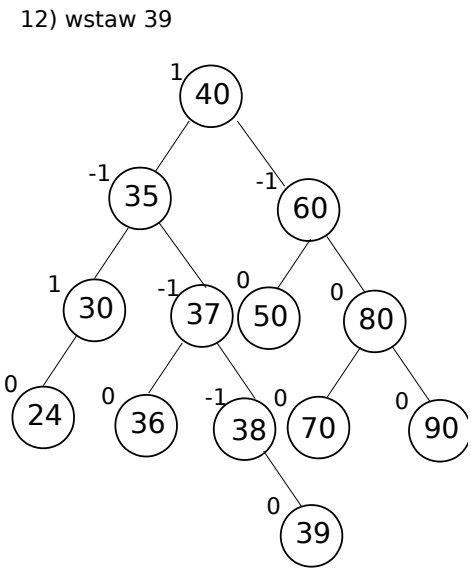




Po wstawieniu tych elementów zaistniała potrzeba wykonania rotacji (odpowiednio RR i LL). Rotacja ta nie wystąpiła w korzeniu drzewa (jak poprzednie), ale w pewnych poddrzewach (oznaczonych przez elementy 38 i 50). Po przeprowadzeniu rotacji drzewa się dalej nie wyważa (po rotacji pojawia się waga 0). Poniżej ostateczne drzewo po wstawieniu elementów 80, 90 i 39:

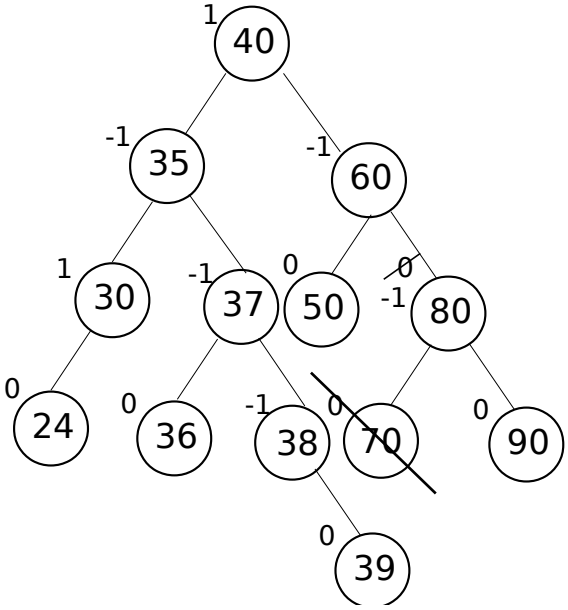






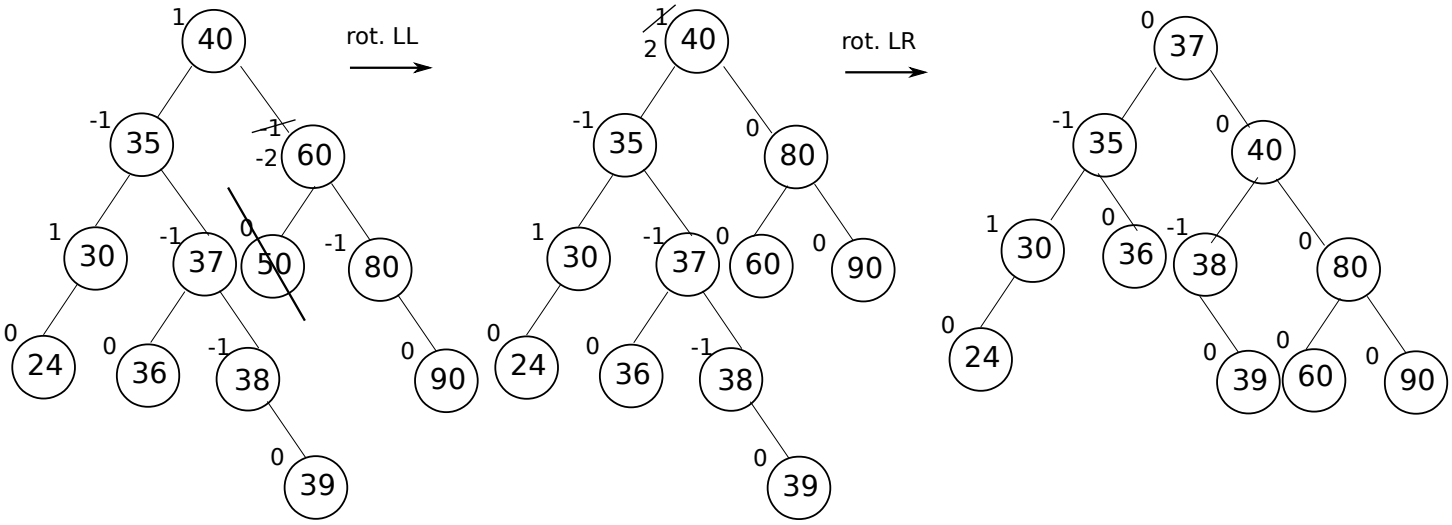
Terez usuńmy dwa elementy. Najpierw 70:

13) usun 70



Po usunięciu 70 przemieszczamy się w górę drzewa od usuniętego elementu. Do węzła 80 wróciliśmy z lewej strony, co oznacza, że wagę należy zmniejszyć o jeden (zmiana wag postępuje więc odwrotnie niż po wstawieniu elementu). Po zmianie wagi uzyskaliśmy jedną z wartości (-1 lub 1), które powodują przerwanie wyważania po operacji usuwania. Dalsza zmiana wag (w węzłach 60 i 40) nie ma sensu, co widać też z rysunku. Teraz usuniemy 50:

14) usun 50



Ten przypadek jest szczególnie interesujący. Początkowo usuwamy 50 i idziemy w górę zmieniając wagi. W elemencie 60 wagę należy zmniejszyć o 1 (przyszliśmy z prawej strony), co w rezultacie skutkuje wagą -2 i koniecznością rotacji LL w poddrzewie złożonym z elementów 60, 80 i 90. po przeprowadzeniu rotacji kierujemy się dalej w górę i zwiększamy wagę elementu 40 o jeden (powrót z lewej strony). To z kolei powoduje konieczność kolejnej rotacji (LR) przeprowadzonej w korzeniu. Po usunięciu elementu nastąpiła więc konieczność wykonania dwóch rotacji. W pesymistycznym przypadku może wystąpić seria rotacji na kolejnych poziomach drzewa. Taka kaskada rotacji możliwa jest tylko po usunięciu elementu (po dodaniu potrzeba najwyżej jednej). Jednak z racji tego , że jedna rotacja ma koszt stały, a wysokość drzewa AVL jest logarytmiczna, to nawet seria rotacji nie zmienia logarytmicznej złożoności operacji usuwania.