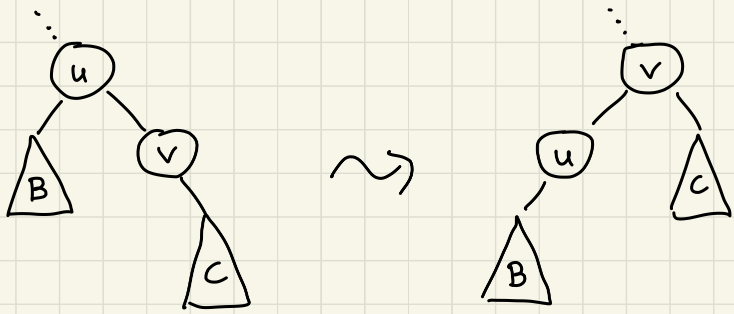
7.2. c)

* dane drzewo T zawierające w węzłach pary liczb (k_i, p_i), gdzie k_i jest kluczem, a p_i - priorytetem
* zakładamy, że T jest BST drzewem ze względu na klucze
* szukamy ciągu rotacji pozwalającego na przekształcenie drzewa T w listę T_{path} (każdy węzeł wewnętrzny ma tylko prawego syna), a następnie listy T_{path} w drzewo T_{max}, które jest nie tylko drzewem BST ze względu na klucze, ale jeszcze kopcem typu MAX ze względu na priorytety
* zaczynamy od przejścia z T do T_{path}
* niech s_r oznacza maksymalnie prawą ścieżkę w drzewie T, a l_r - jej długość

* dopóki któryś węzeł na ścieżce s_r ma lewego syna, dodajemy go do ścieżki przy użyciu prawej rotacji

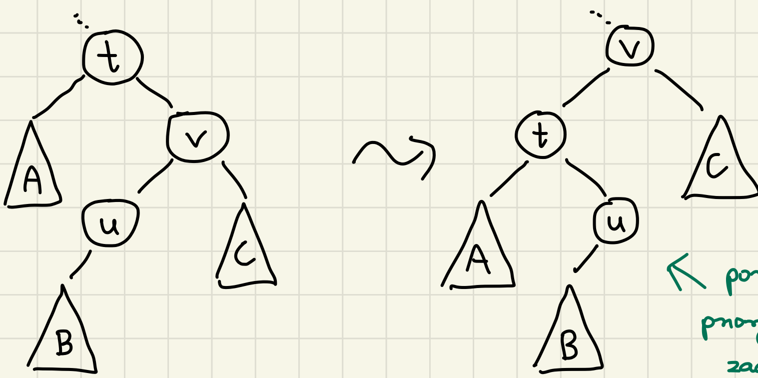
* po rotacji aktualizujemy s_r i l_r
* wiemy, że l_r zwiększy się o 1, zatem przejście z T do Tpath będzie wymagało co najwyżej n-1 rotacji (na początku ścieżka zawiera co najmniej jeden wierzchołek, którym jest korzeń drzewa)
* teraz pokażemy, że przejście z Tpath do Tmax także będzie wymagało co najwyżej n-1 rotacji

* tym razem zaczynamy w korzeniu drzewa i dla kolejnych wierzchołków na ścieżce wykonujemy operację UpHeap (jeśli rodzic u obecnego wierzchołka v ma niższy priorytet niż v, to wykonujemy rotację w lewo wokół u; wtedy v stanie się synem ojca u, oznaczanego przez t, i powtarzamy proces sprawdzenia porządku dla t i v)



* co ważne, idąc od korzenia do liścia wiemy, że v w pierwszym kroku ma puste lewe poddrzewo, więc przy rotacji w lewo nie zaburzymy relacji na priorytetach w dotychczas uporządkowanej części

* w kolejnym kroku v będzie już miało niepuste lewe poddrzewo (zawierające u i jego lewych potomków), ale wiemy, że wszystkie wierzchołki, które w nim są, mają priorytety mniejsze od priorytetu t, czyli ewentualna rotacja nic nie popsuje



porządek na priorytetach jest zachowany

* dalej analiza idzie przez indukcję
* na koniec zauważamy, że przy każdej takiej rotacji długość ścieżki S_r zmniejsza się o 1, więc musimy ich wykonać co najwyżej $n-1$