Algorytmy i struktury danych - lista 5

27 kwietnia 2020

Zadanie 5: W celu ułatwienia dowodu będę koszystał z grafiki na drugiej stronie (przykład rotacji, zarówno lewostronnej, jak i prawostronnej). Przeprowadzę dowód dla rotacji prawostronnej (rotacja zaznaczona niebieską strzałką).
Poniżej znajduje się pseudokod algorytmu *inorder*. Wywołanie $inorder(root)$ sprawi, że wypisane zostaną wszystkie elementy drzewa.

```
inorder(x){
        if(x != null){
                inorder(x.left)
                print(x)
                inorder(x.right)
        }
}
```

Dla drzewa przed rotacją przejście *inorder* tzn. wywołanie ($inorder(Q)$) przebiega nastepująco:

1. Q ma lewego syna tzn. P, więc wywołujemy $inorder(P)$,
2. P ma lewego syna tzn. A, więc wywołujemy $inorder(A)$,
3. A nie ma lewego syna, więc wypisujemy "A",
4. A nie ma prawego syna, więc kończy się wykonanie $inorder(A)$, wracamy do kolejnego etapu $inorder(P)$,
5. wypisujemy "P",
6. P ma prawego syna tzn. B, więc wywołujemy $inorder(B)$,

7. B nie ma lewego syna, więc wypisujemy "B",
8. B nie ma prawego syna, więc kończy się wykonanie $inorder(B)$, wracamy do kolejnego etapu $inorder(P)$,
9. kończy się wykonanie $inorder(P)$, wracamy do kolejnego etapu $inorder(Q)$,
10. wypisujemy "Q",
11. Q ma prawego syna tzn. C, więc wywołujemy $inorder(C)$,
12. C nie ma lewego syna, więc wypisujemy "C",
13. C nie ma prawego syna, więc kończy się wykonanie $inorder(C)$, wracamy do kolejnego etapu $inorder(C)$,
14. kończy się wykonanie $inorder(Q)$, koniec procedury.

Ostatecznie wypisana kolejność to A, P, B, Q, C.
Podobne wykonanie $inorder(P)$ dla sytuacji po obrocie (prawa część rysunku):

1. P ma lewego syna tzn. A, więc wywołujemy $inorder(A)$,
2. A nie ma lewego syna, więc wypisujemy "A",
3. A nie ma prawego syna, więc kończy się wykonanie $inorder(A)$, wracamy do kolejnego etapu $inorder(P)$,
4. wypisujemy "P",
5. P ma prawego syna tzn. Q, więc wywołujemy $inorder(Q)$,
6. Q ma lewego syna tzn. B, więc wywołujemy $inorder(B)$,
7. B nie ma lewego syna, więc wypisujemy "B",
8. B nie ma prawego syna, więc kończy się wykonanie $inorder(B)$, wracamy do kolejnego etapu $inorder(Q)$,
9. wypisujemy "Q",
10. Q ma prawego syna tzn. C, więc wywołujemy $inorder(C)$,

11. C nie ma lewego syna, więc wypisujemy "C",

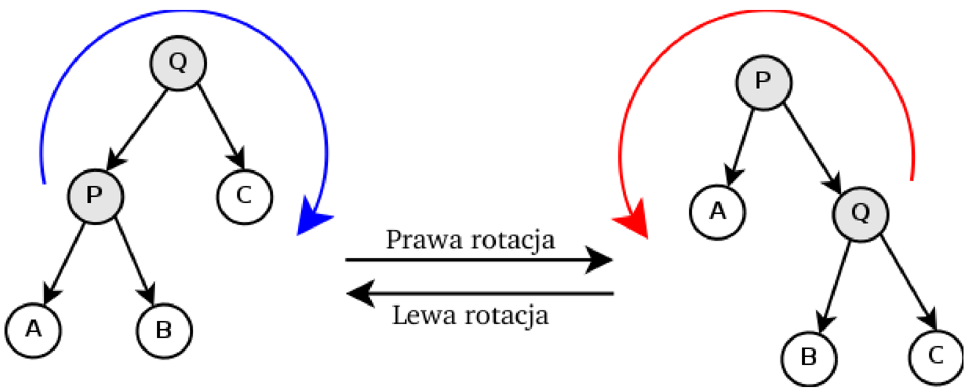
12. C nie ma prawego syna, więc kończy się wykonanie $inorder(C)$, wracamy do kolejnego etapu $inorder(Q)$,

13. kończy się wykonanie $inorder(Q)$, wracamy do kolejnego etapu $inorder(P)$,

14. kończy się wykonanie $inorder(P)$, koniec procedury.

Ostatecznie wypisana kolejność to A, P, B, Q, C. Jest ona identyczna, jak przed rotacją.
Jeśli A, B, C nie są liścmi, tylko drzewami, to rotacja również nie powoduje zmiany kolejności wypisywania wierzchołków (bo zarówno przed jak i po rotacji poddrzewa A, B, C są identyczne, więc dają identyczny rezultat wykonania funkcji $inorder$).
W sytuacji, gdy jeden z liści w założonym modelu nie istnieje (np. w sytuacji przed rotacją P ma tylko jednego syna - A), rotacja również nie zmienia kolejności wypisywania elementów - jest ona taka sama jak w rozważanej sytuacji, poza wykreśleniem brakującego liścia (lub elementów poddrzewa tego liścia, gdy np. B nie jest liściem, lecz drzewem).
Skoro sytuacja przed rotacją lewostronną jest identyczna jak sytuacja po rotacji prawostronnej i sytuacja po rotacji lewostronnej jest identyczna jak sytuacja przed rotacją prawostronną, a dodatkowo rotacja prawostronna zachowuje porządek kluczy $inorder$, to rotacja lewostronna również zachowuje porządek kluczy $inorder$.
Ostatecznie oznacza to, że zarówno lewostronna jak i prawostronna rotacja zachowują porządek kluczy $inorder$ w drzewie binarnym, czyli otrzymaliśmy tezę.

Warto tutaj zaznaczyć, że operacja rotacji jest operacją lokalną, więc zmianie ulegnie tylko ta część drzewa (tutaj: poddrzewo zakorzenione, odpowiednio, w Q lub P).
A zatem jeśli w tych poddrzewach porządek inorder kluczy jest zachowany, to będzie on zachowany także w całym drzewie.



Rysunek 1: Rotacja prawostronna i lewostronna