Lista 6

Zadanie 5

Treść

Rozważmy ukorzenione drzewo, w którego korzeniu pojawia się pewna informacja. W każdej rundzie, wierzchołek posiadający informację, może poinformować jedno swoje dziecko. Pokaż algorytm, który na podstawie struktury drzewa obliczy dla każdego wierzchołka w jakiej kolejności ma on informować dzieci tak, żeby czas dotarcia informacji do wszystkich wierzchołków drzewa był jak najkrótszy.

Rozwiązanie

Można sprawdzić, które poddrzewo jest głębsze lub ma więcej węzłów, jednak czasem nie daje to dobrych rezultatów (przykłady poniżej).

Zliczymy ile rund potrzeba do poinformowania całego poddrzewa, a na tej bazie zdecydujemy komu pierwszemu przekazać informacje.

Int calculate(node)

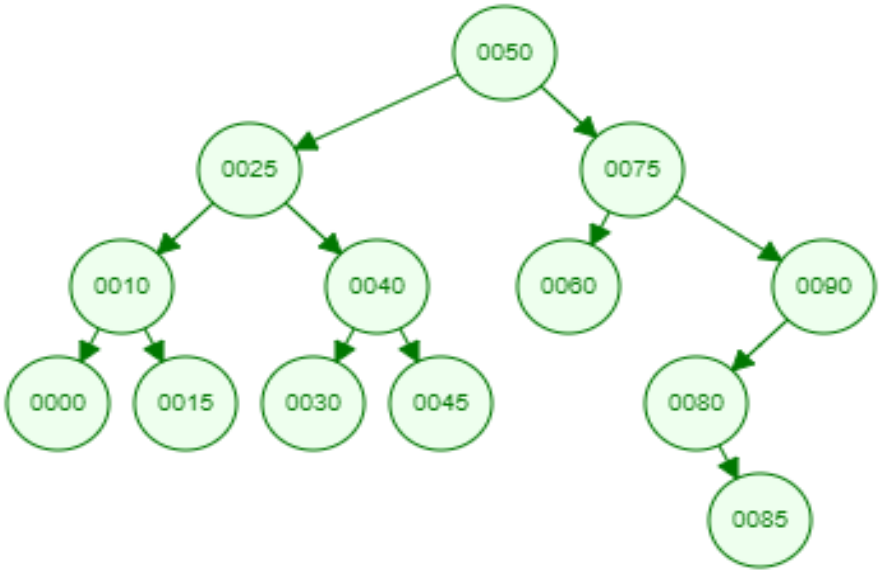
```
IF node.left == null && node.right == null
        return 0

L = 0
R = 0
IF node.left != null
        L = calculate(node.left) + 1
IF node.right != null
        R = calculate(node.right) + 1

IF L == R
        node.informLeft(true)  # obojętnie który
        return L + 1

IF R > L
        node.informLeft(false)
        return R
ELSE
        node.informLeft(true)
        return L
```

Liście zwracają 0.

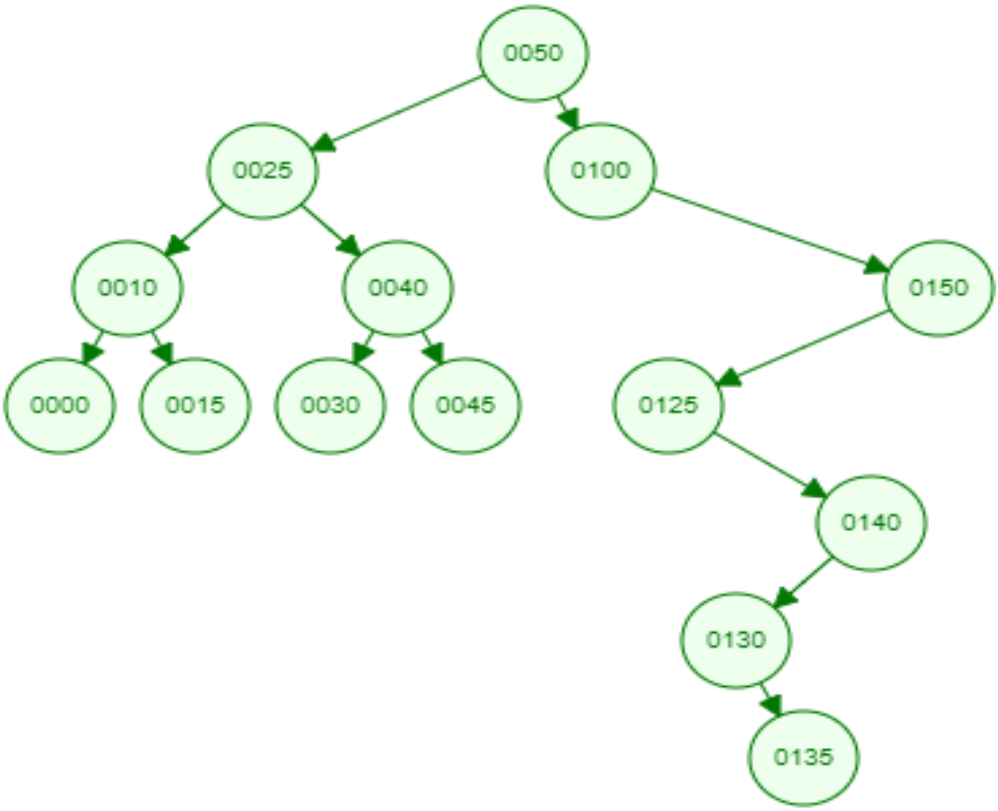
Do każdej zwróconej wartości doliczamy 1, symbolizuje to rundę potrzebną na przejście krawędzi do poddrzewa z którego została zwrócona wartość.

Jeśli wartości z obu poddrzew są równe nie jesteśmy w stanie zapełnić ich równomiernie, więc jedno z poddrzew będzie opóźnione o jedną rundę, dlatego dodajemy do zwracanej wartości 1.

Jeśli wartości nie są równe wybieramy jako pierwsze to poddrzewo, które potrzebuje więcej rund, nie ma w tym przypadku dodatkowych opóźnień, bo w najgorszym przypadku L = x, R = x – 1 decydujemy się na L, więc L i R zostaną zapełnione równo.



1. Lewe poddrzewo 5 rund, prawe 4



2. Lewe poddrzewo 7 elementów 5 rund, prawe 6 elementów 6 rund