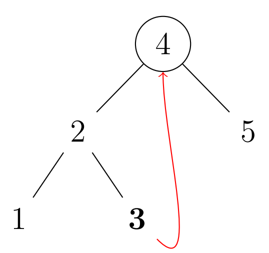
Algorytm:

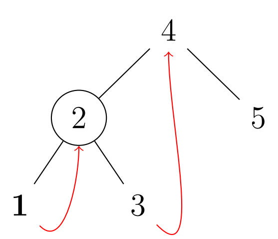
Zaprezentujemy algorytm przechodzący BST w kolejności in-order bez użycia rekurencji.

iterative_in_order (T):
  $current \leftarrow T$
  while $current$ is not null **do**
    if $current.left$ is null **then**
      print(current)
      $current \leftarrow current.right$
    else
      $predecessor \leftarrow current.left$
      while $predecessor.right$ is not null **and** $predecessor.right \neq current$ **do**
        $predecessor \leftarrow predecessor.right$
      end while
      if $predecessor.right$ is null **then**
        $predecessor.right \leftarrow current$
        $current \leftarrow current.left$
      else
        $predecessor.right \leftarrow$ null
        print($current$)
        $current \leftarrow current.right$
      end if
    end if
  end while

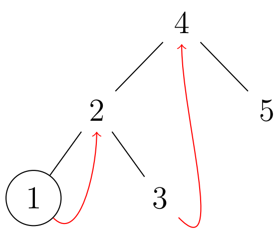
Przykładowy przebieg algorytmu:



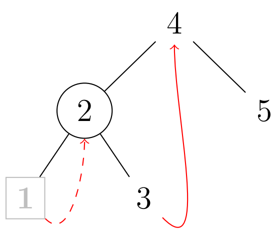
Rozpoczynamy w korzeniu drzewa. Lewe poddrzewo nie jest puste, więc znajdujemy poprzednika dla 4 – jest nim 3. Poprzednik nie ma prawego dziecka, więc tymczasowo czynimy nim nasz obecny węzeł.



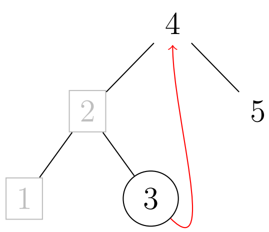
Przechodzimy do lewego dziecka – 2. Ponownie lewe poddrzewo jest niepuste, poprzednik to 1. Tymczasowo ustalamy 2 jako jego prawe dziecko.



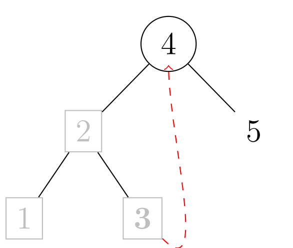
Znów przechodzimy do lewego dziecka – 1. Nie da się iść dalej w lewo, więc drukujemy wartość i przechodzimy w prawo – dzięki ustalonemu przez nas połączeniu wracamy do 2.



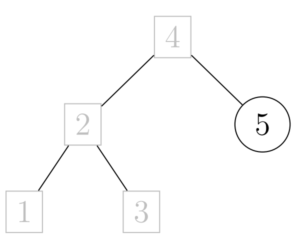
Poszukujemy poprzednika 2. Dodane przez nas połączenie sprawia, że wracamy do punktu wyjścia. Usuwamy je, drukujemy 2 i przechodzimy w prawo.



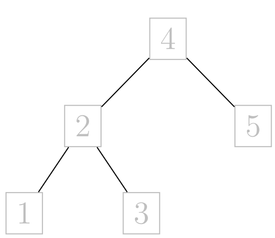
Dla 3 nie istnieje lewe poddrzewo, więc drukujemy ją i idziemy w prawo do 4 po dodanym wcześniej połączeniu.



Ponownie zapętlamy się w poszukiwaniach poprzednika, więc usuwamy tymczasowe połączenie, drukujemy 4 i przechodzimy w prawo.



Węzeł 5 nie ma lewego dziecka, więc drukujemy go i przechodzimy w prawo.



Ścieżka w prawo nie istnieje. Wydrukowaliśmy całe drzewo.

Złożoność:

Zauważmy, że każdy z węzłów odwiedzany jest najwyżej czterokrotnie – jak na przykład 2 w powyższym przykładzie:

1. Dwa razy jako wartość dla *predecessor* podczas szukania poprzednika dla rodzica – dla 2 kroki 1 i 6 w przykładzie
2. Jako *current* podczas pierwszego faktycznego przejścia do węzła (gdy tworzymy tymczasowe połączenie od poprzednika) – krok 2 w przykładzie
3. Jako *predecessor.right* przy powrocie z poprzednika przez tymczasowe połączenie (gdy je usuwamy) – krok 4 w przykładzie

Wnioskujemy więc, że złożoność algorytmu wynosi $O(4n) = O(n)$.

Poprawność:

Podamy szkic dowodu:

1. Jeśli $n = 1$, odwiedzamy korzeń. Lewe poddrzewo jest puste, więc drukujemy jego wartość i poruszamy się w prawo. Prawe poddrzewo również jest puste, więc kończymy algorytm. Drzewo zostało wydrukowane in-order. ✓
2. Dla $n > 1$ uogólnimy rozumowanie zaprezentowane w przykładzie. Spróbujemy zaagitować, że lewe poddrzewo rzeczywiście drukuje się w dobrej kolejności i że wracamy do korzenia. Zaczynając w korzeniu drzewa umieszczamy tymczasowe przejście do niego z jego poprzednika (elementu lewego poddrzewa położonego najbardziej na prawo). Następnie przechodzimy do kolejnych lewych poddrzew i powtarzamy dla nich tę procedurę. Zamysłem jest tutaj stworzenie drogi powrotnej w górę, której użyjemy, gdy skończą nam się elementy danego poddrzewa. Pierwszy raz skorzystamy z takiego połączenia w przypadku liścia położonego najbardziej na lewo – jest to najmniejszy element drzewa, który istotnie powinien być wydrukowany jako pierwszy. Wracamy do korzenia obecnie rozpatrywanego poddrzewa (w tym wypadku będącego rodzicem, ale nie jest to zasadą) mając świadomość o zatoczonej pętli, zatem drukujemy go i przechodzimy w prawo. Prawe poddrzewo drukujemy na tej samej zasadzie. Na jego końcu pojawia się znów dodane przez nas połączenie do poziomu wyżej. Przechodzimy zatem lewe poddrzewo w dobrej kolejności, wracamy do korzenia ze świadomością o użyciu tymczasowego połączenia, a następnie w analogiczny sposób przechodzimy prawe poddrzewo. Jeśli na jego końcu nie dodaliśmy żadnego połączenia, obecnie rozpatrywane drzewo nie jest niczyim poddrzewem, zatem kończymy pracę. ✓

Algorytm znany jest w literaturze jako **Morris Tree Traversal**.