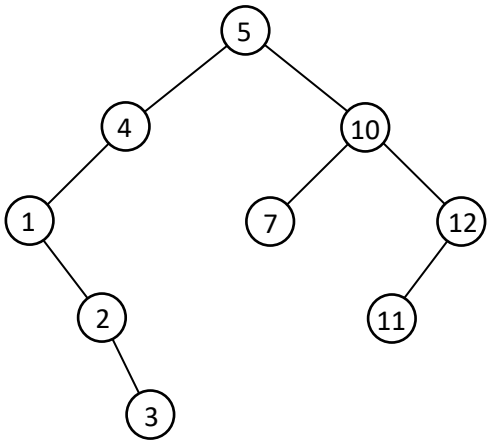
Cílem páté úlohy je naimplementovat základní rozhraní pro práci s binárním vyhledávacím stromem (ang. Binary Search Tree, zkráceně BST) [1]. BST je nelineární spojová struktura (formálně se jedná o souvislý acyklický graf), kde každý prvek (uzel) má nejvýše dva následníky. Uzly obsahují vzájemně porovnatelné klíče, které jsou ve stromě uspořádány tak, že hodnota potomka vlevo je menší (nanejvýš rovna) hodnotě rodičovského uzlu a hodnota potomka napravo je větší než hodnota rodičovského uzlu. Nejvyšší uzel, který nemá předka, se nazývá kořen stromu a uzel bez potomků je listem stromu. Pro účely snadnějšího ověření správnosti výsledků budeme uvažovat tento BST:



Následující body by vás měly postupně provést detaily implementace tohoto úkolu**. Podoba deklarace níže uvedených metod je závazná.**

Úloha 1: (0 bodů)

Vytvoření třídy Node.

Navrhněte vhodnou strukturu nové třídy `Node`, která bude reprezentovat uzel stromu. Musí obsahovat hodnotu klíče a dva potomky. V konstruktoru třídy inicializujte klíč předanou hodnotou a oba potomky nastavte na `None`. Vytvoření kořene našeho BST by pak mohlo vypadat takto:

```
def main():
  root = Node(5)
```

Úloha 2: (1 bod)

Přidání nového uzlu do stromu.

Rozšiřte třídu `Node` o rekurzivní (a nestatickou) metodu `insert`, která na správném místě v daném BST vytvoří nový listový uzel. Očekávaná podoba vložení dvou nových prvků do našeho stromu je následující:

```
def main():
  root = Node(5)
  root.insert(10)
  root.insert(12)
```

Úloha 3: (1 bod)

Zjištění hloubky stromu.

Rozšiřte třídu `Node` o rekurzivní (a nestatickou) metodu `depth`, která vrátí hloubku (výšku) stromu s tím, že hloubka kořene je 0. Výsledkem volání této metody nad kořenem stromu z ukázky je číslo 4:

```
def main():
  root = Node(5)
  for value in [10, 12, 11, 4, 7, 1, 2, 3]:
    root.insert(value)
  print(root.depth())
```

Úloha 4: (1 bod)

Uložení stromu do XML souboru.

Rozšiřte třídu `Node` o metodu `write_xml`, která zajistí uložení stromu od daného uzlu do zadaného XML souboru v odpovídajícím formátu. Pro práci s XML v jazyce Python využijte z modulu `xml.etree.ElementTree` třídy `ElementTree` a `Element`. Metoda write_xml může volat pomocnou privátní rekurzivní nestatickou metodu `__make_element(self, tag)`, která bude zajišťovat průchod stromu a vytváření instancí třídy Element korespondujících s levými (`tag="left"`) nebo pravými (`tag="right"`) uzly stromu.

```
def main():
  root = Node(5)
  for value in [10, 12, 11, 4, 7, 1, 2, 3]:
    root.insert(value)
  root.write_xml("bst.xml")
```

Očekáváné formátování a obsah XML souboru s uloženým stromem z ukázky je následující:

```
<root value="5">
  <left value="4">
    <left value="1">
      <right value="2">
        <right value="3" />
      </right>
    </left>
  </left>
  <right value="10">
    <left value="7" />
    <right value="12">
      <left value="11" />
    </right>
  </right>
</root>
```

Úloha 5: (1 bod)

Načtení stromu z XML souboru.

Rozšiřte třídu `Node` o statickou metodu `read_xml`, která zajistí načtení stromu ze zadaného XML souboru a vrátí jeho kořen. Metoda `read_xml` může volat pomocnou privátní rekurzivní statickou metodu `__parse_element(element)`, která bude zajišťovat průchod stromu a vytváření instancí třídy `Node` korespondujících s levými (`element.find("left")`) nebo pravými (`element.find("right")`) uzly stromu.

```
def main():
  root = Node.read_xml("bst.xml")
```

Úloha 6: (1 bod)

Průchod stromem.

Rozšiřte třídu `Node` o rekurzivní metodu `traverse_inorder`, která provede průchod našeho stromu do hloubky metodou inorder (projde levý podstrom, vrátí klíč, projde pravý podstrom) [2]. Metodu realizujte jako generátor (místo `return self.__value` použijte `yield self.__value`). Jelikož se ale jedná o rekurzivní metodu, musíte při vracení návratové hodnoty z jejího rekurzivního volání využít místo jednoduchého `yield` variantu `yield from` z PEP 380 [3]. Výsledek správnosti implementace průchodu stromem ověříte jednoduše, jelikož inorder průchod BST vrací prvky jako neklesající posloupnost.

```
def main():
  root = Node.read_xml("bst.xml")
  sorted_values = []
  traversal = root.traverse_inorder()
  while True:
    try:
      sorted_values.append(next(traversal))
    except StopIteration:
      break
  print(sorted_values)
```

Uvedený kód vytiskne pro náš uvažovaný strom následující seznam:

```
['1', '2', '3', '4', '5', '7', '10', '11', '12']
```

Reference

[1] https://cs.wikipedia.org/wiki/Bin%C3%A1rn%C3%AD_vyhled%C3%A1vac%C3%AD_strom
[2] https://cs.wikipedia.org/wiki/Strom_(datov%C3%A1_struktura)#Proch%C3%A1zen%C3%AD_stromem_do_hloubky
[3] https://www.python.org/dev/peps/pep-0380/