Představení úlohy – Rodokmen

V této úloze budeme pracovat se stromovou strukturou představující jednoduchý rodokmen, který má jednu výchozí osobu, její potomky, potomky jejích potomků atd. (V českém genealogickém názvosloví se tento strom správně nazývá „rozrod“.)

Část 1 – Reprezentace a vytvoření rodokmenu (2 body)

Do připravené třídy `Person`, která reprezentuje osobu v rodokmenu, doplňte inicializační metodu `__init__`. Objekty této třídy musí mít následující atributy:

- `pid`: identifikační číslo osoby – toto číslo je pro každou osobu unikátní, tj. nebudou existovat dvě osoby se stejným `pid`;
- `name`: jméno osoby (řetězec);
- `birth_year`: rok narození osoby;
- `parent`: rodič osoby nebo `None`, pokud je aktuální osoba výchozí osobou rodokmenu;
- `children`: seznam přímých potomků (dětí) osoby.

Přesná podoba inicializační metody (počet a typy parametrů) je na vás. Kromě uvedených atributů si můžete přidat libovolné vlastní.

Dále implementuje čistou funkci

```
def build_family_tree(names: Dict[int, str],
                      children: Dict[int, List[int]],
                      birth_years: Dict[int, int]) -> Optional[Person]
```

která vytvoří rodokmen podle zadaných údajů. Parametry této funkce jsou:

- `names` je slovník, který `pid` osoby přiřazuje její jméno.
- `children` je slovník, který `pid` osoby přiřazuje seznam `pid` jejích přímých potomků. Pokud nějaká osoba nemá položku v tomto slovníku, znamená to, že nemá žádné potomky. Smíte předpokládat, že tento slovník neobsahuje žádné cyklické vztahy (tj. že by někdo byl svým vlastním přímým nebo nepřímým potomkem).
- `birth_years` je slovník, který `pid` osoby přiřazuje její rok narození. Smíte přitom předpokládat, že hodnoty slovníku jsou vždy kladná celá čísla.

Řešení za 1 bod smí dále předpokládat, že vstup splňuje následující podmínky:

- slovník `names` obsahuje jako klíč každé `pid`, které se vyskytuje ve slovnících `birth_years` nebo `children` *(ve slovníku `birth_years` jsou `pid` jen jako klíče, ve slovníku `children` jako klíče i prvky seznamů)*;
- slovník `birth_years` obsahuje jako klíč každé `pid`, které se vyskytuje jako klíč ve slovníku `names`;
- existuje vždy právě jedna *výchozí osoba* (osoba, která nemá žádného rodiče);
- žádná osoba není přímým potomkem dvou různých osob v rodokmenu;

a vrátí výchozí osobu rodokmenu. Řešení za 2 body navíc v případech porušení jedné z těchto čtyř podmínek vrátí `None`.

Nezapomeňte při vytváření rodokmenu vždy správně nastavit všechny požadované atributy objektů typu `Person` (zejména `parent` a `children`). Pořadí objektů v atributu `children` musí odpovídat pořadí v odpovídajícím seznamu ze vstupního slovníku `children`.

Ve všech dalších částech smíte předpokládat, že objekty typu `Person` jsou součástí rodokmenu, který vznikl funkcí `build family tree`.

Část 2 – Validace rodokmenu (1 bod)

Implementujte následující metodu-predikát třídy `Person`:

```
    def is_valid(self) -> bool
```

Tato metoda zkontroluje, jestli rodokmen směrem „dolů“ od aktuální osoby (tedy směrem k potomkům), splňuje následující podmínky:

- jméno žádné osoby není prázdné;
- *každý rodič se narodil dříve než jeho děti;*
- žádní dva sourozenci nemají stejné jméno.

V případě, že jsou tyto podmínky splněny, vrací `True`, jinak vrací `False`.

Dále implementuje predikát (jako volnou funkci, tedy nikoli metodu)

```
def valid_family_tree(person: Person) -> bool
```

který zkontroluje, že výše zmíněné podmínky platí pro *celý rodokmen*, jehož součástí je zadaná osoba.

Ve všech dalších částech smíte předpokládat, že rodokmen splňuje predikát `valid family tree`.

Část 3 – Vykreslení rodokmenu (1 bod)

Implementuje následující metodu, která textově vykreslí rodokmen podle níže uvedeného vzoru:

```
    def draw(self, names_only: bool) -> None
```

Vykresluje se pouze směrem „dolů“ od aktuální osoby, tj. aktuální osoba, její potomci, potomci jejích potomků, atd.; ignorují se rodiče aktuální osoby. Metoda draw nemodifikuje aktuální objekt.

Příklad výstupu (funkci použitou k vytvoření tohoto rodokmenu najdete v kostře řešení) po zavolání `draw(False)` na výchozí osobě.

```
Qempa' (2256) [17]
├─ Thok Mak (2281) [127]
│  ├─ Ag'ax (2317) [611]
│  ├─ K'alaga (2302) [561]
│  └─ Samtoq (2317) [702]
└─ Worf (2290) [290]
   └─ Mogh (2310) [490]
      ├─ Worf (2340) [390]
      │  ├─ K'Dhan (2388) [898]
      │  └─ Alexander Rozhenko (2366) [1000]
      │     └─ D'Vak (2390) [253]
      └─ Kurn (2345) [590]
         ├─ Grehka (2359) [429]
         ├─ Elumen (2357) [106]
         └─ Ga'ga (2366) [101]
```

Příklad výstupu po zavolání `draw(True)` na osobě jménem `Mogh`:

```
Mogh
├─ Worf
│  ├─ K'Dhan
│  └─ Alexander Rozhenko
│     └─ D'Vak
└─ Kurn
   ├─ Grehka
   ├─ Elumen
   └─ Ga'ga
```

Všimněte si zejména toho, že

- k vykreslování používáme následující znaky ze sady Unicode: `└ ─ ├ │`;
- svislá čára (tvořená znaky `│ ├ └`) sahá vždy jen tak daleko, jak je třeba;
- každá další úroveň začíná přesně o tři znaky více vpravo než předchozí;
- má-li parametr `names_only` hodnotu `True`, vypisujeme pouze jména osob; v opačném případě vypisujeme jméno, následované rokem narození v kulatých závorkách a `pid` osoby v hranatých závorkách, vše oddělené jednou mezerou.

Část 4 – Zjišťování vlastností rodokmenu (3 body)

Implementuje následující čisté metody. Příklady výstupu najdete v kostře řešení.

- `parents_younger_than(self, age_limit: int) -> Set[int]` vrátí množinu `pid` všech osob, které měly dítě ve věku menším než `age_limit`.
- `parents_older_than(self, age_limit: int) -> Set[int]` vrátí množinu `pid` všech osob, které měly dítě ve věku větším než `age_limit`.
- `childless(self) -> Set[int]` vrátí množinu `pid` všech osob, které nemají žádné děti.
- `ancestors(self) -> List['Person']` vrátí seznam všech předků aktuální osoby v pořadí od nejstaršího.
- `order_of_succession(self, alive: Set[int]) -> Dict[int, int]` dostane na vstupu množinu `pid` žijících osob a vrátí slovník, který `pid` každé osoby přiřadí pořadí následnictví (podobně jako u královských rodů). Přitom uvažujeme tzv. absolutní primogenituru – nerozlišujeme pohlaví osob (ani nemáme jak) a přednost má vždy nejstarší potomek. Jsou-li dva potomci stejně staří (narozeni ve stejném roce), přednost má ten, který je dřív v seznamu `children`. (Podrobnější vysvětlení a příklad je níže.)

Všechny tyto metody, kromě metody `ancestors`, se zabývají vždy jen podstromem aktuální osoby, tj. aktuální osobou, jejími potomky, potomky jejích potomků atd.

Část 5 – Modifikace rodokmenu (1 bod)

Implementujte metodu `remove_extinct_branches(self, alive: Set[int]) -> None`, která dostane na vstupu množinu `pid` žijících osob a odstraní z rodokmenu všechny vyhynulé větve, tj. osoby, které již nežijí ani nemají žádné (ani nepřímé) žijící potomky. Přitom zpracováváme pouze podstrom aktuální osoby (`self`) a tuto osobu z rodokmenu neodstraňujeme.

Výše uvedený rodokmen se po zavolání `remove_extinct_branches({101, 106, 253, 429, 561, 611, 702, 898})` na výchozí osobě nijak nezmění.

Výše uvedený rodokmen po zavolání `remove_extinct_branches({101, 106, 253, 390, 898, 1000})` na výchozí osobě bude vypadat takto:

```
Qempa'
└─ Worf
   └─ Mogh
      ├─ Worf
      │  ├─ K'Dhan
      │  └─ Alexander Rozhenko
      │     └─ D'Vak
      └─ Kurn
         ├─ Elumen
         └─ Ga'ga
```

Výše uvedený rodokmen po zavolání `remove_extinct_branches({101, 106, 253, 390, 898, 1000})` na osobě jménem `Mogh` bude vypadat takto:

```
Qempa'
├─ Thok Mak
│  ├─ Ag'ax
│  ├─ K'alaga
│  └─ Samtoq
└─ Worf
   └─ Mogh
      ├─ Worf
      │  ├─ K'Dhan
      │  └─ Alexander Rozhenko
      │     └─ D'Vak
      └─ Kurn
         ├─ Elumen
         └─ Ga'ga
```