Představení úlohy – Číselné křížovky (Kakuro)

Číselné křížovky, známé pod japonským jménem カックロ (Kakuro), jsou obdobou klasických křížovek, kde do prázdný políček nezapisujeme písmena, ale čísla 1 až 9 tak, aby se v dané části žádné z nich neopakovalo a aby jejich součet v dané části odpovídal (číselné) legendě.

Zadání číselné křížovky vypadá například takto:

	11	8			7	16
16			11	4		
7				13 7		
	15	12 21				
12					4	6
13			6			
17				6		

Tato křížovka má jediné řešení, a to vypadá takto:

	11	8			7	16
16	9	7	11	4	1	3
7	2	1	4	13 7	4	9
	15	12 21	5	1	2	4
12	1	5	2	4	4	6
13	6	7	6	2	3	1

V této úloze si zkusíte napsat program, který bude schopen některé jednodušší číselné křížovky řešit pomocí techniky backtrackingu. (Na větší křížovky to stačit nebude, ale pokud si troufáte na implementaci lepšího algoritmu, máte šanci získat bonusové body.) Budeme uvažovat variantu, kdy v zadání mohou být některá čísla již předvyplněná.

Pro identifikaci políček křížovky budeme jako obvykle používat souřadnice ve tvaru (x, y), přičemž levý horní roh má souřadnice (0, 0) a směrem dolů a doprava se hodnota souřadnic zvyšuje.

V kostře jsou připraveny dvě třídy, `Clue` a `Kakuro`. Třída `Clue` reprezentuje jednu nápovědu z legendy. Má tyto atributy:

- `total` – požadovaný součet;
- `position` – pozice nápovědy ve formátu (x, y);
- `is_row` – `True`, pokud jde o řádkovou nápovědu, `False`, pokud jde o sloupcovou nápovědu;
- `length` – délka úseku, ke kterému nápověda přísluší.

Do třídy `Clue` můžete přidávat metody nebo atributy, ale zachovejte ty výše uvedené včetně jejich způsobu inicializace (tj. zejména neměňte hlavičku metody `__init__`).

Třída `Kakuro` reprezentuje stav číselné křížovky. Má tyto atributy:

- `width`, `height` – šířka a výška křížovky;
- `array` – 2D pole reprezentující obsah křížovky, přičemž
  - `-1` reprezentuje políčko, které se nevyplňuje (šedé, s nápovědou),
  - `0` reprezentuje prázdné políčko, které je třeba vyplnit, a
  - čísla `1` až `9` reprezentují vyplněné políčko;
- `clues` – seznam nápověd typu `Clue`.

Dále je v této třídě předpřipravená metoda `set`, která nastaví hodnotu políčka na zadaných souřadnicích. Do třídy `Kakuro` můžete přidávat metody nebo atributy, ale zachovejte ty výše uvedené včetně jejich způsobu inicializace. Rovněž smíte měnit chování metody `set`.

Z hlediska testů budeme s objekty této třídy zacházet pouze takto:

- výše uvedené atributy budeme pouze číst, nebudeme je přímo modifikovat;
- jediné změny budeme provádět pomocí metody `set`, přičemž tuto metodu budeme volat jen na souřadnicích, na který je políčko k vyplnění (ať už je vyplněné nebo nikoli) a pouze s hodnotami 0 až 9 včetně.

Hlavní částí tohoto úkolu je část poslední, v níž je cílem napsat funkci, která řeší zadanou číselnou křížovku. Smyslem ostatních částí je hlavně to, abyste se s řešeným problémem dobře seznámili; mají vás zejména přivést k užitečným myšlenkám. Některé z nich Vám také mohou pomoci při testování a debugování, jiné můžete přímo využít v řešení hlavní části. (Není to ovšem nutné a nijak to nevyžadujeme.)

Část 1 – Načítání zadání ze souboru (1 bod)

Implementujte funkci `load_kakuro(filename: str) -> Kakuro`, která ze zadaného textového souboru načte zadání číselné křížovky a vrátí její stav jako objekt typu `Kakuro`. Smíte předpokládat, že soubor existuje, je přístupný pro čtení a dodržuje níže uvedený formát. Na pořadí nápověd v atributu `clues` nezáleží. Tato funkce nemá žádné jiné vedlejší efekty (tj. zejména nic nevypisuje na standardní výstup).

Formát souboru s křížovkou:

Na každém řádku souboru je popis jednoho řádku křížovky, přičemž popisy jednotlivých políček jsou odděleny jednou nebo více mezerami. Popisy políček jsou následující:

- znak `.` (tečka) reprezentuje prázdné políčko k vyplnění,
- číslice `1` až `9` reprezentují již předvyplněné políčko,
- řetězec ve formátu `číslo\číslo`, v němž jedno nebo obě čísla mohou chybět, reprezentuje „šedé“ políčko s případnými nápovědami – první číslo je nápověda směrem dolů, druhé číslo je nápověda směrem doprava.

Formát souboru dodržuje normu POSIX, tedy každý řádek bude ukončen znakem konce řádku `'\n'`.

Příklad: Prázdná křížovka uvedená výše má tuto textovou reprezentaci

```
\   11\  8\     \   \   7\ 16\
\16   .   .   11\   \4   .   .
\7    .   .     .  7\13  .   .
\   15\ 21\12   .   .    .   .
```

```
\12   .   .     .   .   4\  6\
\13   .   .     \6  .    .   .
\17   .   .     \   \6   .   .
```

Část 2 – Zobrazení a uložení křížovky (1 bod)

Ve třídě `Kakuro` implementujte následující dvě metody:

- Metoda `show_board(self) -> None` vypíše stav křížovky na standardní výstup ve formátu podobném vstupnímu souboru z minulé části. Místo políček s nápovědou se však vypíše pouze znak `\` (zpětné lomítko) a jednotlivá políčka budou oddělena přesně jednou mezerou.
- Metoda `save(self, filename: str) -> None` uloží stav křížovky do zadaného souboru, a to ve formátu vstupního souboru z minulé části, včetně políček s nápovědou. Na počtu mezer mezi políčky nezáleží, ale musí být alespoň jedna. Mezery na začátku nebo na konci řádku se ignorují, stejně tak budeme ignorovat případný chybějící znak konce řádku na konci souboru.

Tyto metody nijak nemodifikují aktuální objekt a nemají jiné vedlejší efekty než ty popsané výše.

Část 3: Validace křížovky (1 bod)

Implementujte ve třídě `Kakuro` metodu-predikát `is_valid(self) -> bool`, který rozhodne, zda je aktuální situace křížovky validní v následujícím smyslu:

- žádný úsek řádku nebo sloupce neobsahuje vícekrát stejné vyplněné číslo a
- součet již vyplněných čísel v každém úseku není větší než hodnota příslušné nápovědy.

Poznámka: To, že je úsek validní dle těchto dvou bodů, ještě neznamená, že je možné jej skutečně doplnit nějakými čísly tak, aby výsledek splňoval požadavek nápovědy. Přesto ale v této části chceme, abyste kontrolovali jen zmíněné dvě podmínky, nic navíc.

Část 4: Všechny úseky splňující nápovědu (2 bod)

Implementuje čistou funkci `cells_from_empty(total: int, length: int) -> List[List[int]]`, která vrátí seznam všech úseků délky `length` vyplněných čísly 1 až 9 bez opakování, jejichž součet je právě `total`. Prvky v seznamu musejí být seřazené lexikograficky.

Příklad: Volání `cells_from_empty(13, 2)` vrátí seznam `[[4, 9], [5, 8], [6, 7], [7, 6], [8, 5], [9, 4]]`.

Dále pak implementujte čistou funkci `cells_from_partial(total: int, partial: List[int]) -> List[List[int]]`, která dostane částečně vyplněný úsek `partial` (políčka k vyplnění jsou označena číslem 0) a vrátí seznam všech doplnění tohoto úseku, která obsahují pouze čísla 1 až 9 bez opakování a mají součet `total`. Prvky v seznamu musejí být opět seřazené lexikograficky.

Příklad: Volání `cells_from_partial(12, [0, 0, 6, 0])` vrátí seznam `[[1, 2, 6, 3], [1, 3, 6, 2], [2, 1, 6, 3], [2, 3, 6, 1], [3, 1, 6, 2], [3, 2, 6, 1]]`.

Při řešení této části použijte techniku backtrackingu.

Část 5: Výběr nápovědy k řešení (1 bod)

Implementujte ve třídě `Kakuro` čistou metodu `pick_clue(self) -> Optional[Clue]`, která ze všech zatím nevyřešených nápověd v křížovce vybere tu, která má nejmenší počet prázdných políček v příslušném úseku. Pokud je takových nápověd více, vyberte tu, jejíž souřadnice (x, y) jsou lexikograficky nejdříve; *jsou-li tyto nápovědy dvě, vyberte z nich tu sloupcovou.*

Poznámka: Lexikografické uspořádání na dvojicích v Pythonu realizuje operátor `<`.

Část 6: Řešení číselné křížovky (2 body)

Implementujte ve třídě `Kakuro` metodu `solve(self) -> True`, která se pokusí najít řešení aktuální číselné křížovky. Pokud řešení existuje, vyplní všechna políčka k vyplnění v atributu `array` a vrátí `True`. Pokud existuje více než jedno řešení, vezme libovolné z nich. Pokud žádné řešení neexistuje, vrátí `False` a uvede aktuální objekt do stavu, v němž byl před zavoláním metody `solve`.

Nápověda: Využijte backtracking. Vybírejte nápovědu postupně pomocí přístupu z předchozí části. Dobře si rozmyslete, kdy už nemá smysl pokračovat. V testech budeme používat jen takové vstupy, které se touto cestou dají dostatečně rychle vyřešit.