Dokumentace k zápočtovému programu

Zadání

Program řeší pro zvolenou šachovou figurku problém proskákání šachovnice tak, aby každé pole bylo navštíveno právě jednou. Také umí pro zvolenou figurku najít všechna pole dosažitelná v n skocích.

Použité technologie

Python 3.9 a knihovna tkinter

Volba algoritmů

Sousedním políčkem je myšleno takové políčko, na které se z aktuálního lze dostat jedním pohybem.

Proskakování šachovnice

Pro tento problém je v programu použité prohledávání do hloubky společně s heuristikou popsanou H. C. Warnsdorffem. Ta spočívá v tom, že jsou v prohledávání do hloubky volena nejprve ta políčka, ze kterých se lze dostat na co nejmenší počet nenavštívených políček.

Před hledáním cesty šachovnicí se klasickým prohledáváním do šířky ověří, jestli je každé políčko z výchozí pozice dosažitelné. Pokud není, tím hledání cesty šachovnicí skončí, žádná neexistuje. Pokud je každé políčko dosažitelné, neznamená to nutně, že řešení existuje.

Před začátkem samotného algoritmu se pro každé políčko spočítá, z kolika jiných políček se do něj lze dostat. Tato *počítadla* se pak během hledání udržují a slouží k aplikaci Warnsdorffova pravidla.

Algoritmus je implementovaný pomocí rekurzivní funkce, která si předává aktuální pozici políčka a pořadí aktuálního políčka. Na začátku se zavolá na startovní políčko a pořadí 0.

Pokud pořadí aktuálního políčka odpovídá pořadí posledního políčka, hledání skončí a rekurzí se „vybublá“ nahoru. Jinak se aktuální políčko označí jako navštívené a *počítadlo* všech sousedních políček se sníží o jedna. Dál se seřadí tato políčka dle jejich *počítadla* od nejmenšího po největší. Prohledávání bude pak zkoušet vybírat následující políčka podle tohoto seřazení. Pro vybrané políčko se vždy zavolá rekurzivní funkce s parametry daného políčka a o jedna větším pořadím, než je pořadí aktuálního políčka. Pokud žádné z nich nepovede k úspěchu, musí se backtrackovat. Počítadlo všech sousedních políček se zvýší o jedna a aktuální políčko se označí jako nenavštívené.

Případné remízy při řazení políček dle *počítadla* se řeší pevným pořadím možných pohybů.

V aplikaci je navíc naimplementovaný časovač, který algoritmus zastaví po 10 sekundách. Jak již bylo zmíněno, řešení někdy nemusí existovat, nebo i s popsanou heuristikou by jeho nalezení trvalo dlouho.

Dosažitelná políčka v n skocích

Úloha je v programu vyřešena variantou prohledávání do šířky.

To je implementované pomocí fronty, ve které se ukládají uspořádané dvojice políčko a číslo, kolika skoky se na něj lze dostat.

K řešení problému by nestačilo každé políčko navštívit maximálně jednou, jako příklad se nabízí dva skoky běžného koně. Ten určitě může skončit na startovním políčku, ale na cestě ho navštíví dvakrát.

Nejde si tedy u každého políčka jednoduše pamatovat, že už bylo navštívené, a dál ho ignorovat. Můžeme si ale pamatovat, jestli bylo v aktuální „úrovni“ hledání do šířky (všechna políčka na které se jde dostat stejným počtem skoků) přidáno do fronty. Přidávat ho tam víckrát totiž nemá smysl.

Před začátkem algoritmu se do fronty zařadí uspořádaná dvojice startovní políčko a 0. Dál se do fronty ještě přidá speciální hodnota, která značí konec „úrovně“.

Pak začne cyklus, který vždy na začátku odebere nejstarší prvek z fronty. Cyklus skončí, až bude ve frontě pouze speciální hodnota.

Pokud se jedná o speciální hodnotu značící konec „úrovně“, každého políčko se označí jako nenavštívené a tato speciální hodnota se přidá na konec fronty.

Pokud jde o běžný prvek, zkontroluje se, jestli už není druhý prvek v uspořádané dvojici rovný zadanému počtu pohybů. Pokud je, lze se na zkoumané políčko dostat požadovaným počtem skoků. Pokud není, projdou se všechna sousední políčka aktuálního. Pro každé nenavštívené z nich se do fronty zařadí uspořádaná dvojice daného políčka a vzdálenosti o jedna větší, než je ta aktuální. Navíc se políčko nastaví jako navštívené.

Diskuse volby algoritmů

Remízy ve Warnsdorffově heuristice lze rozhodovat i jinak než dle pevného pořadí políček. Takové metody jsou ovšem komplikované a specifické pro jednotlivé figurky, proto jsem žádnou v aplikace neimplementoval.

U proskakování šachovnice jsem také uvažoval nad průběžným kontrolováním toho, jestli se vzhledem k aktuální cestě stále dají navštívit všechna políčka, nebo zda jsou nějaké části šachovnice již nedosažitelné. To ovšem stále rozhodně nezaručí rychlé nalezení řešení, naopak je ve většině případů zbytečné.

Struktura programu

Součástí programu jsou tři typy oken.

První je hlavní okno, které slouží k většině komunikace s uživatelem. Je na ní i samotná šachovnice a většina ovládacích prvků.

Druhým typem je okno nastavení, které slouží k změně velikosti šachovnice v hlavním okně (je pevně provázané s hlavním oknem).

Posledním je okno vyskakovací zprávy, které předchozím typům oken slouží k informování uživatele.

V hlavním okně se k vykreslování šachovnice i k algoritmickým výpočtům používá třída představující jedno políčko na šachovnici. Celá šachovnice je uložená jako dvourozměrný seznam těchto políček. Když chce program změnit, co je aktuálně nakresleno na šachovnici, projde všechna políčka a do referencí na nakreslené objekty (reference si drží každé políčko) uloží nová data.

Vstup a výstup

Jako první je potřeba nastavit velikost šachovnice. To jde udělat pomocí stisknutí tlačítka „Settings“. Ve vyskakovacím okně se pak nastaví velikost šachovnice a velikost jednoho políčka.

Poté je potřeba definovat šachovou figurku. Oranžově je zvýrazněná aktuální pozice figurky, červeně její možné pohyby. Levým kliknutím myši na políčko se na něj přesune figurka. Pravým kliknutím myši na políčko se přidá příslušný pohyb figurky. Pokud je zaškrtnuté políčko „Symmetry mode“ (výchozí a doporučený stav) přidají se zároveň i všechny pohyby, kdyby se figurka otočila a zrcadlila (například klasický kůň jde z figurky bez pohybů vytvořit jedním kliknutím). Zmáčknutím tlačítka „Clear moves“ jde smazat všechny doposud zadané pohyby.

Pokud uživatel definuje nesymetrickou figurku a následně zaškrtne políčko „Symmetry mode“, přidají se všechny pohyby tak, aby figurka byla symetrická.

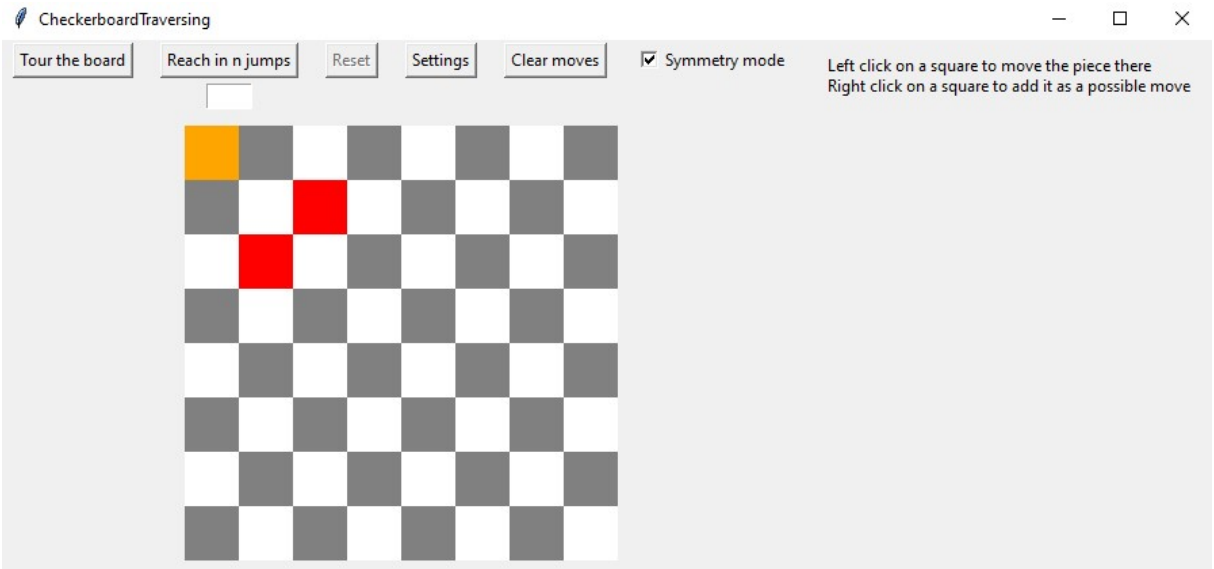
Pokud po definování figurky uživatel zmenší šachovnici a nějaké ze starých pohybů by na nové nebyly možné, tyto pohyby se trvale odstraní.

Dál zbývá vybrat jedenu ze dvou možných funkcí. Obě vyžadují startovací políčko. Za to se bere to políčko, na kterém aktuálně figurka stojí.

Zmáčknutím tlačítka „Tour the board“ se řeší úloha o projití celé šachovnice tak, že každé políčko se navštíví právě jednou. Hledání cesty může skončit jednou ze tří variant: 1. figurka nemůže navštívit celou šachovnici (řešení tedy rozhodně neexistuje), 2. řešení neexistuje nebo ho program neumí najít v krátkém čase, nebo 3. program vypíše nalezené řešení na šachovnici. Na každém políčku je v tom případě vypsané jeho pořadí v cestě (indexované od 0). Levým kliknutím na nějaké políčko jde zobrazit figurku a všechny její možné pohyby. Pravým kliknutím kamkoliv na šachovnici se figurka a její možné pohyby odstraní.

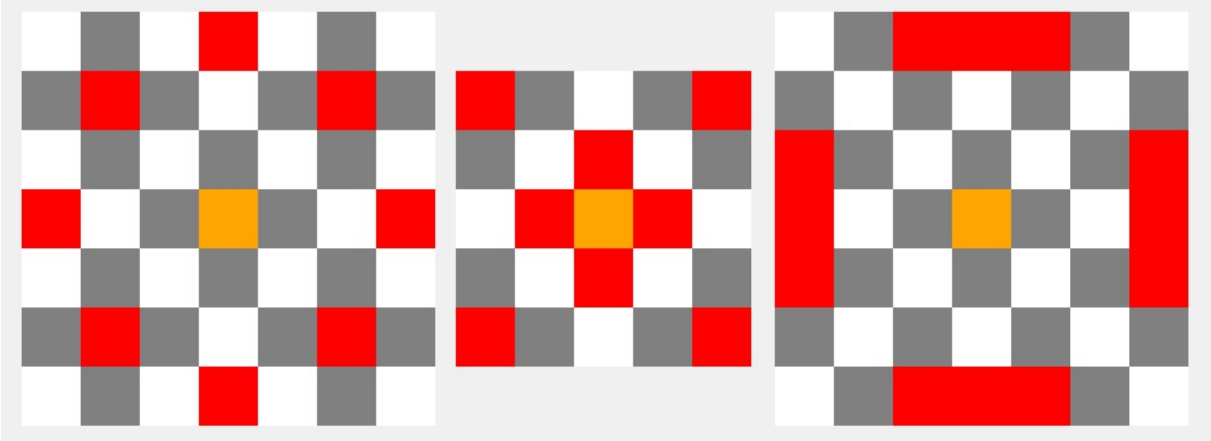
Před zmáčknutím tlačítka „Reach in n jumps“, které vykreslí všechna dosažitelná políčka ze startovní pozice v zadaném počtu pohybů, je potřeba do vedlejšího vstupního pole napsat cílený počet pohybů (mezi 0 a 100 včetně). Po zmáčknutí tlačítka se modře vybarví všechna políčka dosažitelná z výchozí pozice v zadaném počtu skoků.

Po prohlédnutí výsledků jedné z úloh se lze do výchozího stavu aplikace vrátit zmáčknutím tlačítka „Reset“.



Zajímavé vstupy

Tři zajímavé figurky (kromě běžného koně):



Závěr

Program správně využívá heuristiky k hledání cesty šachovnicí pro většinu možných figurek. Výjimkou jsou některé figurky s velmi málo pohyby nebo figurky s málo nebo žádnými pohyby jedním směrem.

Hledání všech dosažitelných políček funguje dobře pro libovolnou figurku.