Anastasiia-Solomiia Hrytsyna (ID: 105323) 27.10.2020

Eulerov kôň (3. problém, úloha f)

❖ Zadanie

Úlohou je prejsť šachovnicu legálnymi ťahmi šachového koňa tak, aby každé políčko šachovnice bolo prejdené (navštívené) práve raz. Riešenie treba navrhnúť tak, aby bolo možné problém riešiť pre štvorcové šachovnice rôznych veľkostí a aby cestu po šachovnici bolo možné začať na ľubovoľnom vychádzajúcom políčku.

❖ Opis riešenia

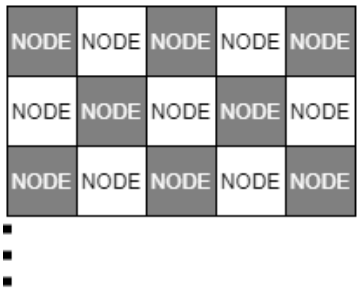
Implementáčne prostredie

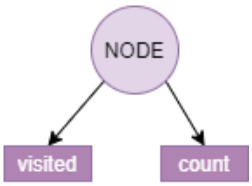
Moje riešenie som implementovala pomocou programovacieho jazyku C vo vývojovom prostredí Visual Studio 2019. Myslím si, že práve pomocou procedurálneho programovania (nie objektovo orientovaného) dá sa efektívnejšie a rýchlejšie vyriešiť danú úlohu.

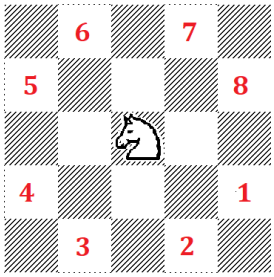
Návrh riešenia

Pri riešení svojej úlohy som použila algoritmus slepého prehľadávania (do hĺbky) (v kóde funkcia DFS), ktorý funguje rekurzivnym spôsobom. Zadanie riešim pre šachovnice veľkosťou 5×5 až 7×7, lebo pre väčšie šachovnice to už nebude tak efektívne.

Mojim stavom bola šachovnica (table, 2D array uzlov - víď obrázok) na ktorej sa pohyboval kôň. Každý uzol (structure node) obsahoval poradové číslo kroku koná a svoj stav (teda či bol navštevovaný alebo nie).



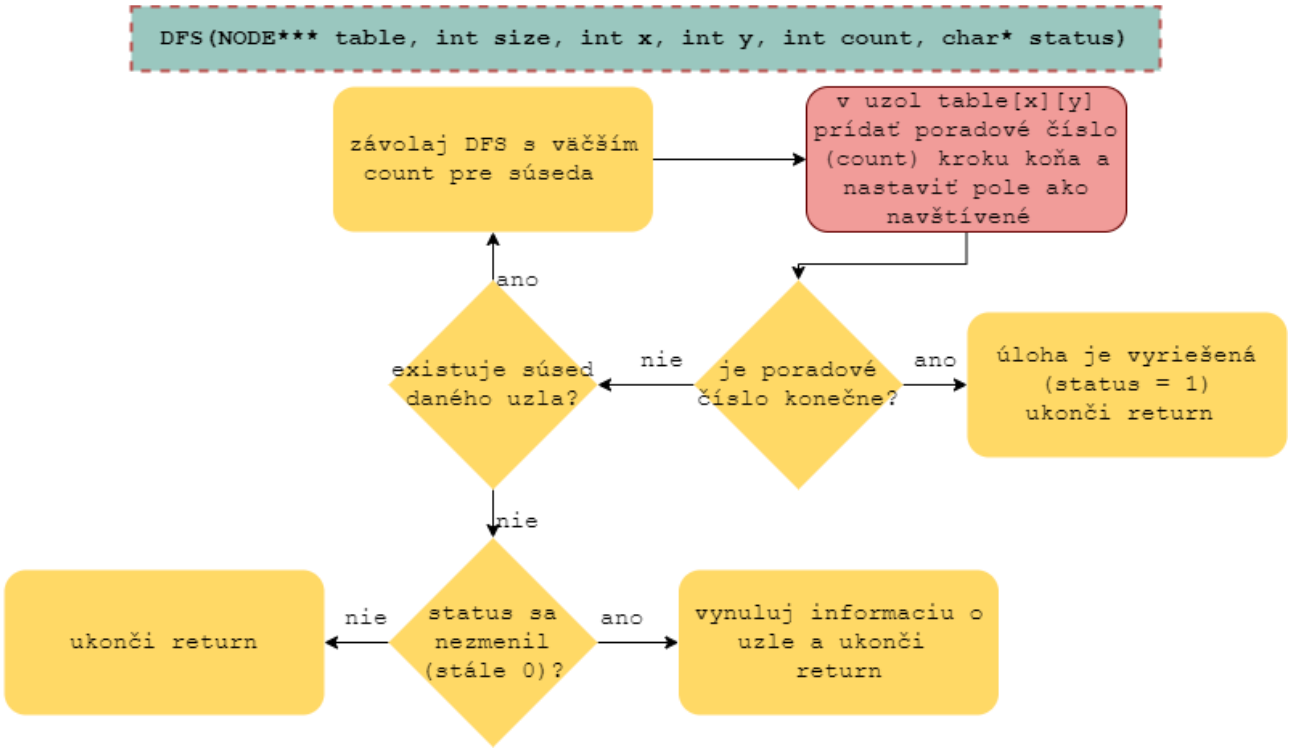




Najväčší počet políčok (operátorov) na ktorý môže stúpať kôň je 8 (ale môže byť aj menej, napríklad ak kôň je na okraji šachovnice). Na obrázku ja som zobrazila poradie navštevovania políčok v svojom programe.

Prehľadávanie do hĺbky funguje tak, že z daného uzlu, v ktorom sa nachádzame, skúsime nájsť políčko, na ktoré môže stúpať kôň. Ak ho našlo, tak prejde na to políčko a zopakuje postup predtým (rekurzia). Ak nie - tak vráti sa na pole naspäť a pôjde do ďalšieho v poradí možného nenavštíveného políčka. Keď už všetky možné varianty boli prejdene a stále to nenašlo riešenia, tak program sa skončí (a status bude sa rovnať 0). Na šachovniciach nepárnej veľkosti (5×5 či 7×7) riešenie neexistuje ak kôň štartuje z bieleho políčka (čiže kde súčet súradníc je tiež neparný). Taktiež sa program môže skončiť keď prekročí časový (status = 2) alebo krokovy limit (status = 3). V mojom programe oni sú nastavené na 15 sekúnd a 20 miliónov krokov.

Teda **hlavný princíp fungovania** mojho algoritmu je takýto:



Používateľské rozhranie

Výsledky svojho riešenia ja vypisujem do consoly. Taktiež tam je počet urobených krokov pri hľadaní správneho riešenia a čas, koľko to trvalo. Najviac, v prípade, že riešenie nebolo nájdené, program vypíše chybovú správu a jej dôvod (či riešenie neexistuje, či funkcia prekročila nejaký výpočtový limit).

❖ Spôsob testovania

Svoje riešenie ja testujem tak, že pre každý typ šachovnice určitej veľkosti (5×5, 6×6 alebo 7×7) ja generujem 5 štartujúcich políčok (prvý štart je vždy v ľavom dolnom rohu, iné štyri sú randomne). Potom spúšťam DFS a printujem výsledok, v prípade, že on bol nájdený algoritmom. Napríklad jeden z testov vyzerá tak:

```
************************************************************
Choose the number of the test that you want to perform:
(5) - size 5x5  (6) - size 6x6  (7) - size 7x7  (0) - exit
5

Movement of the Euler's horse:
 25   6  15  12  23
 16  11  24   7  14
  5   2  13  22  19
 10  17  20   3   8
  1   4   9  18  21

Total steps: 14009              Total time: 0.0030

The programm can't find any solutions for starting point (1;0).

The programm can't find any solutions for starting point (3;0).

Movement of the Euler's horse:
 25   4  15  10  23
 14   9  24   5  16
  1  18   3  22  11
  8  13  20  17   6
 19   2   7  12  21

Total steps: 365421             Total time: 0.0580

The programm can't find any solutions for starting point (3;0).
```

Z testov dá sa vypočítať aj priemernú rýchlosť skúmania uzlov. Teda približne je to 10 tisíc za 0.002 sek.

❖ Zhodnotenie

Toto riešenie je podľa môjho názoru úspešne a dosť rýchle. Môj algoritmus dokáže vyriešiť aj väčšie šachovnice bez hociakých problémov. Ja si myslím, že hlavné je to kvôli tomu, že neriešim úlohu cez objektovo-orientovane jazyky a nevytváram objekty, neukladám zbytočnej informácie a vždy uvoľňujem pamäť.

Pri riešení veľkých šachovnic (7×7) sa častejšie (v porovnaní s malými) stáva prekročenie krokoveho limitu, nie časového, takže kľudne môžte si dať aj väčší limit (ak RAM pamäť to dokáže zvládnuť) a snáď to zbehne úspešne. Preto podľa mňa na malých šachovniciach DFS je dobrým algoritmom na hľadania riešenia (lepšie ako BFS napríklad), ale pre zložitejšie situácie a väčšie šachovnice ako moje on už nebude vhodný, lebo čas riešenia problému raste dosť rýchlo (expotencialne).

Inak možným vylepšením by asi bolo riešiť úlohu nie rekurzivnym spôsobom, a iterativným (vtedy by to malo zabehnúť rýchlejšie) a ešte by sa zdalo pridať aj nejake GUI.