# Optimální rozmístění královen na šachovnici

Úvod do programování

David Martínek

Geografie a kartografie, 3. ročník

13.02.2022

## Zadání úlohy

Na šachovnici rozmístěte 8 královen tak, aby se vzájemně neohrožovaly. Vygenerujte alespoň jedno možné řešení.

Součástí odevzdání bude:

- Dokumentace se zadáním (4–5 stran)
    - Rozbor problému
    - Existující algoritmy
    - Popis zvoleného algoritmu
    - Struktura programu
    - Problematická místa
    - Možná vylepšení
- Zdrojový kód aplikace
- Vstupy/výstupy

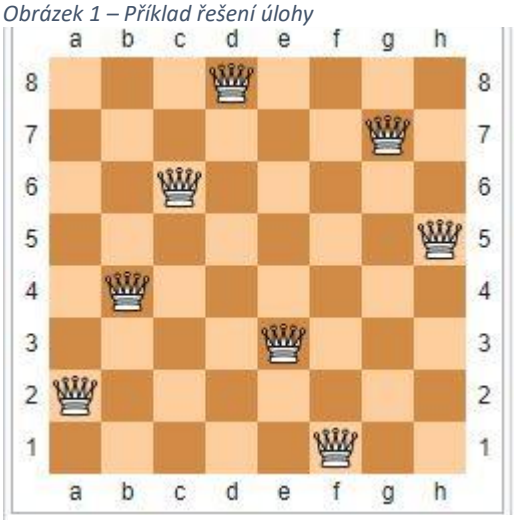
Za nefunkční bude aplikace považována, pokud:

- Při zpracování dat dojde k pádu
- Vrací špatné výsledky
- Neřeší možné singulární případy

## Rozbor problému

Úloha osmi královen je klasickým příkladem pro procvičování programátorských dovedností a logického uvažování. Tomuto problému se věnovali i významní matematici devatenáctého století (Wikipedia, 2022). Programátor se zde potýká s problémem, kde na šachovnici o 64 políčkách musí umístit osm královen, přičemž každá z nich musí zaujímat vlastní řádek, sloupec i obě diagonály.

*Obrázek 1 – Příklad řešení úlohy*




*Zdroj: Wikipedia (2022b)*

Podobě jako úlohy bludiště nebo jezdcova cesta i úloha osmi královen je vhodná pro porozumění rekurzi – backtrackingu. Principiálně tak navrhovaný program postupně volí z nabízených možností pro umístění královny a v případě, že další královnu již umístit nemůže, vrací se zpět a označuje tuto cestu za nevedoucí k řešení. Program takto hledá řešení až do doby, dokud neumístí všech osm královen podle zadaných podmínek a prezentuje se svým řešením, anebo zjišťuje, že pro zadanou úlohu řešení neexistuje. Tento program pracuje s šachovnicí 8x8 políček, přičemž v tomto zadání je známo pro umístění osmi královen celkem 92 možných řešení.

## Existující algoritmy

Tento problém a jeho řešení je poměrně rozšířený a bylo k němu vytvořeno nemalé množství studijních materiálů, informativních a instruktážních videí a programátorských návodů. Z nejčastějších variant řešení se uplatňuje právě metoda backtrackingu, použita i v tomto programu, nebo metoda hrubé síly (*Brute-force algoritm*), která hledá všechna možná řešení, ale i neřešení (freeCodeCamp, 2020).

Backtrackingové algoritmy se mezi sebou mohou mírně lišit, nicméně všechny vychází z principu samotné metody: Umístí královnu a následně zkouší další možnosti, dokud jim to podmínky dovolují. V případě neprůchodnosti jedné cesty se vrátí zpět a zkouší cestu jinou.

## Popis zvoleného algoritmu

Program vytvoří herní plochu (*board*) a pokouší se na něj umisťovat královny. Funkce *Placement* začíná první sloupcem (předem definuje, že každá královnu bude zaujímat právě jeden sloupec) a umisťuje královnu do levého horního rohu (pole *[1][1]*). Přesouvá se do druhého sloupce a zkoumá, do kterého řádku může královnu umístit. Zkusí první řádek a ptá se funkce *LineChecker*, jestli se na daném řádku, sloupci a k políčku přilehlým diagonálám již nějaká královna nenachází. Pokud ano, program zkouší jiné políčko ve sloupci. Takto program postupuje do té doby, dokud se na šachovnici nenachází osm královen.

### Pseudokód zvoleného algoritmu:

***funkce Placement(column)***

***pokud*** *column >= počtu královen*

*funkce končí – má splněno*

***cyklus*** *pro každý řádek sloupce*

***pokud LineChecker*** *nenachází královnu*

***zapiš*** *královnu do tohoto políčka*

***pokud*** *je to správně,* ***vrať se*** *a hledej další královnu*

***jinak*** *smaž tuto královnu*

***pokud*** *královna nejde umístit do žádného řádku,* ***vrať se*** *k předchozí královně a hledej pro ni nové místo*

***funkce LineChecker(row, column)***

***pokud*** *v tomto řádku najdeš královnu*

***nesouhlas*** *s volbou políčka*

***pokud*** *v diagonálách najdeš královnu*

***nesouhlas*** *s volbou políčka*

***jinak*** *povol zápis královny*

## Struktura programu

Program je tvořen 41 řádky včetně komentářů a mezer.

Program nepotřebuje uživatele k zadání žádného vstupu a pracuje pouze s již předdefinovanými parametry. Počet hledaných královen je 8. Program vytvoří seznam seznamů, které vykreslují šachovnicovou hrací plochu o velikosti 8x8.

Funkce Placement, pokud jí to funkce LineChecker dovolí, umisťuje (Q) nebo odstraňuje (-) z šachovnice královny tak, aby se navzájem neohrožovaly. Je-li korektně umístěna i osmá královna, program vykreslí právě jedno nalezené řešení.

## Problematická místa

Program sám o sobě najde vždy jedno a to samé řešení, které samo o sobě může vykreslovat jeho backtrackingový postup (zleva doprava, shora dolů).

Problémem bylo naprogramovat správný přechod k rekurzi *[if Placement(column+1) == True:*]. Dokud nebyl zápis správný, docházelo k zacyklení celého programu s vytvořením značné námahy na procesor i paměť počítače. Úloha samotná není složitá na uvědomění si toho, čeho a jakým způsobem chce programátor dosáhnout, ale předat tu informace správným způsobem programu se ukázalo být jako značná potíž.

## Možná vylepšení

Program by mohl hledat i další řešení, případně by mu mohly být nastavovány výchozí pozice některých královen, aby nevycházelo vždy právě jedno a to samé řešení. Zajímavou úlohou by rovněž mohla být situace, kdy jsou z hrací plochy vyplněné královnami královny odebírány, přičemž se program musí dobrat podobného výsledku jako v současném případě.

Program by mohl poskytnou estetičtější formou vizualizace hezčí vykreslení celé situace. Kromě grafického využití samotné šachovnicové plochy by mohl být zaznamenán backtrackingový postup programu (kupříkladu formou bílých a červených šipek atp.).

## Zdroje

freeCodeCamp (2020): freeCodeCamp, Brute Force Algoritms Explained, https://www.freecodecamp.org/news/brute-force-algorithms-explained/

Wikipedia (2022): Wikipedia The Free Encyclopedia, Eight queens puzzle, https://en.wikipedia.org/wiki/Eight_queens_puzzle