

PROBLÉM N SUDOKU CIFER

PAVEL STRÍŽ (CZ)

Abstrakt. V článku je představeno rozšíření klasického programátorského problému známého jako problém N dam. Autor ukáže program pro Python3 s rekurzí a zpětným vyhledáváním všech možností plus výsledky svých experimentů a možnosti dalšího bádání.

Klíčová slova. Sudoku, rekurze, zpětné vyhledávání, problém N dam.

N SUDOKU DIGITS PUZZLE

Abstract. The article introduces an extension to a classical programming problem known as the N queens puzzle. The author shows the program for Python3 with recursion and backtracking of all possibilities. He also mentions results of his experiments and open problems.

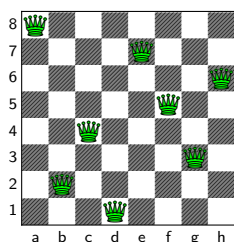
Keywords. Sudoku, recursion, backtracking, N queens puzzle.

1. Problém N dam

Problém, který v článku představím, je rozšířením problému N dam. Je to oblíbený problém soutěžních programátorů. Pamatuji si, že když jsem byl prcek, tak program pro QBasic jsem uměl nazpaměť, jak jsem si to neuměl dobře představit a naprogramovat, jak se sluší a patří. Po letech se tomu musím zasmát, kdy už tehdy jsem se nebál žádné výzvy, byť byla evidentně nad mé síly.

Nejběžnější varianta je pro $N = 8$ a definice je prostá. Úkolem je rozmístit na šachovnici právě osm dam, ale tak, aby se vzájemně nenapadaly: horizontálně (řádek), vertikálně (sloupec) ani v žádné diagonální linii. Zde je ukázka jednoho z 92 řešení. Vedle typograficky a šachově hezkého schématu zobrazuji textovou verzi se znakem Q pro dámu a hvězdičku pro prázdné políčko, a pak se znakem 1 reprezentující dámu a 0 reprezentující prázdné pole. Je to z důvodu, aby se čtenář rychleji dostal do nitra programu, kde se snažím o zobecnění. Z figur bílého přejdu na barvu zelenou, aby bylo zřejmé, že se nejedná už o čistě šachový problém.

K vysázení šachovnice je užít balíček `chessboard`, kde jsem poprvé zahlédl vrstvení znaků k dosažení efektu barevné šachovnice a figur, nyní běžný přístup přes `TikZ` a balíčky jako jsou `tikzducks`, `tikzpingus`, `tikzmarmots` či `tikzlings`.



```

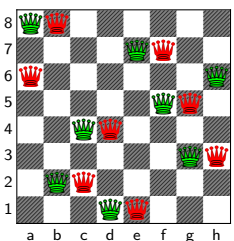
Q*****      10000000
****Q***      00001000
*****Q        00000001
*****Q**      00000100
**Q*****      00100000
*****Q*        00000010
*Q*****      01000000
***Q*****      00010000

```

První důležitý postřeh pro budoucí program je, že na volné políčko může útočit více dam. Proto při návratu zpět z rekurze (odebrání dámy) nesmíme zakázané políčko uvolnit jen s ohledem na odebíranou dámu. Proto neužijí typ True/False, ale počet, kolik dam na dané políčko útočí. Jen tehdy, když je počet nula, tak na dané políčko lze dámu umístit.

2. Problém N sudoku cifer

Poněvadž se zabývám sudoku a jejími variantami, napadlo mě, jestli by nebylo možné přidat další várku dam, řekněme dámy černého hráče. Ty mezi sebou také nesmí útočit, ale dámy černé s bílými si nezavazí. V terminologii sudoku cifry 1 a 2. Vedle barvy zelené užíjím barvu červenou, opět pro zdůraznění, že se dostáváme nad rámec hry v šachy.



```

Qq*****      12000000
****Qq**      00001200
q*****Q        20000001
*****Qq*        00000120
**Qq*****      00120000
*****Qq          00000012
*Qq*****      01200000
***Qq***          00012000

```

Otázka zní, zda-li je možné umístit všech 9 sudoku cifer. Pokud nikoliv, jaký je nejvyšší možný počet cifer. Z naší ukázky můžeme říci, že pro $N = 9$ určitě půjde umístit cifry 1–2, protože jsme našli alespoň jedno řešení. V dalších částech textu upouštím od sazby přes balíček `chessboard`, víc typů dam než dvě se tam neuzívá. Dá se to obejít různou barvou písma, víc o tom na přednášce.

3. Program

Naše úvaha by mohla být tato. Umístíme první cifru rekurzí. Vezmeme postupně výsledky a rekurzí se do volných polí pokusíme umístit další cifru. Abychom však mohli program zobecnit, uděláme si vektor `pole` s devíti 1, devíti 2 atd. Plus si přidáme informaci, na kterém řádku se cifra může pohybovat. Zakázané pole cifrou bude v mapě `reseni`, tam už se nesmí umístit nic dalšího. Dílčí zákazy pro danou cifru budou v mapě `zakazy` s indexem cifry. Jinými slovy

reseni je globální zákaz pro všechny třídy, proměnná **zakazy** pak lokální zákaz pro danou třídu.

Druhý postřeh je, že zákazy nemusíme dávat do řádku (tam se pohybuje cifra), ale ani do řádků nad, zákazy dáváme od zkoumané cifry níž.

Program zobecníme tím, že si N volíme. U kombinatorických úloh si častokrát volím $L = N - 1$ (pomůcka L pro less z angličtiny), abych se z indexování od jedné dostal na indexování od nuly. Zde podobně užijeme $M = N + 1$ (pomůcka M pro more z angličtiny), abychom si hlídali horní mez v Pythonu, funkce **range**, ta horní mez nebere včetně.

Ukázka je nastavena na prvním řádku pro $N = 8$ a první dvě cifry (dámy).

```
N=8; M=N+1; C=2+1 # číslo+1, nebo N-2+1, nebo M (N+1)
reseni=[[0 for k in range(N)] for l in range(N)]
citac=0; zakazy=[0]*M
for k in range(1,M):
    zakazy[k]=[[0 for k in range(N)] for l in range(N)]

pole=[]
for k in range(1,C): # M, nebo N-2 + 1
    for tradek in range(N):
        pole.append([k,tradek])
lenpole=len(pole)

def printreseni():
    for tradek in range(N):
        for tsloupec in range(N):
            print(reseni[tradek][tsloupec],end="")
        print()

def mal(pozice):
    global citac
    if pozice==lenpole:
        citac=citac+1; print("Řešení č.",citac)
        printreseni(); print()
        return
    zkoumane=pole[pozice]
    cifra=zkoumane[0]; radek=zkoumane[1]
    for sloupec in range(N):
        if reseni[radek][sloupec]==0: # pole je volné
            if zakazy[cifra][radek][sloupec]==0: # není zde cifrový zákaz
                reseni[radek][sloupec]=cifra # sem dám cifru
                # sloupec, řádky pod pozicí
                for R in range(radek+1,N):
                    zakazy[cifra][R][sloupec]+=1
                # doleva dolů
                usloupec=sloupec+1; uradek=N-radek; u=min(usloupec,uradek)
                for kroku in range(1,u):
                    Nradek=radek+kroku; Nsloupec=sloupec-kroku
                    zakazy[cifra][Nradek][Nsloupec]+=1
                # doprava dolů
                vsloupec=N-sloupec; vradek=N-radek; v=min(vsloupec,vradek)
                for krokv in range(1,v):
```

```

    Mradek=radek+krokv; Msloupec=sloupec+krokv
    zakazy[cifra][Mradek][Msloupec]+=1

    mal(pozice+1) # rekurze

    # vrátit vše zpět
    reseni[radek][sloupec]=0
    for R in range(radek+1,N):
        zakazy[cifra][R][sloupec]-=1
    for kroku in range(1,u):
        Mradek=radek+kroku; Nsloupec=sloupec-kroku
        zakazy[cifra][Mradek][Nsloupec]-=1
    for krokv in range(1,v):
        Mradek=radek+krokv; Msloupec=sloupec+krokv
        zakazy[cifra][Mradek][Msloupec]-=1

mal(0) # Spuštění rekurze

```

4. Získané výsledky

Pro $N = 1$ je jedno řešení, 1.

Pro $N = 2$ a $N = 3$ nemáme řešení.

Podobně pro $N = 4$, nejvyšší možný počet cifer je 2. Řešení jsou dvě pro jednu i dvě cifry.

0120	0210
2001	1002
1002	2001
0210	0120

Pro $N = 5$ dostáváme 240 řešení, první a poslední jsou následující. Zároveň vidíme, že jsou vertikálně překlopená jako pro $N = 4$.

12345	54321
45123	32154
23451	15432
51234	43215
34512	21543

Dle cifer: 10, 40, 120, 240 a 240.

Pro $N = 6$ nemáme řešení, nejvyšší počet cifer jsou 4. Dostáváme 24 řešení, zde jsou první dvě.

012340	012430
304102	403102
420031	320041
130024	140023
201403	201304
043210	034210

Dle cifer: 4, 12, 24, 0 a 0.

Pro $N = 7$ dostáváme 20160 řešení. Zde jsou první dvě.

1234567	1234576
6712345	7612345

4567123	4576123
2345671	2345761
7123456	6123457
5671234	5761234
3456712	3457612

Dle počtu cifer: 40, 496, 2400, 5280, 11520, 20160 a 20160.

Pro $N = 8$ nemáme řešení. Nejvyšší počet cifer je 6 a získáme 12960 řešení. V takovém případě dostáváme první dvě řešení tato.

12345600	12346500
45003126	46003125
06254031	05264031
30160254	30150264
60530412	50630412
03412065	03412056
21006543	21005643
54621300	64521300

Můžeme shrnout, že pro jednu cifru máme 92 řešení (problém osmi dam), pro dvě 3252, pro tři 37224, pro čtyři 109080, pro pět 44160, pro šest již zmíněných 12960, pro sedm a osm 0.

Pro $N = 9$. Tohle je jádro mých snah: získat podklady pro skutečné sudoku. Zjistil jsem následující. Můžeme shrnout, že pro jednu cifru máme 352 řešení (problém devíti dam), pro dvě 48328, pro tři 2315208, pro čtyři 33563424, pro pět až devět se mi to zatím nedopočítalo.

Pozn. Ve funkci `printreseni` jsem si upravil řádek na:

```
print(hex(reseni[tradek][tsloupec])[2:].upper(),end="")
```

abych místo 10 měl A, místo 11 B apod.

Pro $N = 10$. Ve výpočetním čase mne daném jsem nebyl schopen projít všechny možnosti a potvrdit, že to řešení nemá. Pro osm cifer by první dvě řešení byla tato. Zjistit počet řešení jsem se již nesnažil.

1203456078	1203456087
0512637840	0512638740
4730810625	4830710625
5687002134	5678002134
8304275061	7304285061
7065184302	8065174302
2840360517	2740360518
0156723480	0156823470
6071548203	6081547203
3428001756	3427001856

Pro $N = 11$ dostáváme řešení neuvěřitelně rychle. První řešení je tzv. žebřík, druhé také s prohozenými A a B. Zjistit počet řešení bylo nad mé výpočetní síly.

123456789AB	123456789BA
AB123456789	BA123456789
89AB1234567	89BA1234567
6789AB12345	6789BA12345
456789AB123	456789BA123

23456789AB1	23456789BA1
B123456789A	A123456789B
9AB12345678	9BA12345678
789AB123456	789BA123456
56789AB1234	56789BA1234
3456789AB12	3456789BA12

Zkusil jsem ještě $N = 12$ (počet cifer deset) a $N = 13$ (najít alespoň jedno řešení), ale bylo to již moc silné káfé.

5. Domněnka a otevřené problémy

Lze si ze získaných výsledků stanovit pracovní hypotézu:

- Pro $N \in \mathbb{N}$ lichá bez trojky: existuje řešení pro počet cifer 1 až N .
- Pro $N \in \mathbb{N}$ sudá bez dvojky: existuje řešení pro počet cifer 1 až $N - 2$.

Něco takového potvrdit či vyvrátit je nad mé matematické a výpočetní možnosti. Zdá se však, dle charakteru úlohy, že pro libovolné $N \geq 4$ je počet řešení stejný pro N a $N - 1$.

Bylo by zajímavé zkusit heuristiku místo rekurze, např. jsou-li pro větší N dostupná řešení typu žebřík.

Pro $N = 9$, má-li řešení, by bylo zajímavé zjistit, jestli je to platné řešení pro sudoku. Tedy po získání řešení vybrat jen ta, která navíc splňují podmínku sudoku bloků 3×3 .

Podobně jako u problému N dam by bylo zajímavé se zabývat počtem řešení se zahrnutím symetrie (rotace, překlápění).

Z pohledu programování: Navyšoval jsem počet cifer přes proměnnou C po jedné a zapisoval si počet řešení. Bylo by efektivnější zasáhnout do rekurze, dát $C=M$ a vypadla by tabulka s počty řešeními pro všechny možnosti 1 až N .

Za typografií by bylo příjemné si jednotlivé třídy vysázet samostatně automatizovaně, aby neútočení bylo lépe vidět, např. u prvního řešení pro $N = 5$:

1****	*2***	**3**	***4*	****5
1	***2*	****3	4****	*5***
****1	2****	*3***	**4**	***5*
*1***	**2**	***3*	****4	5****
1*	*2	3****	*4***	**5**

To je nad rámec tohoto článku a mého cíle představit tento problém. Žhavé numerické novinky od mých výpočetních strojů zmíním v Žilině. Na viděnou!

Kontaktní adresa

Ing. Pavel Stríž, Ph.D., U Škol 940, Bučovice, okres Vyškov, 685 01, Česká republika,

E-mailová adresa: pavel@striz.cz