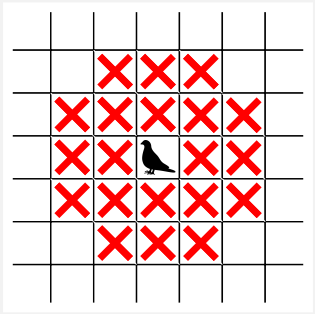
Riešenia 2. sady domácich úloh

Úloha 1

Figúrka *holub* ohrozuje políčka ako je znázornené na obrázku. Koľko najviac holubov možno umiestniť na šachovnicu rozmerov a) 6×6, b) 8×8 tak, aby sa žiadne dva neohrozovali? Vaše tvrdenie dokážte.

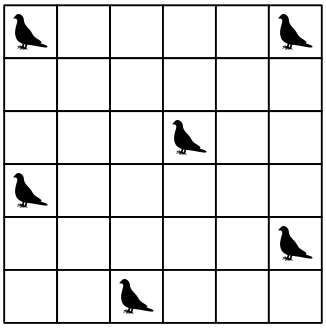
Pre získanie 1,5 boda stačí úplne vyriešiť časť a) a v časti b) korektne obmedziť možný výsledok na dve hodnoty.



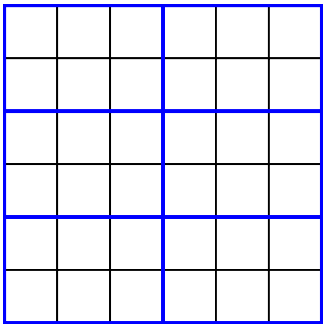
Šachovnica 6×6

Ukážeme, že najviac možno umiestniť 6 holubov.

Konštrukcia. Obrázok 1 znázorňuje umiestnenie 6 holubov, o ktorom sa ľahko presvedčíme, že sa žiadne dva neohrozujú.



Obr. 1



Obr. 2

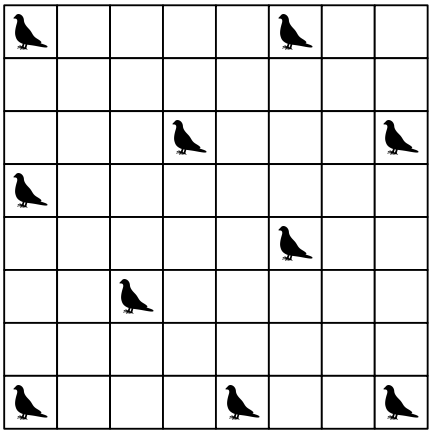
Odhad. Na obrázku 2 je šachovica rozdelená na 6 obdĺžnikov rozmeru 2×3. Ľahko skontrolujeme, že ľubovoľné dva holuby v obdĺžniku 2×3 sa ohrozujú. Ak by sme na šachovnicu umiestnili viac ako 6 holubov, tak v niektorom obdĺžniku by boli aspoň dva holuby, a tie by sa ohrozovali. Preto na šachovnicu nie je možné umiestniť viac ako 6 neohrozujúcich sa holubov.

Odhad (bez Dirichletovho princípu). Na obrázku 2 je šachovica rozdelená na 6 obdĺžnikov rozmeru 2×3. Každý z nich môže obsahovať najviac jedného holuba. Preto na šachovnicu možno umiestniť najviac 6 holubov.

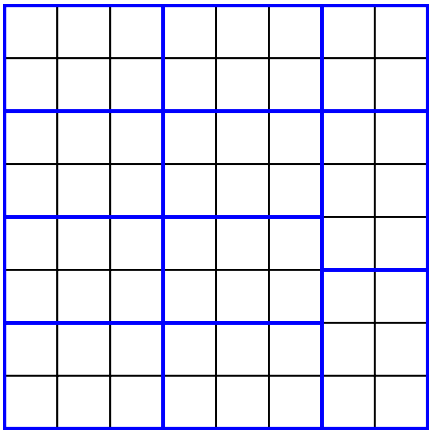
Šachovnica 8×8 (čiastočné riešenie)

Uvedieme najskôr čiastkové riešenie pre šachovnicu 8×8. toto riešenie sme od vás očakávali a stačilo na získanie 1,5 boda za úlohu (spolu s so správnou časťou a)).

Konštrukcia. Obrázok 3 znázorňuje umiestnenie 10 holubov, o ktorom sa ľahko presvedčíme, že sa žiadne dva neohrozujú.



Obr. 3



Obr. 4

Odhad. Na obrázku 4 je šachovica rozdelená na 11 oblastí. Ak by sme na šachovnicu umiestnili viac ako 11 holubov, tak v niektorej oblasti by boli aspoň dva holuby. Tieto dva holuby by sa ohrozovali, ako sme už overili pre obdĺžnik 2×3 v časti a), čo platí aj pre štvorec 2×2, ktorý je jeho podmnožinou. Preto na šachovnicu nie je možné umiestniť viac ako 11 neohrozujúcich sa holubov.

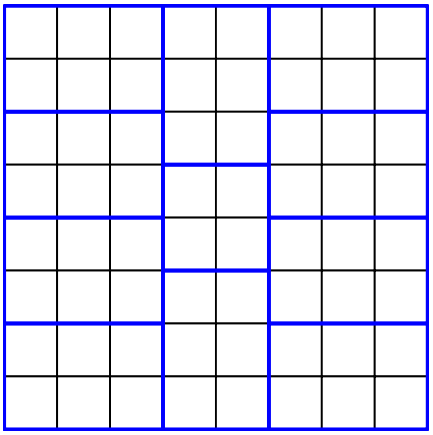
Spojením týchto dvoch úvah vieme, že hľadaný najväčší možný počet holubov je aspoň 10 a najviac 11. Teda možný výsledok sme zúžili na hodnoty 10 a 11.

Šachovnica 8×8 (úplné riešenie)

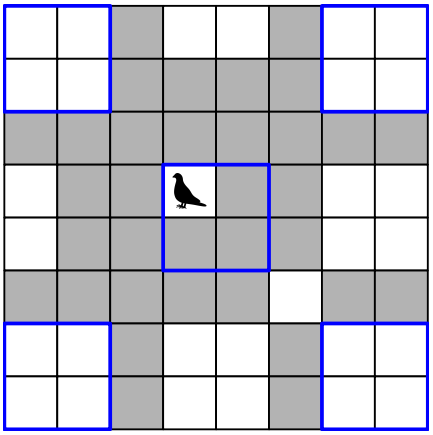
Ukážeme, že správnym riešením úlohy je 10. Umiestnenie 10 neohrozujúcich sa holubov sme už ukázali na obrázku 3. Ostáva nám ukázať, že 11 holubov nemožno umiestniť.

Správny odhad. Pre spor predpokladajme, že na šachovnici 8×8 máme umiestnených 11 holubov, z ktorých sa žiadne dva neohrozujú. Keďže v každej z 11 oblastí na obrázku 4 môže byť najviac jeden holub a máme práve 11 holubov, tak v každej oblasti musí byť práve jeden holub. Špeciálne, práve jeden holub musí byť v rohovom štvorci 2×2. Rovnaký záver platí pre každý rohový štvorec 2×2, keďže šachovnicou môžeme otáčať o $90°$. Takisto z rozdelenia šachovnice ako na obrázku 5 vyplýva, že práve jeden holub sa nachádza v prostrednom štvorci 2×2. Vďaka rotačnej symetrii môžeme bez ujmy na všeobecnosti predpokladať, že tento holub je v ľavom hornom rohu.

Na obrázku 6 si zaznačíme šedou políčka, ktoré tento prostredný holub ohrozuje. Takisto zaznačíme šedou pre každý z rohových štvorcov 2×2 štyri políčka, ktoré s nimi susedia – o nich totiž vieme, že sú ohrozené holubom v tomto rohovom štvorci bez ohľadu na to, na ktorom políčku sa presne nachádza. Ostalo nám tak 5 bielych oblastí (dva štvorce 2×2, dva obdĺžniky 1×2 a jeden štvorec 1×1). V každom z nich môže byť najviac jeden holub. Preto celkovo na šachovnici môže byť najviac 10 holubov. To je ale spor s predpokladom, že máme umiestnených 11 holubov.



Obr. 5



Obr. 6

Komentár. Upozorňujeme vás, aby ste sa neskúšali riešiť úlohy takýmto štýlom, pokiaľ mu isto nerozumiete. A ani takýto sofistikovaný prístup nebudeme od vás vyžadovať na skúškach. Riešiť úlohu uvažovaním kde musia byť holuby (resp. iné objekty) je náročné a veľa takých pokusov končí nesprávnymi úvahami za 0 bodov.

V čom je toto riešenie odlišné? V prvom rade, má jasný začiatok – hovoríme, že dokazujeme sporom, teda je jasné, z čoho vychádzame. Ťažiskovou úvahou je rozdelenie šachovnice na oblasti. Bez neho sa ťažko dopracujeme ku korektným úvahám o tom, že holub nikde musí byť.Tiež si všimnite, že na šachovnicu sme umiestnili len jedného holuba, a to len vďaka tomu, že všetky štyri možnosti jeho umiestnenia sú symetrické. Do rohových oblastí sme už holubov neumiestňovali. Totiž nevieme, na ktorých zo štyroch políčkach môže byť. Stačilo nám však vedieť, že niektoré políčka budú týmto holubom ohrozené, a to nech bude na hociktorom z týchto štyroch políčok.

Uvedený spôsob riešenia je teda akousi nadstavbou k Dirichletovmu princípu (resp. princípu delenia na oblasti), kedy vieme získaný odhad o trochu zlepšiť.

Úloha 2

Zistite, či pre ľubovoľné množiny A, B, C, D platí

a) $(A \times B) - (C \times D) \subseteq (A - C) \times (B - D)$,

b) $(A \times B) - (C \times D) \supseteq (A - C) \times (B - D)$.

Vaše tvrdenia zdôvodnite iba na základe definícií, teda bez odvolávania sa na známe tvrdenia o množinách alebo karteziánskom súčine.

Dôkaz, že a) neplatí

Nech $A = B = D = \{1\}$ a $C = \emptyset$. Potom

- $(A \times B) - (C \times D) = \{(1,1)\} - \emptyset = \{(1,1)\}$.
- $(A - C) \times (B - D) = \{1\} \times \emptyset = \emptyset$.

Avšak, vtedy $\{(1,1)\} \not\subseteq \emptyset$.

Dôkaz, že b) platí

Keďže ľavá strana obsahuje len usporiadané dvojice, tak nám stačí ukázať, že pre každú usporiadanú dvojicu (x, y) platí:

$$\begin{gathered}(x,y) \in (A - C) \times (B - D) \\ \Downarrow \\ x \in A - C \wedge y \in B - D \\ \Downarrow \\ x \in A \wedge x \notin C \wedge y \in B \wedge y \notin D \\ \Downarrow \\ (x \in A \wedge y \in B) \wedge (x \notin C \wedge y \notin D) \\ \Downarrow (*) \\ (x,y) \in A \times B \wedge (x,y) \notin C \times D \\ \Downarrow \\ (x,y) \in (A \times B) - (C \times D)\end{gathered}$$

Komentár. Pozor, implikácia (*) je jediná, ktorá nie je ekvivalenciou – a taká tam musí byť, keďže inak by šlo o dôkaz nepravdivého tvrdenia a), ktorý nemôže existovať. Správna ekvivalencia je

$$(x \notin C \vee y \notin D) \Leftrightarrow (x,y) \notin C \times D.$$

(Odporúčame si znegovať $(x, y) \in C \times D$, teda $x \in C \wedge y \in D$.) Avšak my nepotrebujeme ekvivalenciu. Zdôvodnenie implikácie zhora nadol je jednoduché: keď máme dvojicu (x, y), z ktorej $x \notin C$ a tiež $y \notin D$, tak vieme, že sa nenachádza v karteziánskom súčine. Ak by sme to chceli dokázať striktne z definície, tak môžeme takto:

$$(x \notin C \wedge y \notin D) \Rightarrow x \notin C \Rightarrow (x \notin C \vee y \notin D) \Leftrightarrow (x,y) \notin C \times D.$$

Úloha 3

a) *(0,5 b)* Koľko je všetkých 3-ciferných čísel s ciferným súčtom 21, ktoré majú všetky cifry rôzne?

b) *(0,5 b)* Koľkými spôsobmi možno vyplniť tabuľku 3×3 číslami 0 a 1 tak, aby v každom štvorci 2×2 bol rovnaký počet jednotiek ako núl?

c) *(1 b)* Koľkými spôsobmi možno vyplniť tabuľku 3×7 (riadky $\times$ stĺpce) číslami 0, 1, 2 tak, aby každý riadok obsahoval práve 4-krát číslo 0?

Správnosť vašich riešení zdôvodnite (stačí neformálne). K podúlohám a), b) **vypíšte všetky možnosti**. Aj v týchto podúlohách treba riešenie odôvodniť – musíte vysvetliť, prečo ste každú možnosť vypísali práve raz.

Podúloha a)

Najskôr určíme, z akých navzájom rôznych cifier sa môže skladať 3-ciferné číslo so súčtom 21:

- Ak je najväčšia cifra 9, tak zvyšné dve majú súčet $21 - 9 = 12$. Keď postupne skúšame druhú cifru od najväčšej, tak dostaneme možnosti $8 + 4$ a $7 + 5$.
- Ak je najväčšia cifra 8, tak zvyšné dva majú súčet $21 - 8 = 13$. Ten možno dosiahnuť jedine ako $7 + 6$.
- Ak je najväčšia cifra 7 alebo menej, tak ciferný súčet môže byť najviac $7 + 6 + 5 = 18$, čo je menej ako 21.

Prípustné množiny cifier sú teda $\{9, 8, 4\}$, $\{9, 7, 5\}$ a $\{8, 7, 6\}$. Každú z nich môžeme zoradiť do 3-ciferného čísla $3!$ spôsobmi (ide o permutácie), čo nám dá spolu $3 \cdot 3! = 18$ čísel. Vypísať ich môžeme v troch riadkoch, každý riadok obsahuje čísla s rovnakými ciframi, len v inom poradí. Táto tabuľka nám pekne ilustruje, prečo čísla 3 a $3!$ pri výpočte násobíme.

984	948	894	849	498	489
975	957	795	759	597	579
876	867	786	768	687	678

Jednoduchý program, ktorý nám vypíše tieto možnosti môže vyzerať takto.

```
from itertools import permutations

for cifry in [[9, 8, 4], [9, 7, 5], [8, 7, 6]]:
    for p in permutations(cifry):
        print(*p, sep='')
```

Tento program priamo kopíruje naše matematické riešenie. Keďže obsahuje dva vnorené for cykly, prvý s 3 opakovaniami a druhý s $3!$ opakovaniami, tak vypíše naozaj celkom $3 \cdot 3! = 18$ možností.

Pokiaľ by sme chceli do programu zakomponovať aj hľadanie možných ciferných súčtov, tak to môžeme napr. takto.

```
from itertools import permutations

trojice = []
# Skusame najvacsiu cifru
for a in range(9, 0, -1):
    # Skusame druhu najvacsiu cifru
    for b in range(a - 1, 0, -1):
        # Tretiu cifru si dopocitame
        c = 21 - a - b
```

```
        # Ak uz nie je tretia cifra mensia ako druha, tak nemusime hladat dalej
        if c >= b:
            break
        # Inak mame trojicu so spravnym suctom
        trojice.append([a, b, c])

for cifry in trojice:
    for p in permutations(cifry):
        print(*p, sep='')
```

Podúloha b)

V druhom riadku tabuľky máme 3 políčka. Na každé z nich máme na výber 2 možnosti. To nám dáva spolu 2^3 možností, ako môže vyzerať prostredný riadok. Tieto čísla napíšeme zelenou. Pre každú z možností určíme čiernou všetky čísla, ktoré sú jednoznačne určené. Takto dostaneme práve jednu tabuľku v 6 prípadoch, v tých, kde je stredný riadok od 010 a 101. V týchto dvoch príapadoch nevieme jednoznačne doplniť žiadne číslo. Rozoberieme preto všetky možnosti, ako môže vyzerať druhý stĺpec. Keďže v ňom máme dve políčka, tak máme 2^2 možností na jeho vyplnenie. Pre každú z nich už dostaneme jednoznačné doplnenie zvyšku.

1	1	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	0	1	0	0	0
0	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0	1	1	1	0	1	1	1
1	1	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	0	1	0	0	0

1	0	1	1	0	1
0	1	0	1	0	1
0	1	0	1	0	1
0	1	0	0	1	0
0	1	0	0	1	0
1	0	1	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0

Podúloha c)

Najskôr sa pozrieme na jeden riadok. Najskôr vyberieme zo všetkých jeho 7 políčok 4 miesta, kam dáme nuly. Na to máme $\binom{7}{4}$ možností. Potom na každé zo zvyšných 3 miest máme 2 možnosti – dáme tam 1 alebo 2 – to nám dáva 2^3 možností. Celkovo na vyplnenie jedného riadku máme tak $\binom{7}{4} \cdot 2^3$ možností. Keďže máme riadky 3 a každý vieme vyplniť nezávisle, tak máme spolu

$$\left(\binom{7}{4} \cdot 2^3\right)^3 = 280^3 = 21\,952\,000 \quad \text{možností}$$

Toto riešenie vieme priamočiaro previesť na nasledovný program.

```
from itertools import combinations, product

# Najprv si zistime moznosti pre riadok
riadky = []
# Prejdeme cez vsetky pozicie nul - zodpoveda cislu (7 nad 4)
for poz_nul in combinations(range(7), 4):
    # Prejdeme cez vsetky monosti pre zvyne 3 policka - zodpoveda 2^3
```

```
    for zvysok in product([1, 2], repeat=3):
        # Na zaklade tychto dvoch volieb vytvorime riadok - na policka, kde
        # nemaju byt nuly dame postupne cisla z trojice zvysok
        r = [0] * 7
        j = 0
        for i in range(7):
            if i not in poz_nul:
                r[i] = zvysok[j]
                j += 1
        riadky.append(r)

# Teraz prejdeme cez vsetky trojice riadkov
# Tento cyklus nam umocni pocty riadkov na tretiu
for tabulka in product(riadky, repeat=3):
    # Vypiseme tabulku a prazdny riadok
    for riadok in tabulka:
        print(*riadok, sep='')
    print()
```

Pri matematickom riešení sa odvolávame na vetu o počte kombinácií bez opakovania a variácií s opakovaním (resp. pravidlo súčinu). To zodpovedá tomu, že pri programovaní využijeme knižničné funkcie. Avšak v tejto konkrétnej situácii nie je ťažké prepísať kód aj bez nich. Pre jednoduchosť spravíme ešte jednu úpravu – miesto pozície núl vyberieme 3 pozície pre čísla, ktoré nie sú nuly.

```
from itertools import combinations, product

# Najprv si zistime moznosti pre riadok
riadky = []
# Prejdeme cez vsetky pozicie nul - zodpoveda cislu (7 nad 4)
for poz_nul in combinations(range(7), 4):
    # Prejdeme cez vsetky monosti pre zvyne 3 policka - zodpoveda 2^3
    for zvysok in product([1, 2], repeat=3):
        # Na zaklade tychto dvoch volieb vytvorime riadok - na policka, kde
        # nemaju byt nuly dame postupne cisla z trojice zvysok
        r = [0] * 7
        j = 0
        for i in range(7):
            if i not in poz_nul:
                r[i] = zvysok[j]
                j += 1
        riadky.append(r)

# Teraz prejdeme cez vsetky trojice riadkov
# Tento cyklus nam umocni pocty riadkov na tretiu
for tabulka in product(riadky, repeat=3):
    # Vypiseme tabulku a prazdny riadok
    for riadok in tabulka:
        print(*riadok, sep='')
    print()
```

Komentár. Úloha bola cielene zvolená tak, aby nebolo ľahké nechať dobehnúť celý program. Bolo to preto, aby Vás to prinútilo napísať kód efektívne (teda nie vygerovať všetky tabuľky a kontrolovať podmienky – tak to robila väčšina v úlohe a), kde to bolo schodné) a tiež viac sa zamyslieť nad tým, prečo je váš program správny. Ak bežne spustíme tento program, nemá šancu do konzoly vypísať všetky možnosti. Efektívnejšie je presmerovať výstup do súboru (napr. tak, že program spustíme príkazom `python ries_c.py > vystup.txt`. Ďalšie výrazné zrýchlenie možno docieliť použitím `sys.stdout.write` miesto `print`.

Ak máte záujem o riešenie bonusu 2, tak napíšte Jozefovi Rajníkovi.