Kategorie mladší

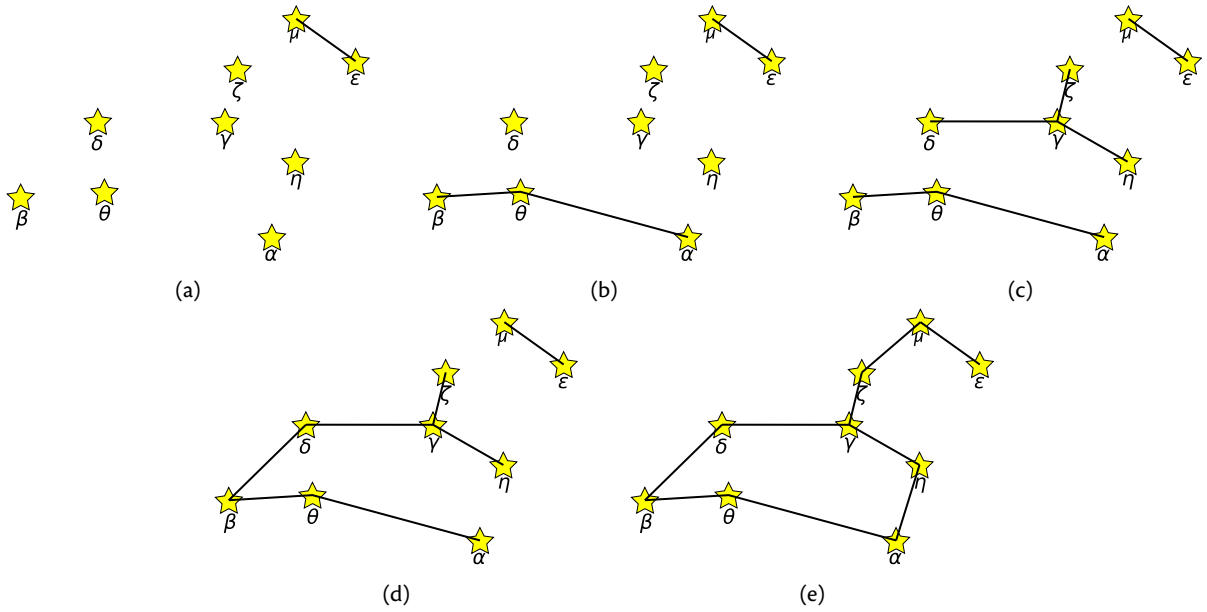
Úloha 1A Souhvězdí

Začneme pokyny, které jsou jednoznačné:

(a) Spojíme hvězdu ε s μ.
(b) Spojíme θ s β a α.

Na základě toho můžeme pokračovat:

(c) Jelikož γ není propojena s α, β, ani μ, z ε vychází pouze jedna spojnice a z θ už žádná spojnice vycházet nemůže, musí být γ propojena s ζ, δ a η.
(d) Protože je γ spojena s ζ, na základě pravidla bude β propojena s δ.
(e) A na závěr spojíme ζ s μ a α s η (jinak by se čáry protínaly).



Obrázek 1

Úloha 2A Sova a liška

Abychom mohli sově pomoci vymyslet pro lištičku postup, který bude vždy fungovat, bude dobré rozdělit si všechna čísla podle potřeby do několika skupinek. Začneme tím, že si odlišíme čísla kladná, záporná a nulu.

Nejprve se podívejme, jaký postup zvolíme u kladných čísel. Pro většinu z nich to bude snadné – stačí odečíst jedničku a dostaneme jak nejbližší možné nižší číslo, tak o jedna nižší součet číslic – například pro 16 bude takové číslo 15. Takto můžeme postupovat u všech čísel, jejichž poslední cifra je v rozmezí od jedné do devíti. Pokud je na konci nula, bude situace trochu jiná, protože odečtením jedničky dostaneme rázem číslo, které má na konci číslici 9, a nižší ciferný součet tedy rozhodně nezískáme. Protože nám v našem postupu nevyhovuje poslední číslice, budeme se muset přesunout o řád výš (tedy na úroveň desítek atd.) a budeme to dělat tak dlouho, dokud nenarazíme na nenulovou cifru, kterou pak o 1 snížíme. Tedy například pro číslo 1 000 bude dokonce nejbližší hledané číslo 0.

Podívejme se nyní na samotnou nulu. Protože ciferný součet je vždy nezáporný, je jasné, že nula bude mít tento součet nejnižší ze všech čísel. Tedy 0 bude stát na začátku celé posloupnosti.

Pro nezápornost ciferného součtu nastane problém také u záporných čísel s ciferným součtem 1 (třeba -100), protože číslo s nižším ciferným součtem je jedině 0, která je ale vyšší než tato čísla. Sova s liškou se tedy budou muset smířit s tím, že tato čísla do nové posloupnosti patřit nemohou. U ostatních záporných čísel však už postup nalézt dovedou, i když to nebude tak snadné jako u kladných. Po odečtení jedničky se zde totiž ciferný součet naopak zvýší (z -8 se třeba stane -9), tedy s řádem jednotek si nyní nevystačí. Místo jedničky tedy odečtou od daného čísla 10, a aby se ciferný součet snížil o 2 (protože předchozím krokem jej naopak o 1 zvýšíme), celkem budeme odčítat 8 (pro onu -8 tak nejbližší hledané číslo bude -16). Ne pro všechna čísla ale po odečtení osmičky dostaneme číslo s jiným řádem desítek – konkrétně pro všechna čísla s poslední cifrou 0 nebo 1 to platit nebude zcela jistě. U těchto čísel se tak budeme muset opět přesunout o řád výše a odečíst dokonce 100 a následně přičíst 20 – celkem tedy odečteme 80 (k -121 bude nejbližší -201 a podobně). Ani to nám však neošetřilo všechny možnosti, protože i v řádu desítek, stovek atd. se mohou vyskytovat 0 a 1 a máme-li

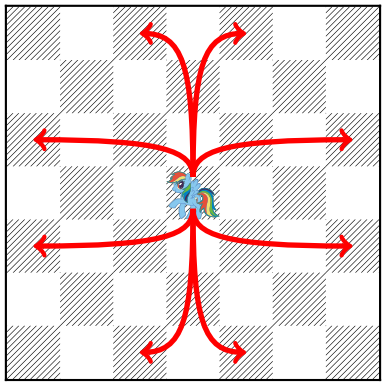
číslo, které má na několika posledních pozicích právě 1 a 0, musíme vždy odečítat 8 až u řádu, který je nenulový i nejedničkový (u -1201 tak musíme odečíst 800 a dostaneme -2001). Není to úplně snadné, nicméně jsme takto ale konečně dovedli ošetřit všechna čísla.

Sova tak může dát lištičce návod, který jí popíše, jak s kterými čísly pracovat, a lištička může svou posloupnost vesele začít přerovnávat.

Úloha 3A Hyperkůň

Hyperkůň svůj nápad úplně nedomyslel (možná to má na svědomí jeho nedávná kolize s věží).

Když normální šachový kůň začíná svůj skok na bílém políčku, doskočí na políčko černé, a naopak. Pro hyperkoně to ale neplatí: Když začne na bílém políčku, na bílé políčko také doskočí, a naopak, z černého skočí zase jenom na černé (vyznačeno na obrázku 2).



Obrázek 2: Pohyb Hyperkoně

Hyperkůň tudíž nemá na naší 2D šachovnici, ať už bude jakkoliv dlouhá, šanci přeskákat přes všechna bílá i všechna černá políčka zároveň. Může tedy jenom doufat, že jako pěšák se někdy stane dámou…

Úloha 4A Zaneprázdněný bobr

První, co od OnNa určitě chceme, je, aby nestál pořád na jednom místě, nebo jen tak nekráčel sem a tam – pokud má tedy pod sebou 0, bude vhodné ji přepsat na 1. Vybereme si, že začíná třeba ve stavu Norek a že v tomto stavu přepíše první číslici na jedničku, a následně se vydá vlevo (při prvním příkazu nám na směru nezáleží, stejně tak jsme tedy mohli zadat i vpravo).

Takto OnNo narazí na další 0, v následujícím kroku na další a celý postup se bude opakovat po celé délce pásky – tudy tedy cesta za čtyřmi jedničkami nepovede. OnNo bude muset změnit svůj stav – tedy na Ondatru. Máme tak první příkaz:

Příkaz 1: Pokud jsi Norek a pod tebou je 0, přepiš ji na 1, posuň se doleva a přejdi do stavu Ondatra.

OnNo je nyní ve stavu Ondatry a stojí na 0 vedle jedničky. O příkazu 2 víme, že budeme chtít, aby po jeho vykonání OnNo zamířil směrem doprava, tedy proti původně nabranému směru (jinak by na své cestě ať už v jakémkoli stavu narážel jen na 0 a z těch pokračoval pořád doleva). To, zda 0 přepíše na 1, nebo ne, zatím necháme nevyřešené a podíváme se na následující krok.

V tuto chvíli je OnNo opět na počáteční jedničce. Vlevo už jsme byli, z tohoto místa tedy chceme zkusit jít v rámci příkazu 3 dále doprava (opět jak otázku změny stavu, tak změny čísla necháme stranou). Při kroku doprava tedy stoupneme na další políčko s 0 a OnNo na ně vstoupí buď jako Ondatra, nebo jako Norek. Promysleme si, co by se nám více hodilo. Aby se teď OnNo opět vrátil na počáteční políčko je asi trochu zbytečné – to přesně by udělal stav Norek. Na toto místo bude tedy přicházet OnNo ve stavu Ondatry a posune se z něj dále doprava. Navštíví tak již čtvrté políčko, a nezmění-li se zde ze stavu Ondatry na Norka, opět nám uteče někam úplně pryč. K příkazu 2 tedy navíc přiřadíme podmínku, že OnNo změní stav na Norka, a k příkazu 3 to, že se stav změní zase na Ondatru (jinak bychom nemohli uskutečnit předchozí kroky).

Podívejme se nyní, co OnNo zatím udělá.

1. `...1....` (stav Norek, mění se na Ondatru, píše 1 – krok vlevo)
2. `..?1....` (stav Ondatra, mění se na Norka, psaní neznáme – krok vpravo)
3. `..??....` (stav Norek, mění se na Ondatru, psaní neznáme – krok vpravo)
4. `..???...` (stav Ondatra, mění se na Norka, psaní neznáme – krok vpravo)
5. `..???1..` (stav Norek, mění se na Ondatru, píše 1 – krok vlevo)

Končíme tedy na otazníku, značící neznámé číslo nalevo od jedničky ve stavu Ondatra. Pro Ondatru na 1 zatím žádný příkaz nemáme – za předpokladu, že by místo otazníků byly jedničky, by se tedy hodilo zde umístit konec (pokud bychom chtěli OnNa naprogramovat hodně pěkně, mohli bychom jej předtím ještě přesunout o krok doprava, aby stál na konci čtveřice jedniček, a ne uprostřed, to už je ale kosmetický detail). Podívejme se, zda by to bylo možné. K příkazu 2 přidáme příkaz přepisu 0 na 1, v příkazu 3 žádný přepis neuskutečníme a pouze skrz políčko s 1 projdeme. Takto dostaneme čtveřici příkazů:

Příkaz 1: Pokud jsi Norek a pod tebou je 0, přepiš ji na 1, posuň se doleva a přejdi do stavu Ondatra.
Příkaz 2: Pokud jsi Ondatra a pod tebou je 0, přepiš ji na 1, posuň se doprava a přejdi do stavu Norek.
Příkaz 3: Pokud jsi Norek a pod tebou je 1, nic nepřepisuj, posuň se doprava a přejdi do stavu Ondatra.
Příkaz 4: Pokud jsi Ondatra a pod tebou je 1, nic nepřepisuj, posuň se doprava a skonči.

Zkusme nyní zjistit, jak se bude OnNo chovat, když se bude řídit těmito příkazy:

1. `...1....` (stav Norek, mění se na Ondatru, píše 1 – krok vlevo)
2. `..11....` (stav Ondatra, mění se na Norka, píše 1 – krok vpravo)
3. `..11....` (stav Norek, mění se na Ondatru, nic nemění – krok vpravo)
4. `..111...` (stav Ondatra, mění se na Norka, píše 1 – krok vpravo)
5. `..1111..` (stav Norek, mění se na Ondatru, píše 1 – krok vlevo)
6. `..1111..` (stav Ondatra, nic nepřepisuje – posouvá se doleva a končí)

Zadarilo se, pomohli jsme bobrovi vymyslet jeho těžký úkol a všichni můžeme jít šťastně spát.

Úloha 5A HAF

Shrňme si nejprve, jaké operace můžeme se slovem na dveřích provádět:

1. `AAA` můžeme změnit na `F`.
2. Na druhé místo slova můžeme přidat písmenko `F`.
3. Z `FF` můžeme udělat `AAA`.

Na začátku je na dveřích napsáno slovo `HA`. Co s ním můžeme provést? Tři `A` za sebou v něm nejsou, dvě `F` také ne, takže jediné pravidlo, které můžeme použít, je pravidlo druhé, které nám z `HA` vytvoří slovo `HFA`.

Co můžeme provést se slovem `HFA`? První ani třetí pravidlo opět použít nemůžeme, takže nám nezbyde nic jiného, než znovu aplikovat pravidlo druhé, přidat další písmenko `F` a z `HFA` vyrobit `HFFA`.

Nyní máme poprvé na výběr. Mohli bychom samozřejmě potřetí použít druhé pravidlo, které by nám na začátek přidalo další `F`, a na dveřích bychom tak vytvořili slovo `HFFFA`, lepší ale bude použít pravidlo třetí a přeměnit `HFFA` na `HAAAA`. Teď totiž můžeme použít první pravidlo, poslední tři písmenka `A` změnit na `F`, a na dveřích nám stojí nápis `HAF`. Poklad je náš!

Celý postup ještě shrneme (čísla nad šipkami značí použité pravidlo):

$$\texttt{HA} \xrightarrow{(2)} \texttt{HFA} \xrightarrow{(2)} \texttt{HFFA} \xrightarrow{(3)} \texttt{HAAAA} \xrightarrow{(1)} \texttt{HAF}$$

Kategorie starší

Úloha 1B Zoomino

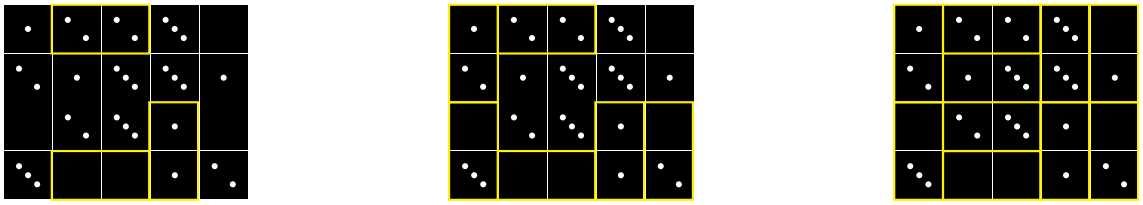
Protože si zvířátka mohou být jistá, že se při hře využijí všechny hrací kameny právě jednou, je dobré vybrat si pro začátek nějakou konkrétní dvojkostičku, kterou se pokusí na plánku nalézt. Vhodné mohou být například takové kameny, které mají oba znaky stejné, (tedy například 2 a 2 puntíky), protože takové dvojice se dají nalézt docela snadno na první pohled.

Na tomto konkrétním rozmístění najdeme pouze jednu možnost, kam umístit dvojice (0, 0), (1, 1), (2, 2) (viz obrázek 3a) – u (3, 3) je možností jak umístit dvojkostičku několik, tento hrací kámen tedy zvířátka zatím nepoloží.

Dále si můžeme všimnout, že v pravém dolním rohu, levém dolním rohu a levém horním rohu jsou políčka, která mají nyní pouze jednoho nevyužitého souseda a tedy přesně s tímto sousedem budou tvořit dvojici (obrázek 3b).

Nyní už zbývá umístit pouze poslední čtyři kameny, což už nebude nijak obtížné. Podívejme se například na poslední zbývající políčko se dvěma puntíky a na jeho sousedy – jeden má puntík jeden, druhý tři. Kostičku s kombinací puntíků (1, 2) už ale máme v levém horním rohu, vybereme tedy dvojici (2, 3), kterou jsme ještě neumístili. Nad tímto kamenem nám automaticky na sebe zbyla další dvojice, protože pole s jedním puníkem už mimo toho s puntíky třemi žádné neobsazené sousedy nemá. A ani poslední dva kameny nebudou nijak záludné, protože dvojici (3, 3) jsme stále nikam neumístili a zde nám přesně vychází.

Na závěr už jen překontrolujeme naše rozdělení, nicméně protože jsme postupovali systematicky a řešení jsme nehádali, mělo by být vše v pořádku.



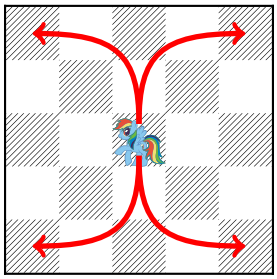
(a) (b) (c)

Obrázek 3

Úloha 2B Hyperkůň

Hyperkůň svůj nápad úplně nedomyslel (možná to má na svědomí jeho nedávná kolize s věží).

Když normální šachový kůň začíná svůj skok na bílém políčku, doskočí na políčko černé, a naopak. Pro hyperkoně to ale neplatí: Když začne na bílém políčku, na bílé políčko také doskočí, a naopak, z černého skočí zase jenom na černé (vyznačeno na obrázku 4).



Obrázek 4: Pohyb Hyperkoně

Hyperkůň tudíž nemá na naší 2D šachovnici, ať už bude jakkoliv dlouhá, šanci přeskákat přes všechna bílá i všechna černá políčka zároveň. Může tedy jenom doufat, že jako pěšák se někdy stane dámou…

Úloha 3B Kde se stala chyba

Pokud program nedobíhá, ve většině případů je chyba v některém z cyklů (v programovacím jazyce Múzy příkaz `Dokud(podmínka) opakuj(příkazy)`). Pokud je podmínka splněna pořád, program nikdy nebude pokračovat na další příkazy za závorkou. V našem

případě je podmínka `číslo` $\geq$ `0`, program tedy bude pokračovat jen pokud proměnná `číslo` bude záporná. Když se podíváme na příkazy v závorce, vidíme, že proměnná `číslo` je dělena 10, což nám ale z přirozeného čísla nikdy neudělá číslo záporné (kladné / kladným = kladné). Chyba je tedy v podmínce. Aby program spočítal ciferný součet, měl by cyklus skončit ve chvíli, kdy je proměnná `číslo` rovna 0 (číslo už nemá žádné další cifry k přičtení k cifernému součtu). Aby program fungoval, stačí přepsat podmínku na `číslo` > `0`.

Úloha 4B Marticové počty

Podívejme se nejprve na druhou otázku, kterou nám zvědaví zvířecí matematikové kladou: Musí mít marticová jednička nějaké speciální rozměry?

Zopakujme si, jak Martin marticové násobení zavedl: Číslo na políčku (i, j) (i-tý řádek, j-tý sloupec) získáme tak, že spolu vynásobíme první číslo i-tého řádku první martice (označíme ji M_1) s prvním číslem j-tého sloupce druhé martice (M_2), druhé číslo i-tého řádku M_1 s druhým číslem j-tého sloupce M_2,... a všechny součiny spolu poté sečteme. To ovšem znamená, že počet čísel v řádcích M_1 musí být stejný jako počet čísel ve sloupcích M_2, jinými slovy, šířka M_1 se rovná výšce M_2. Pokud tedy řekneme, že M_1 má rozměry $a \times b$ (výška $\times$ šířka), pak M_2 musí mít rozměry $b \times c$. Vynásobit tak spolu vesele můžeme např. martice o rozměrech 1×3 a 3×17, nebo 5×5 a 5×8, ale s marticemi o rozměrech 5×5 a 8×5 už se nám to nepovede. Na tomto místě se můžeme rovnou zamyslet také nad tím, jaké rozměry bude mít martice výsledná: Číslo na políčku (i, j) získáme vynásobením i-tého řádku M_1 a j-tého sloupce M_2, výsledná martice tedy zdědí počet řádků od M_1 a počet sloupců od M_2.

Celý proces marticového násobení si můžeme lépe představit tak, že si nakreslíme následující schéma: Řekněme, že chceme vynásobit martice

$$M_1 = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 3 & 2 & 1 \\ \hline \end{array} \quad \text{a} \quad M_2 = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & 1 \\ \hline 1 & 2 \\ \hline \end{array}$$

Nakreslíme si kříž, do levého dolního rohu umístíme M_1, do pravého horního M_2, do pravého dolního budeme vyplňovat výslednou martici. Jak? Inu, budeme spolu násobit čísla v příslušném řádku s čísly v příslušném sloupci:

			1	2
			2	1
			1	2
1	2	3	$1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1$	$1 \cdot 2 + 2 \cdot 1 + 3 \cdot 2$
3	2	1	$3 \cdot 1 + 2 \cdot 2 + 1 \cdot 1$	$3 \cdot 2 + 2 \cdot 1 + 1 \cdot 2$

Z tohoto schématu jsou také patrné jak požadavky na rozměry násobených martic, tak rozměry výsledné martice.

Na druhou otázku jsme tedy odpověděli, čas vrátit se k té první. Existuje něco jako marticová jednička?

Ze všeho nejdřív se zamysleme nad tím, zda může existovat univerzální marticová jednička, tedy taková martice J, kterou můžeme vynásobit s libovolnou marticí M a jako výsledek dostaneme tutéž martici M. Víme, že násobit spolu můžeme jenom martice, které mají správné rozměry, zdá se tedy, že pokud už nějaká marticová jednička existuje, nebude nám stačit jedna, ale budeme jich potřebovat celou sadu o různých rozměrech. Jaké to budou rozměry? Budou marticové jedničky obdélníkové, nebo čtvercové? Řekněme, že máme martici M o rozměrech $a \times b$. Vynásobíme ji marticovou jedničkou, J, a získáme tu samou martici M, tedy $M \cdot J = M$. Výšku martice J (počet řádků) můžeme odvodit z toho, že ji můžeme vynásobit s marticí M – výška J musí být stejná jako šířka M, tedy b. Šířku J můžeme naopak odvodit z výsledku násobení, víme, že bude stejně široká jako výsledná martice – tedy b. Rozměry J jsou tedy $b \times b$, J je čtvercová.

No dobrá, ale co kdybychom se rozhodli násobit naopak, tedy místo $M \cdot J$ vypočítali $J \cdot M$? Stejnými úvahami jako výše dojdeme k tomu, že nyní musí mít J rozměry $a \times a$. Ale $a \times a$ přece není totéž jako $b \times b$! V normální matematice je jedno, v jakém pořadí čísla násobíme ($1 \cdot 5$ je totéž jako $5 \cdot 1$), v marticové matematice to ale evidentně neplatí. K jediné martici M mohou existovat dokonce dvě marticové jedničky, každá jinak veliká!

Mohlo by se zdát, že hledání univerzální marticové jedničky můžeme vzdát – vždyť jsme právě ukázali, že i k jedné jediné martici M můžou (pokud je M obdélníková) existovat takové jedničky hned dvě, v závislosti na tom, zda chceme jedničkou násobit zleva či zprava. Nevzdávejme se ale ještě – třeba se ukáže, že i tak můžeme všechny marticové jedničky nějak jednoduše popsat.

Budeme uvažovat případ, kdy jedničkou násobíme zprava, řešíme tedy rovnici $M \cdot J = M$. Jak získáme číslo na políčku (i, j) (budeme je

značit $M_{i,j}$)? Postup známe, budeme násobit čísla z i-tého řádku matice M s čísly z j-tého sloupce matice J a výsledky sčítat. Tedy:

$$\begin{aligned}
M_{1,1} &= M_{1,1} \cdot J_{1,1} + M_{1,2} \cdot J_{2,1} + \cdots + M_{1,b} \cdot J_{b,1} \\
M_{2,1} &= M_{2,1} \cdot J_{1,1} + M_{2,2} \cdot J_{2,1} + \cdots + M_{2,b} \cdot J_{b,1} \\
M_{3,1} &= M_{3,1} \cdot J_{1,1} + M_{3,2} \cdot J_{2,1} + \cdots + M_{3,b} \cdot J_{b,1} \\
&\vdots \\
M_{a,1} &= M_{a,1} \cdot J_{1,1} + M_{a,2} \cdot J_{2,1} + \cdots + M_{a,b} \cdot J_{b,1} \\
M_{1,2} &= M_{1,1} \cdot J_{1,2} + M_{1,2} \cdot J_{2,2} + \cdots + M_{1,b} \cdot J_{b,2} \\
M_{2,2} &= M_{2,1} \cdot J_{1,2} + M_{2,2} \cdot J_{2,2} + \cdots + M_{2,b} \cdot J_{b,2} \\
&\vdots
\end{aligned}$$

Jinými slovy, dostaneme hrozně moc rovnic pro hrozně moc neznámých ☺ Mohli bychom se nyní samozřejmě pokusit tuto soustavu vyřešit, jistě by to šlo a (pokud bychom někde neudělali chybu) jistě bychom i došli ke správnému výsledku, snazší ale bude pořádně se na rovnice podívat. Vidíme, že všechna čísla v prvním sloupci výsledné matice získáme tak, že řádky matice M násobíme prvním sloupcem matice J, čísla v druhém sloupci výsledné matice tak, že řádky M násobíme druhým sloupcem matice J, atd. Zaměřme se zatím na první sloupec. Byli bychom rádi, aby výsledkem tohoto násobení a sčítání byl jenom první prvek každého řádku – ale to přeci umíme jednoduše zařídit! Stačí, aby první číslo v prvním sloupci J byla jednička a všude jinde byly 0, a dostaneme přesně to, co potřebujeme! Jak to nyní bude s druhým sloupcem výsledné matrice? Tam po vynásobení potřebujeme získat pouze druhý prvek každého řádku – uděláme tedy podobnou úvahu a řekneme, že druhý prvek druhého sloupce J musí být 1 a všude jinde musí být 0. Podobně můžeme pokračovat až po b-tý sloupec. Jinými slovy, J bude vypadat tak, že na diagonále budou samé jedničky a všude jinde samé 0. Např. maticová jednička o rozměrech 2×2 tak bude vypadat takto:

1	0
0	1

o rozměrech 5×5 takto:

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

Skvělé, známe tedy maticovou jedničku pro násobení zprava. Bude stejně vypadat i maticová jednička pro násobení zleva? Vypišme si opět několik rovnic (a tentokrát nebudeme postupovat po sloupcích, ale po řádcích):

$$\begin{aligned}
M_{1,1} &= J_{1,1} \cdot M_{1,1} + J_{1,2} \cdot M_{2,1} + \cdots + J_{1,b} \cdot M_{b,1} \\
M_{1,2} &= J_{1,1} \cdot M_{1,2} + J_{1,2} \cdot M_{2,2} + \cdots + J_{1,b} \cdot M_{b,2} \\
M_{1,3} &= J_{1,1} \cdot M_{1,3} + J_{1,2} \cdot M_{2,3} + \cdots + J_{1,b} \cdot M_{b,3} \\
&\vdots
\end{aligned}$$

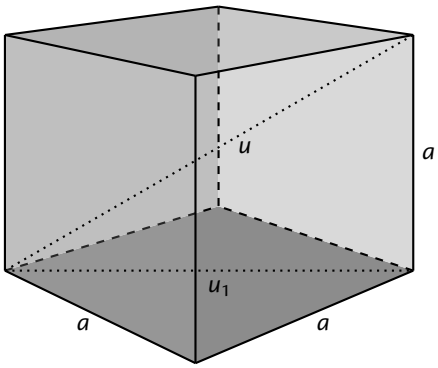
Vidíme, že vše funguje podobně jako v předchozím případě, jen tentokrát násobíme sloupce matice M řádky matice J. Chceme, aby pro první řádek J platilo, že prvním prvkem je 1 a ostatní jsou 0, pro druhý řádek J platilo, že druhým prvkem je 1 a ostatní jsou 0, atd., dostaneme tedy úplně stejný typ matice: Na diagonále jsou 1 a všude jinde jsou 0.

Zvířecím matematikům tedy můžeme odpovědět, že maticová jednička existuje i neexistuje. Jediná matice J, pro kterou by platila požadovaná vlastnost, neexistuje, pokud ale libovolnou matici M vynásobíme (ať už zprava, nebo zleva) maticí o příslušných rozměrech, která má na diagonále 1 a jinak všude 0, dostaneme opět tutéž matici M.

Úloha 5B Jihoafrická autobusová

Kdybychom se přesunuli do 2D prostoru, a udělali tedy z oštěpu úsečku a z krychle čtverec, oštěp by mohl mít maximálně délku úhlopříčky (delší úsečku uvnitř čtverce nenajdeme). Na základě Pythagorovy věty bychom snadno došli k tomuto výsledku:

$$u_1 = \sqrt{1^2 + 1^2} = \sqrt{2}$$



Pro krychli to bude úplně stejné, jen nejdelší úsečkou nebude úhlopříčka čtverce, ale tělesová úhlopříčka krychle. Ta se vypočte rovněž pomocí Pythagorovy věty. Jednou odvěsnou bude hrana krychle a, druhou úhlopříčka čtverce u_1. Dosadíme do vzorce a jsme hotovi.

$$u = \sqrt{\sqrt{2}^2 + 1^2} = \sqrt{2+1} = \sqrt{3} \doteq 1{,}73m$$

Skokan Pepan si tedy do autobusu může vzít oštěp dlouhý maximálně 1,73 m.

Poznámka: Existuje i kratší postup a to využití vzorce pro výpočet tělesové úhlopříčky kvádru $u = \sqrt{a^2 + b^2 + c^2}$. *V případě krychle pak získáme zjednodušený vzorec* $u = \sqrt{3a^2}$.