# Jakýsi úvod do diskrétní matematiky

Áďa Klepáčovic

15. května 2023

# Obsah

# 1 | Zajímavé problémy

Jednou z hlavních a oblíbených podmnožin diskrétní matematiky a kombinatoriky je *teorie grafů*. Jednoduše řečeno, graf je množina bodů – vrcholů, mezi některými vedou spojnice – hrany.

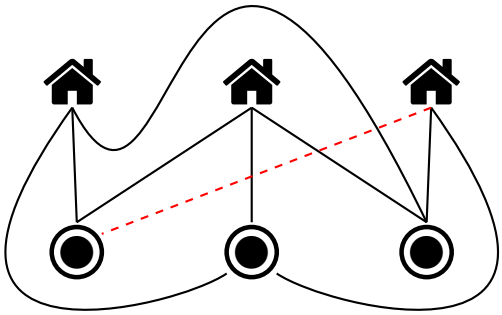
Představíme si pár úloh jako ze života, jak bývá v matematice zvykem, jejichž zdánlivá nevinnost je v rozporu s jejich významem pro rozvoj teorie.

## 1.1 Problém tří domů a tří studní

V zemi za sedmero horami a sedmero řekami žili byli tři sousedi. Každý z nich vlastnil rodinný domek a dělili se o vodu ze tří blízkých studní. Jednou se ale sousedi po sporu rozkmotřili a už se nechtěli ani vidět. Potřebovali ale pitnou vodu. Každý se odmítl vzdát nároku na nějakou ze studní, tak si jako poslední společný čin najali dvorního architekta, by nechal postavit cesty od každého domu ke každé studni. Cesty se nesměly nikde potkat, aby do sebe sousedi na cestě pro vodu náhodou nenarazili.

Dvorní architekt, probděv tři dny a tři noci hledaje způsob, jak nasupené sousedy uspokojit, klekl nakonec únavou a prohlásil, že cesty bez křížení vystavět nelze.

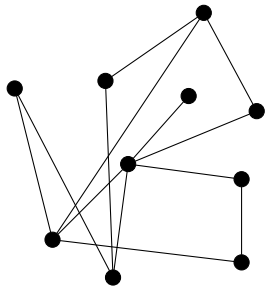
My s ním souhlasíme, ale není lehké najít způsob, jak úlohu matematicky formalizovat, a podat důkaz.



Obrázek 1: Tři domy a tři studně.

## 1.2 Hrátky s puntíky

Ukážeme si ještě dvě pěkné úlohy s puntíky a čarami. Mějme třeba deset puntíků v rovině a pár z nich spojme čarami, jako na obrázku 2.



Obrázek 2: Náhodný graf na deseti vrcholech.

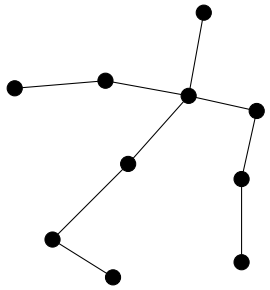
Otázka, kterou se budeme časem zabývat zní „Kolik maximálně mohu nakreslit spojnic, než mi vznikne trojúhelník?“ Trojúhelníkem zde myslíme trojici bodů, z nichž každé dva jsou spojeny. Do tohoto grafu se určitě ještě dají nějaké přikreslit, ale kolik přesně? A jak tuto úlohu řešit obecně pro jakýkoliv počet bodů?

Podobně zajímavá, ale už víc algoritmická otázka, by zněla „Jak poznám, jestli v nějakém grafu existuje trojúhelník?“. Samozřejmě by šlo se prostě podívat na každou jednu trojici bodů, ale jde to i líp?

Ještě na pár odstavců zůstaneme u spojených puntíků. Úloha, která se ukázala jako zásadní v teorii grafů má co dělat s cestami. *Cestou* v grafu nazveme posloupnost bodů – vrcholů, takovou, že mezi sousedními vrcholy na cestě vždycky vede spojnice. Jinak řečeno, mohu se v klidu projít od jednoho vrcholu k druhému, aniž bych musel skákat mezi vrcholy. Naším úkolem je najít takovou množinu spojnic – hran, že se mezi každými dvěma vrcholy dá projít po cestě.

Pro deset vrcholů jedno možné řešení vidíte na obrázku 3.

Je jednoduché si rozmyslet, kolik nejméně hran je třeba nakreslit. Ovšem, co když můžeme vybírat jen z nějaké předem dané množiny? Řešení pak nemusí vždycky existovat (může se totiž stát, že žádné hrany k dispozici nemáme). Dá se nějak efektivně poznat, kdy úlohu lze řešit a kdy ne? A co třeba otázka, kolik existuje řešení s minimálním počtem hran; co řešení, součet přes všechny jehož hrany je nejmenší? V této obecné podobě



Obrázek 3: Minimální kostra grafu na deseti vrcholech.

se úloze (i jejímu řešení) říká *minimální kostra* grafu a v budoucnu ji, stejně jako předchozí dvě úlohy, potkáme.

# 2 | Úvodní pojmy

Žádná matematická disciplína se neobejde bez pochopení základů logiky a teorie množin. Pro jistotu zde nejnutnější části připomeneme, ale tyto krátké úryvky nezamýšlejí naučit, leč osvěžit.

## 2.1 Logické spojky a kvantifikátory

**Definice 2.1.1** (Výrok)**.** Výrokem nazveme jakoukoli větu, o které lze rozhodnout, zda je pravdivá, či nikoliv.

**Příklad.** Věty „Je mi zle.“ a „Sumec je drůbež.“ jsou výroky, zatímco „Tvoje máma.“ a „Cos' dostala z matiky?“ nikoliževěk.

Je též dlužno mít na paměti, že naše znalost pravdivosti věty nemění nic na tom, jestli daná věta je, nebo není výrokem. Třeba „Do pěti století kolonizujeme celou Sluneční soustavu.“ je zcela jistě výrok.

Další text vyžaduje znalost operátorů $\neg, \wedge, \vee, \Rightarrow$ a $\Leftrightarrow$. Je-li $x$ výrok „Prší.“ a $y$ výrok „Vezmu si deštník.“, pak

- výrok $\neg x$ znamená „**Ne**prší.“,
- výrok $x \wedge y$ znamená „Prší **a** vezmu si deštník.“,
- výrok $x \vee y$ znamená „Prší **nebo** si vezmu deštník.“,
- výrok $x \Rightarrow y$ znamená „**Když** prší, **tak** si vezmu deštník.“ a
- výrok $x \Leftrightarrow y$ znamená „Prší, **právě tehdy když** si vezmu deštník.“

**Výstraha.**

- Logická spojka $\vee$ **není výlučná**. Tedy $x \vee y$ platí v situaci, kdy
  - platí pouze $x$,
  - platí pouze $y$,
  - platí $x$ i $y$.
- Výrok $x \Rightarrow y$ je vždy **pravdivý**, pokud $x$ je **lživý**. Jinak řečeno, $x \Rightarrow y$ platí za situace, kdy
  - platí $x$ i $y$,
  - neplatí $x$ a platí $y$,
  - neplatí $x$ a neplatí $y$.

Jako znalost logických spojek je kritická i znalost kvantifikátorů $\forall$ a $\exists$, které se čtou „pro všechny“ a „existuje“, resp.

Pokud je $p(x)$ výrok závislý na proměnné $x$ (třeba „$x$ je sudé.“), pak výrok

- $\forall x \in \mathbb{N} : p(x)$ zní „Všechna přirozená čísla jsou sudá.“ a
- $\exists x \in \mathbb{N} : p(x)$ zní „Existuje sudé přirozené číslo.“

Budeme rovněž užívat kvantifikátory $\exists!$ a $\nexists$, které znamenají „existuje přesně jeden“ a „neexistuje“.

Podáno intutivně: chci-li tvrdit, že $\forall x \in \mathbb{N} : p(x)$, musím dokázat, že ať mi nepřítel dá **jakékoliv** přirozené číslo $x$, tak $p(x)$ platí. Naopak, dokázat $\exists x \in \mathbb{N} : p(x)$ je obvykle zásadně jednodušší, neboť musím pouze najít **jedno** přirozené číslo $x$, pro které $p(x)$ platí.

## 2.2 Množiny

Požaduji znalost značek $\in, \cap, \cup, \setminus, \times$ a $\subseteq$. Pro připomenutí, jsou-li $A, B$ dvě množiny, pak

- výrok $x \in A$ říká, že „$x$ je prvkem $A$.“ nebo „ $x$ patří do $A$.“;
- $A \cap B$ je **průnik** $A$ s $B$, čili množina obsahující prvky, které patří jak do $A$, tak do $B$;
- $A \cup B$ je **sjednocení** $A$ s $B$, čili množina obsahující prvky, které patří do $A$ nebo do $B$;
- $A \setminus B$ je **rozdíl** $A$ s $B$, čili množina obsahující prvky, které patří do $A$ a nepatří do $B$;
- $A \times B$ je **součin** $A$ s $B$, čili množina **uspořádaných** dvojic $(a, b)$, kde $a \in A$ a $b \in B$. Uspořádaná dvojice zde znamená, že $(a, b) \neq (b, a)$, tedy záleží na tom, který prvek je první a který druhý;
- výrok $A \subseteq B$ říká, že $A$ je podmnožinou $B$, tedy, že každý prvek $A$ je rovněž prvkem $B$.

Pro mnohonásobné a nekonečné verze budeme používat stejné symboly (s výjimkou součinu). Tedy, mám-li množiny $A_1, \ldots, A_n$, pak

- $\bigcap_{i=1}^{n} A_i$ je jejich průnik,
- $\bigcup_{i=1}^{n} A_i$ je jejich sjednocení a
- $\prod_{i=1}^{n} A_i$ je jejich součin.

Když jsou počáteční a koncový index známy z kontextu, budeme je vynechávat a psát pouze třeba $\bigcup A_i$. Součin množiny se sebou samou budeme často zkracovat mocninným zápisem, třeba $A \times A \times A = A^3$.

**Příklad.** Je-li $A = \{1, 3, 4\}$ a $B = \{2, 4, 5\}$, pak

- $A \cap B = \{4\}$,
- $A \cup B = \{1, 2, 3, 4, 5\}$,
- $A \setminus B = \{1, 3\}$ a

- $A \times B = \{(1,2),(1,4),(1,5),(3,2),(3,4),(3,5),(4,2),(4,4),(4,5)\}$.

**Definice 2.2.1.** Je-li $A$ množina, pak

- $\#A$ značí **počet prvků** $A$ neboli **velikost** $A$,
- $2^A$ značí **množinu všech podmnožin** $A$, čili

$$2^A := \{B \mid B \subseteq A\}.$$

Pro nekonečné množiny píšeme $\#A = \infty$.

**Výstraha.** Pojem velikosti takto zavedený není korektně definovaný. Není totiž jasné, co by měl „počet“ prvků znamenat. Pojem *bijekce* ze sekce o zobrazeních tento problém vyřeší.

**Tvrzení 2.2.2** (Vlastnosti velikosti množiny)**.**

(1) $\#A \times B = \#A\#B$.

(2) $\#2^A = 2^{\#A}$.

**Důkaz.**

(1) Pro každý prvek $a \in A$ je v $A \times B$ právě $\#B$ dvojic $(a,b)$, kde $b \in B$. Jelikož prvků $a \in A$ je z definice $\#A$ a každému odpovídá $\#B$ dvojic $(a,b)$, je celkový počet uspořádaných dvojic v $A \times B$ právě $\#A\#B$.

(2) Pro nekonečné množiny tvrzení platí zřejmě. Předpokládejme, že $A$ je konečná.

Očíslujeme si podmnožiny $A$ binárními čísly délky $\#A$. Každá podmnožina $A$ vznikne totiž tak, že procházíme postupně všechny prvky $A$ a u každého se rozhodujeme, zda ho do ní zařadíme či nikoliv. Kladnému rozhodnutí bude odpovídat cifra 1 a zápornému 0. Má-li $A$ řekněme 5 prvků, pak podmnožina očíslovaná číslem 00110 je podmnožina, která obsahuje pouze 3. a 4. prvek z $A$ (při libovolném, **ale fixním**, očíslování samotné množiny $A$).

Odtud plyne, že $A$ má tolik podmnožin, kolik je různých binárních čísel délky $\#A$. Těch je však $2^{\#A}$, jak jsme chtěli. □

## 2.3 Relace

Pojem *relace* zobecňuje věci jako zobrazení (se kterým jste se setkali, ale říkali jste mu bůhvíproč funkce) nebo uspořádání (které taky znáte, jen vám bůhvíproč neprozradili, oč jde).

Základní myšlenkou je to, že i relace – vztahy mezi objekty se dají pomocí množin (a jejich součinu) úspěšně definovat. Celá matematika, kterou jste dosud poznali, je založená na *teorii množin*, jinak řečeno, **všechno** je množina.

**Definice 2.3.1** (Relace). Jsou-li $A, B$ množiny, pak **relací** mezi $A$ a $B$ nazveme *libovolnou* podmnožinu $A \times B$. Je-li $A = B$, pak $R$ nazýváme relací na $A$.

Pojem relace v matematice je založen na konceptu, že vztah mezi množinami je dokonale popsán výpisem všech dvojic prvků, které v tom vztahu jsou. To se trochu liší od běžného chápání slova „vztah“. Asi byste nebyli úplně spokojení, kdybychom vám tvrdili, že vztah manželský na množině všech lidí je to samé, co výpis všech manželských párů. Z toho důvodu bude asi lepší se držet latinské verse, „relace“.

Protože nejstarší typy relací, mezi nimi třebas $<$ nebo $=$, lidé používali ještě před vznikem samotné teorie množin, značení je zde trochu matoucí. Fakt, že dvojice $(x, y) \in A \times B$ je v relaci $R$, nezapisujeme (jak by se čekalo) $(x, y) \in R$, ale spíš $xRy$. Podobně jako nepíšeme $(x, y) \in <$, ale $x < y$.

Jako spoustu věcí v matematice, relace je dobré si umět vizualizovat. Ukážeme si teď tři standardní způsoby, jak si lidé relace kreslí.

### 2.3.1 Kreslení relací

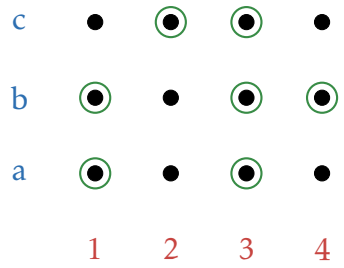
Po celou podsekci budeme předpokládat, že máme množiny $A = \{1, 2, 3, 4\}$ a $B = \{a, b, c\}$.

Jedním ze způsobů, jak se dají kreslit relace, je *mříž*. Uvážíme relaci

$$R = \{(1, a), (1, b), (2, c), (3, a), (3, b), (3, c), (4, b)\}$$

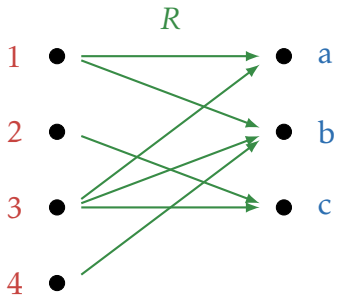
mezi $A$ a $B$. Vizualizaci součinu $A \times B$ a relace $R$ pomocí mříže vidíte na

obrázku 4.




Obrázek 4: Kreslení relace $R \subseteq A \times B$ pomocí mříže.

Ještě jeden užitečný způsob kreslení, který funguje pro obecné relace, je kreslení pomocí šipek. V zásadě si člověk zobrazí obě množiny jako sloupce bodů a mezi příslušnými body kreslí šipky. Například jako na obrázku 5.




Obrázek 5: Kreslení relace $R \subseteq A \times B$ pomocí šipek.

Tenhle způsob se může zdát méně přehledný než mříž, ale má svoje nesporné využití, především v oblasti *skládání* relací, kterým se budeme zabývat za chvíli.

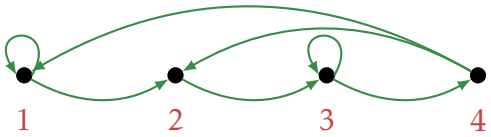
Ještě před tím si ale ukážeme způsob, jak přehledně kreslit relace na nějaké množině. Řekněme, že tentokrát je třeba

$$R := \{(1,1),(1,2),(2,3),(3,4),(3,3),(4,1),(4,2)\}$$

relace na množině $A$. Množinu $A$ si nakreslíme jako body v rovině a relaci $R$ jako šipky a smyčky. Vizte obrázek 6.

### 2.3.2 Skládání relací

V této podsekci si řekneme, co znamená, že dvě (nebo více) relace složíme dohromady. Tato operace se dá vnímat jako jakési „zobecnění“ skládání




Obrázek 6: Kreslení relace $R$ na $A$ pomocí šipek a smyček.

zobrazení/funkcí. Jak si ale ukážeme, zobrazení jsou speciálním typem relací, takže tahle představa není úplně vhodná.

Pro jednoduchost se budeme soustředit na relace na nějaké množině $A$. Tohle ovšem není nutné; mám-li relaci $R \subseteq A \times B$ a relaci $S \subseteq B \times C$, vždy je mohu složit a dostat relaci mezi $A$ a $C$.

Skládání relací není nijak divoká věc a vztahy (například mezi lidmi) v životě běžně skládáme, ale málokdy se na to asi díváme tímto způsobem. Například, řekněme, že Adéla má přítelkyni Simona a Simona má přítelkyni Terezu. Když složíme relace „býti přítelkyně Adély“ a „býti přítelkyně Simony“ dostaneme relaci, ve které je Tereza přítelkyně Adély. Na druhý příklad, třeba samotné přísloví „Nepřítel mého nepřítele je můj přítel.“, se dá vyložit jako skládání relací.

Teď formálně.

**Definice 2.3.2** (Složení relací)**.** Mějme množinu $A$ a relace $R, S \subseteq A \times A = A^2$. Složením relací $R$ a $S$ nazveme množinu

$$\{(x,z) \in A^2 \mid \exists y \in A : xRy \wedge yRz\}$$

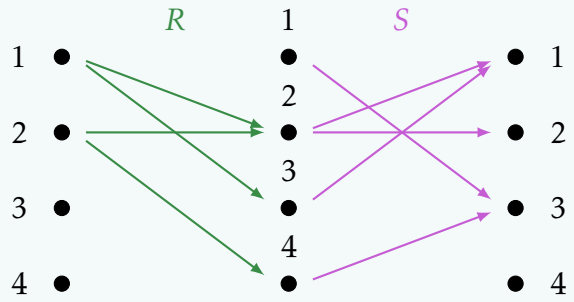
a značíme ji $R \circ S$.

Řečeno asi možná třeba trošku lidštěji, když pro dané $x, z \in A$ najdu takový prvek $y \in A$, že dvojice $(x,y)$ je v relaci $R$ a dvojice $(y,z)$ je v relaci $S$, pak $(x,z)$ je v relaci $R \circ S$. Vlastně $(x,y)$ a $(y,z)$ slepím dohromady skrze $y$.

**Příklad.** Řekněme, že je $A = \{1,2,3,4\}$ a máme relace

$$R := \{(1,2),(1,3),(2,2),(2,4)\},$$
$$S := \{(1,3),(2,1),(2,2),(3,1),(4,3)\}$$

na $A$. V podsekci o kreslení relací jsme zmínili, že šipky jsou velmi užitečné při skládání. Teď uvidíte proč. Když si obě relace nakreslíme přímo vedle sebe, dostaneme obrázek 7.




Obrázek 7: Složení relací $R$ a $S$.

V roli $x$ z Definice 2.3.2 je zde první sloupec, v roli $y$ druhý a v roli $z$ třetí. Čili, prvek $(x,z)$ bude v relaci $R \circ S$ jenom tehdy, když najdu v prostředním sloupci prvek $y$ (aspoň jeden, ale klidně víc), přes který dokážu po šipkách dojít z $x$ do $z$.

Z obrázku je teď už zřejmé, že

$$R \circ S = \{(1,1),(1,2),(2,1),(2,2),(2,3)\}.$$

## 2.4 Ekvivalence

Jedním speciálním typem relace na množině je tzv. *ekvivalence*. Důvodem pro tenhle název je fakt, že prvky, které jsou v relaci ekvivalence, jde za jisté interpretace považovat za „stejné“. Asi nejobyčejnější příklad užití ekvivalence je při definici množiny racionálních čísel, $\mathbb{Q}$, jak si brzy ukážeme. Nejprve ale definice ekvivalence.

**Definice 2.4.1** (Ekvivalence)**.** Relace $R \subseteq A^2$ je

- **reflexivní**, když je každý prvek v relaci sám se sebou, tj.
$$xRx \quad \forall x \in A;$$
- **symetrická**, když ke každé dvojici obsahuje i opačně uspořádanou, tj.
$$xRy \Rightarrow yRx \quad \forall x, y \in A;$$
- **transitivní**, když ke každým dvěma dvojicím, které jdou „slepit přes prostředníka“ (vizte definici skládání) obsahuje i tu slepenou dvojici. Formálně,
$$xRy \wedge yRz \Rightarrow xRz \quad \forall x, y, z \in A.$$

Relace, která je *reflexivní*, *symetrická* a *transitivní* se nazývá **ekvivalence**.

Vlastnosti reflexivity, symetrie a transitivity nejsou principiálně v žádném vztahu. Existují relace, které jsou jen reflexivní, ale nejsou ani symetrické ani transitivní apod. Jeden příklad za všechny.

**Příklad.** Položme $A := \{1, 2, 3, 4\}$. Relace

- $\{(1,1), (2,2), (3,3), (4,4), (1,2), (2,3)\}$ je reflexivní, ale nikoli symetrická nebo transitivní;
- $\{(1,1), (2,2), (1,2), (2,1), (2,3), (3,2)\}$ je symetrická, ale není reflexivní ani transitivní;

- $\{(1,2),(2,3),(1,3),(3,4),(1,4),(2,4)\}$ je transitivní, ale není reflexivní ani symetrická.

Ekvivalence je velmi přirozený způsob, jak ztotožnit prvky, které bychom, často z technických důvodů, nechtěli považovat za různé. Vrátím-li se k příkladu zlomků, asi bychom nechtěli vidět třeba 1/5 a 2/10 jako dva různé zlomky. Zlomek 1/5 v tomto smyslu je vlastně množina všech zlomků, které představují stejnou hodnotu. Tuto intuici zobecňuje pojem *třídy ekvivalence*.

**Definice 2.4.2** (Třída ekvivalence)**.** Mějme ekvivalenci $R \subseteq A^2$ a prvek $x \in A$. **Třídou ekvivalence** prvku $x$ **vzhledem k R** myslíme množinu

$$[x]_R := \{y \in A \mid xRy\},$$

čili množinu všech prvků, které jsou s ním v relaci $R$. Dolní index $R$ v zápisu $[x]_R$ budeme často vynechávat a psát jen $[x]$. Uvědomme si, že nezáleží na tom, jestli napíšu $xRy$ nebo $yRx$ v definici výše, protože $R$ je symetrická.

**Příklad** (Racionální čísla)**.** Symbolem $\mathbb{N}$ značím množinu přirozených čísel $\{1,2,3,\ldots\}$ a symbolem $\mathbb{Z}$ množinu celých čísel, tj. množinu přirozených čísel, čísel k nim opačných a 0.

Racionální čísla se dají definovat jako všechny možné podíly celého čísla přirozeným. Když si zlomek $a/b$, kde $a \in \mathbb{Z}$ a $b \in \mathbb{N}$ představím jako uspořádanou dvojici $(a,b)$, tj. (čitatel, jmenovatel), pak množina

$$A := \{(a,b) \mid a \in \mathbb{Z}, b \in \mathbb{N}\}$$

je množina všech zlomků.

Ujasníme si, kdy dva zlomky považujeme za stejné. Snadno úpravou člověk dostane, že

$$\frac{a}{b} = \frac{c}{d} \Leftrightarrow ad = bc,$$

což nám dává návod, jak definovat ekvivalenci na množině všech zlomků, $A$. Relaci $R \subseteq A^2$ definujeme tím způsobem, že $(a,b)R(c,d)$ právě tehdy, když $ad = bc$. Správně bychom měli dokázat, že to je opravdu ekvivalence, ale tím se nehodláme zdržovat.

Množina racionálních čísel, na kterou jste zvyklí, se pak nejelegantněji definuje jako množina tříd ekvivalence prvků z $A$ vzhledem k $R$. Konkrétně,

$$\mathbb{Q} := \{[(a,b)]_R \mid a \in \mathbb{Z}, b \in \mathbb{N}\}.$$

Třídy ekvivalence jistým způsobem „parcelují“ množinu $A$ na disjunktní (mající prázdný průnik) množiny. To je obsahem následujícího tvrzení, jehož důkaz je cvičení.

**Tvrzení 2.4.3** (Vlastnosti tříd ekvivalence)**.** Nechť $A$ je libovolná množina a $R$ je ekvivalence na $A$. Pak

(1) $[x] \neq \emptyset$ pro všechna $x \in A$,

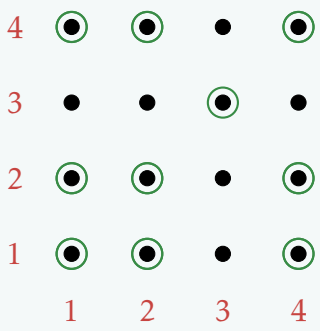
(2) Buď $[x] = [y]$, nebo $[x] \cap [y] = \emptyset$ pro všechna $x, y \in A$.

**Důkaz.** Cvičení. □

**Příklad.** Řekněme, že $A$ je naše oblíbená množina $\{1,2,3,4\}$. Snadno ověříme, že

$$R := \{(1,1),(1,2),(1,4),(2,1),(2,2),(2,4),(3,3),(4,1),(4,2),(4,4)\}$$

je ekvivalence na $A$. Její mříž vidíte na obrázku 8.

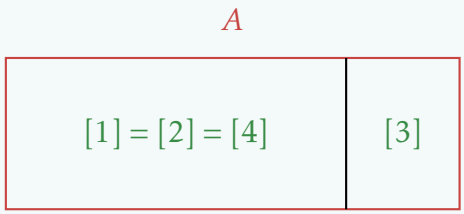



Obrázek 8: Mříž ekvivalence $R$ na množině $A$.

Obecně, mříž každé ekvivalence má zaplněnou diagonálu z levého dolního rohu do pravého horního (kvůli reflexivitě) a je symetrická

podle této diagonály (kvůli symetrii). Jak na první pohled poznat transitivitu nevím.

Všimněme si, že $1R2$ a $1R4$, takže $2 \in [1]$ a $4 \in [1]$. Podle tvrzení nahoře je $[1] = [2] = [4]$, protože tyto třídy ekvivalence nejsou disjunktní. Naopak, třída $[3]$ **je** disjunktní s každou z nich. Můžeme proto rozdělit množinu $A$ na třídy ekvivalence třeba jako $A = [1] \cup [3]$. Náhled na obrázku 9.




Obrázek 9: Množina $A$ rozdělená na třídy ekvivalence $R$.

Pár cvičení nakonec.

**Cvičení 2.4.1.** Dokažte Tvrzení 2.4.3.

**Cvičení 2.4.2** (Skládání relací vs. transitivita)**.** Dokažte, že relace (ne nutně ekvivalence!) $R$ je transitivní, právě tehdy když $R \circ R \subseteq R$.

**Pozor!** Píšeme „právě tehdy, když“, tedy se jedná o (logickou) ekvivalenci. Je třeba dokázat, že když $R \circ R \subseteq R$, pak $R$ je transitivní, a že když je $R$ transitivní, pak $R \circ R \subseteq R$.

## 2.5 Zobrazení

Druhým ze tří zvláště užitečných typů relace je tzv. *zobrazení*, které spíš znáte pod pojmem *funkce*. Narozdíl od ekvivalence, zobrazení budeme uvažovat jak na množině, tak mezi množinami.

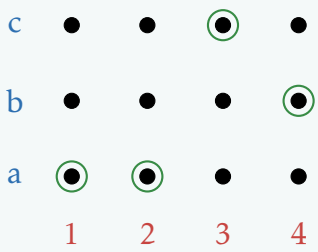
Definující vlastností funkce/zobrazení je fakt, že každý prvek nezobrazí buď na nic (pokud v něm „není definováno“) nebo na jeden jiný prvek. V jazyce relací to znamená, že každý prvek z množiny „nalevo“ je v relaci s maximálně jedním prvkem „napravo“.

**Definice 2.5.1** (Zobrazení)**.** Relaci $R$ mezi množinami $A$ a $B$ nazveme **zobrazením**, pokud pro každé $x \in A$ existuje **nejvýše jedno** $y \in B$ takové, že $xRy$.

**Příklad.** Mezi množinami $A := \{1,2,3,4\}$ a $B := \{a,b,c\}$ uvažme zobrazení

$$R := \{(1,a),(2,a),(3,c),(4,b)\} \subseteq A \times B.$$

Jeho mříž vypadá následovně.




Obrázek 10: Mříž zobrazení $R \subseteq A \times B$.

Fakt, že relace je zobrazení poznáte z její mříže velmi snadno tak, že (za předpokladu, že prvky levé množiny píšete vždy dole) v každém sloupci je **maximálně** jeden zelený kroužek.

Jelikož lidé přemýšleli o zobrazeních dříve než o relacích, je jejich zápis a názvosloví dost odlišné (a dost zmatené). Budeme je v dalších textu pravidelně užívat, takže vás s ním chca nechca musíme seznámit.

Pro zápis zobrazení se obvykle používají malá písmena latinské abecedy počínaje $f$ (pro **f**unction) nebo malá písmena řecké abecedy počínaje $\varphi$ (čteno „fí“, opět pro **f**unction). Fakt, že relace $f \subseteq A \times B$ je zobrazení mezi $A$ a $B$ (též říkáme „z $A$ do $B$“), zapisujeme obvykle jako

$$f : A \to B \quad \text{nebo} \quad A \xrightarrow{f} B.$$

Několik dalších názvů:

- Fakt, že $x f y$ pro $x \in A$ a $y \in B$, zapisujeme jako $f(x) = y$ nebo jako $f : x \mapsto y$. Prvku $y$ říkáme **obraz** prvku $x$ **při zobrazení** $f$. **Obrazem zobrazení** $f$ pak myslíme množinu všech obrazů prvků z $A$ a značíme ji $\operatorname{im} f$ (z angl. **im**age). Konkrétně,
$$\operatorname{im} f := \{f(x) \mid x \in A\}.$$
- Pro dané $y \in B$ značíme množinu všech $x \in A$ takových, že $f(x) = y$, jako $f^{-1}(y)$ a říkáme jí **vzor** prvku $y$ **při zobrazení** $f$. Čili, $x \in f^{-1}(y)$ vyjadřuje fakt, že $f(x) = y$.
- Pokud $f : A \to B$, množině $A$ říkáme **doména zobrazení** $f$ a množině $B$ **kodoména zobrazení** $f$.

**Výstraha.** Vzor prvku $y \in B$ při zobrazení $f$ je **množina**. Definice zobrazení mi říká jenom, že jedno $x \in A$ se zobrazí na jedno $y \in B$. To ale nebrání tomu, aby se víc různých prvků z $A$ zobrazilo na **ten samý** prvek z $B$. Naopak, množina $f^{-1}(y)$ může být i prázdná, pokud se na $y$ nezobrazuje žádný prvek z $A$.

**Příklad** (Kvadratická funkce)**.** Kvadratická funkce daná předpisem

$$f(x) := x^2 + 4x + 5$$

je zobrazení $\mathbb{R} \to \mathbb{R}$, čili jeho **doménou** i **kodoménou** jsou reálná čísla. **Obrazem** prvku 3 je $f(3) = 26$, ale **vzorem** prvku 26 je množina $\{-7, 3\}$. Dále třeba vzorem prvku 0 je prázdná množina, což je totéž, co říci, že rovnice

$$x^2 + 4x + 5 = 0$$

nemá v $\mathbb{R}$ řešení. Tradiční zápis $f$ jako relace by vypadal

$$f = \{(x, x^2 + 4x + 5) \mid x \in \mathbb{R}\} \subseteq \mathbb{R} \times \mathbb{R}.$$

Bohužel následuje ještě poslední kus názvosloví, protože pro určité „zajímavé“ typy zobrazení máme zvláštní názvy.

**Definice 2.5.2.** Zobrazení $f : A \to B$ nazveme

- **prosté** (nebo **injektivní**), pokud se každé dva *různé* prvky v $A$ zobrazují na dva *různé* prvky v $B$. Formálně, zobrazení $f$ je prosté, když
$$f(x) = f(x') \Rightarrow x = x' \quad \forall x, x' \in A.$$
Ještě jinak řečeno, zobrazení je prosté, když vzorem každého prvku je buď prázdná nebo jednoprvková množina. Fakt, že $f$ je prosté, často zapisujeme jako $f : A \hookrightarrow B$.
- **na** (nebo **surjektivní**), když má každý prvek z $B$ nějaký vzor v $A$. Formálně, zobrazení je na, když
$$\forall y \in B \, \exists x \in A : f(x) = y.$$
Ještě jinak řečeno, zobrazení je na, když je vzor každého prvku neprázdná množina. Fakt, že $f$ je na, často symbolicky zapisujeme jako $f : A \twoheadrightarrow B$.
- **vzájemně jednoznačné** (nebo **bijektivní**), když je *prosté* a *na*, čili vzorem každého prvku je přesně jednoprvková množina. Fakt, že $f$ je bijekce, často zapisujeme jako $f : A \leftrightarrow B$ nebo $f : A \cong B$.

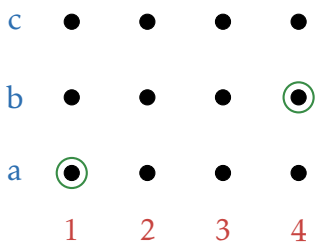
**Příklad.**

- Zobrazení $f : \mathbb{R} \leftrightarrow \mathbb{R}, x \mapsto 2x + 3$ je **bijektivní**. Obecně, každá lineární funkce je bijektivní zobrazení. Důkaz je ponechán jako cvičení.
- Zobrazení $f : \mathbb{R} \hookrightarrow \mathbb{R}, x \mapsto 3/x$ je **prosté**, ale není na. To proto, že $f^{-1}(0) = \emptyset$.
- Zobrazení $f : \mathbb{R} \twoheadrightarrow \mathbb{R}, x \mapsto (x-2)(x-3)(x+1)$ je **na**, ale není prosté.

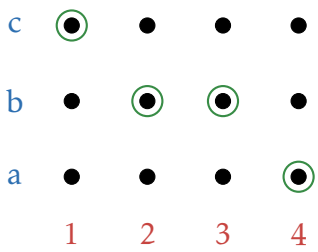
Třeba $f^{-1}(0) = \{-1, 2, 3\}$.

- Zobrazení $f : \mathbb{R} \to \mathbb{R}, x \mapsto 1 + 2/(x^2 - 1)$ není ani prosté, ani na. Například $f^{-1}(5/3) = \{-2, 2\}$ a $f^{-1}(1) = \emptyset$.

Prostá, surjektivní i bijektivní zobrazení mezi konečnými množinami z jejich mříží poznáte velmi snadno. Prostá zobrazení mají v řádcích maximálně jeden prvek; surjektivní zobrazení mají v každém řádku aspoň jeden prvek; ta bijektivní mají v každém řádku přesně jeden prvek. Pár obrázků.




Obrázek 11: Mříž **prostého** zobrazení $f := \{(1, a), (4, b)\}$.




Obrázek 12: Mříž **surjektivního** zobrazení $f := \{(1,c),(2,b),(3,b),(4,a)\}$.

**Bijekce mezi množinami, které mají různý počet prvků existovat nemůže**. Důkaz si zkusíte za cvičení. Částečně ho ale dává následující slibovaná definice velikosti množiny pomocí bijektivních zobrazení.

**Definice 2.5.3** (Velikost množiny pořádně)**.** Pro přirozené číslo $n \in \mathbb{N}$ označíme symbolem $[n]$ množinu všech přirozených čísel od 1 až do $n$ včetně. Čili,

$$[n] := \{1, 2, \ldots, n\}.$$

Množinu $A$ nazveme **konečnou**, pokud existuje přirozené číslo $n \in \mathbb{N}$ a bijekce $f : [n] \cong A$. V takovém případě číslu $n$ říkáme **velikost** množiny $A$ a značíme $\#A := n$.

Bijekci $f : [n] \cong A$ z definice nahoře můžeme vnímat jako „očíslování“ prvků množiny $A$ čísly od 1 do $n$. Takových očíslování je samozřejmě mnoho. Kolik?

**Příklad.** Množina $B := \{a, b, c\}$ má tři prvky. Jedna z možných bijekcí $f : [3] \cong B$ je

$$f := \{(1,c),(2,a),(3,b)\}.$$

Posledním důležitým konceptem je pojem *inverzního zobrazení*. Intuitivně, a vlastně i formálně, inverzní zobrazení je zobrazení, které jde opačným směrem a obrazy posílá zpátky na vzory. Toto samozřejmě vyžaduje například, aby vzor byl vždy nejvýše jeden. Detaily si rozmyslíte jako cvičení.

**Definice 2.5.4** (Inverzní zobrazení)**.** Nechť $f : A \to B$ je zobrazení. **Inverzním zobrazením** k $f$, značeným dost nevhodně $f^{-1}$, nazveme zob-

razení $B \to A$ splňující

$$f^{-1}(f(x)) = x \wedge f(f^{-1}(y)) = y \quad \forall x \in A, y \in \operatorname{im} f.$$

Pozor! Inverzní zobrazení *nemusí existovat*.

**Výstraha.** Pokud k $f : A \to B$ existuje inverzní zobrazení, značí $f^{-1}(y)$ jak množinu vzorů prvku $y \in B$, tak obraz prvku $y$ při inverzním zobrazení.

Toto však je problém pouze formální. Pokud totiž existuje inverzní zobrazení, pak má množina $f^{-1}(y)$ buď jeden prvek, nebo žádný. V prvním případě tedy akorát ztotožňuji jednoprvkovou množinu s jejím jediným prvkem. To je totéž, co považovat třeba množinu $\{2\}$ a číslo 2 za to samé. Vskutku, problém pouze formální, bez praktických důsledků.

Sekci završíme ku radosti všech párem cvičení.

**Cvičení 2.5.1.** Vyřešte následující úlohy rozprostřené po sekci. Konkrétně,

- dokažte, že mezi dvěma konečnými množinami různé velikosti neexistuje žádná bijekce.
- pro množinu $A$ velikosti $n$ určete počet různých bijektivních zobrazení $f : [n] \cong A$.
- určete, jakou podmínku splňují zobrazení $f : A \to B$, ke kterým existuje zobrazení inverzní.

**Cvičení 2.5.2.** Dokažte, že každá lineární funkce $f : \mathbb{R} \to \mathbb{R}$, tedy funkce daná předpisem

$$f(x) = ax + b \quad \text{pro } a, b \in \mathbb{R}, a \neq 0$$

je bijektivní zobrazení.

**Cvičení 2.5.3.** Nechť $A$ je konečná množina. Zformulujte důkaz, že zobrazení $f : A \to A$, které je definované pro každé $x \in A$, je **prosté, právě tehdy když je na**.

Tento fakt se často též používá v teorii množin jako definice konečné množiny. Je hezčí než naše v tom, že nespoléhá na množinu přirozených čísel. Tedy, množinu $A$ nazvu *konečnou*, když každé zobrazení $f : A \to A$ definované všude je prosté, právě tehdy když je na.

**Cvičení 2.5.4.** Najděte příklad zobrazení $f : \mathbb{N} \to \mathbb{N}$ definovaného na celém $\mathbb{N}$, které je

(1) prosté, ale není na.

(2) na, ale není prosté.

## 2.6 Uspořádání

Uspořádání je poslední v jistém smyslu speciální relací, na kterou se podíváme. Podobně jako ekvivalence, uspořádání mezi dvěma množinami nedává úplně smysl, takže se po celou sekci budeme soustředit na relaci na množině.

Určitě nejznámějším typem uspořádání je relace „menší nebo rovno“ (nebo „menší“, „větší nebo rovno“ atd., to je jedno), kterou jistě všichni známe. Tohle je ten příklad uspořádání, který doporučujeme mít na paměti, kdykoli se zdají obecné definice těžko stravitelnými.

Existuje ale samozřejmě spousta jiných druhů uspořádání s rozlišnými spektry užitku. Uveďme například uspořádání dělitelností, zcela zásadní v elementární teorii čísel, nebo lexikografické uspořádání, kterým se řadí slova ve slovnících a encyklopediích a dá se použít i pro řazení polynomů (například v důkazu slavného Gaussova algoritmu).

Ještě před definicí uspořádání se ale musíme zmínit o pro ni klíčové vlastnosti relací.

**Definice 2.6.1** (Antisymetrická relace)**.** Relace $R$ na množině $A$ se nazývá

- **antisymetrická**, pokud $(x, y) \in R \Rightarrow (y, x) \notin R$ pro všechny prvky $x, y \in A$.
- **slabě antisymetrická**, pokud $xRy \wedge yRx \Rightarrow x = y$ pro všechna $x, y \in A$.

Jak název napovídá, vlastnost antisymetrie je opravdu jakýmsi protikladem symetrie.

Přeložena do jazyka aspoň některých lidí, relace je (silně) antisymetrická tehdy, když to, že je prvek $x$ v relaci s prvkem $y$, zakazuje, aby byl zároveň $y$ v relaci s $x$. Může se však samozřejmě stát, že $x$ není v relaci s $y$ ani $y$ není v relaci s $x$. Všimněte si, že vlastnost antisymetrie mimo jiné *nedovoluje*, *aby daná relace byla reflexivní*.

**Příklad.** Všeobecně oblíbených příkladem (silně) antisymetrické relace je relace $<$, třeba na množině $\mathbb{R}$. V moment, kdy pro dvě reálná čísla $x, y \in \mathbb{R}$ platí, že $x < y$, pak automaticky **nemůže platit** $y < x$. Zároveň, žádné reálné číslo není nikdy ostře menší než ono samo, tedy $<$ je vskutku antisymetrická a není reflexivní.

Ačkoliv to možná z definice není zřejmé, vlastnost *slabé* antisymetrie je opravdu jen oslabená vlastnost (silné) antisymetrie v tom smyslu, že slabě antisymetrická relace může být reflexivní. Čili, mám-li slabě antisymetrickou relaci $R$, pak $xRy$ nutně **nezakazuje**, aby $yRx$, ale jediný prvek $y$, pro který tato situace může nastat, je $x$ samotné.

**Příklad.** Asi tušíte, co přijde. Když vám řekneme, abyste zeslabili vztah $<$, prvním takovým přirozeným nápadem je vztah $\leq$, což je skutečně slabě antisymetrická relace. Vskutku, když $x \leq y$, pak se může stát, že i $y \leq x$, ale to nutně znamená, že $x = y$.

**Definice 2.6.2** (Uspořádání)**.** Relace $R$ na množině $A$ se nazývá

- (neostré) **uspořádání**, pokud je *reflexivní*, *slabě antisymetrická* a *transitivní*.
- **ostré uspořádání**, pokud je *antisymetrická* a *transitivní*.

Pokud je $R$ (ostré) uspořádání na $A$, nazýváme dvojici $(A, R)$ (ostře) **uspořádanou množinou**.

**Příklad.**

(1) Relace $<$ na $\mathbb{R}$ je **ostré uspořádání**, protože
   - (antisymetrie) $x < y \Rightarrow y \not< x$ a
   - (transitivita) $x < y \wedge y < z \Rightarrow x < z$

   pro všechna čísla $x, y, z \in \mathbb{R}$.

(2) Relace $\leq$ na $\mathbb{R}$ je (neostré) **uspořádání**. Vskutku, platí
   - (reflexivita) $x \leq x$,

- (slabá antisymetrie) $x \leq y \wedge y \leq x \Rightarrow x = y$ a
- (transitivita) $x \leq y \wedge y \leq z \Rightarrow x \leq z$

pro všechna $x, y, z \in \mathbb{R}$.

Ještě poslední sousto nomenklatury.

**Definice 2.6.3** (Lineární uspořádání)**.** (Ostré) uspořádání $R$ na množině $A$ nazveme **lineárním**, pokud pro každé dva prvky $x, y \in A$ platí, že $xRy$ nebo $yRx$. Čili, každé dva prvky $A$ spolu *lze porovnat* prostřednictvím $R$. (Ostré) uspořádání, které *není lineární*, často označujeme jako **částečné**.

**Příklad.** Jak $<$, tak $\leq$, jsou lineární uspořádání.

Protože základní idea za pojmem „uspořádání“ je, no, uspořádání prvků na množině, ujaly se pro jejich kreslení, spíše než mříže nebo šipky, tzv. *Hasseho diagramy*. Hasseho diagram vypadá tak, že prvky množiny jsou značeny tečkami a mezi porovnatelnými prvky (tj. prvky, které jsou v relaci) se dá dostat po úsečkách (někdy přes více prvků). Navíc, prvky se kreslí zezdola nahoru vzhledem k jejich pozici v rámci daného uspořádání.

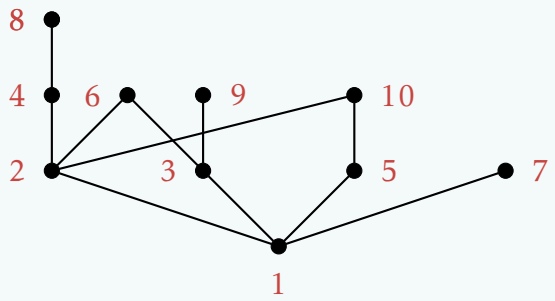
**Příklad** (Uspořádání velikostí)**.** Již jsme zpozorovali, že $\leq$ je uspořádání. Jeho Hasseho diagram na množině $A := \{1, 2, 3, 4, 5\}$ vidíte na obrázku 13.


Obrázek 13: Hasseho diagram uspořádáné množiny $(A, \leq)$.

Ve skutečnosti, Hasseho diagram **každého** lineárního uspořádání vypadá takto; liší se pouze počet prvků. Detaily si rozmyslíte za cvičení.

**Definice 2.6.4** (Dělitelnost)**.** Mějme čísla $m, n \in \mathbb{N}$. Říkáme, že $m$ **dělí** $n$, když existuje přirozené číslo $k \in \mathbb{N}$ takové, že $n = km$. Tento fakt zapisujeme jako $m \mid n$.

**Příklad** (Uspořádání dělitelností)**.** Relace $\mid$ z definice dělitelnosti je ve skutečnosti uspořádání (důkaz jako cvičení), které **ale není lineární**. Jeho Hasseho diagram na množině $A := [10]$ je výrazně košatější.




Obrázek 14: Hasseho diagram uspořádané množiny $(A, \mid)$.

Lexikografické uspořádání je v principu uspořádání na slovech, ale může být úspěšně použito třeba i pro uspořádání polynomů více proměnných. Funguje následovně: slovem délky $n$ nazveme posloupnost $a_1 a_2 \dots a_n$, kde $a_i$ je libovolné písmeno mezi „a“ a „z“. Slovo $a_1 a_2 \dots a_n$ je lexikograficky níž než slovo $b_1 b_2 \dots b_m$, pokud existuje $i \leq \min(n, m)$ takové, že $a_1 = b_1 \wedge a_2 = b_2 \wedge \dots \wedge a_{i-1} = b_{i-1}$ a $b_i > a_i$.

Řečeno lidsky, o slově $a_1 a_2 \dots a_n$ řeknu, že je níž než $b_1 b_2 \dots b_m$, když nějaké jeho písmeno $a_i$ je dřív v abecedě než písmeno $b_i$ na stejném místě ve slově $b_1 b_2 \dots b_m$. Pokud je jedno slovo plně součástí druhého, lexikograficky níž je to kratší. Lexikografické uspořádání se obvykle značí rovněž $\leq$, ale pro přehlednost ho budeme značit třeba $\leq_{\text{lex}}$.

**Příklad** (Lexikografické uspořádání)**.** Jak si můžete ověřit (ale cvičení to nutně není), lexikografické uspořádání je lineární, takže jeho Has-

seho diagram není dvakrát zajímavý. Pro úplnost zde ale přesto ukážeme diagram množiny

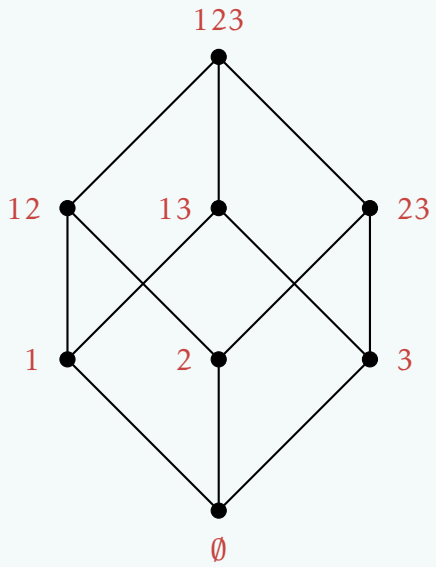
$$A := \{a_1 a_2 \mid a_i \in \{a, b, c\}\},$$

tedy množiny všech dvojpísmenných kombinací písmen „a“ až „c“.


Obrázek 15: Hasseho diagram uspořádané množiny $(A, \leq_{\text{lex}})$.

Velmi pěkné obrázky uspořádání vznikají i na množině všech množin $A$, tedy na $2^A$, kde uspořádání je inkluzí $\subseteq$. Tedy, nejníž je prázdná množina, která leží uvnitř každé množiny, a nejvýš je množina $A$, ve které je obsažena každá její podmnožina. Jeden malý příklad tu nakreslíme, určitě se vám bude líbit.

**Příklad.** Mějme množinu $A := \{1, 2, 3\}$. Množinu $2^A$ uspořádáme inkluzí, čili dvě podmnožiny $A$ jsou v relaci inkluze, když je jedna obsažena v druhé. Jeden příklad za všechny je třeba $\{1\} \subseteq \{1, 3\}$. Hasseho diagram takového uspořádání vidíte níže. V rámci úspory píšeme třeba 12 místo množiny $\{1, 2\}$.




Obrázek 16: Hasseho diagram uspořádané množiny $(2^A, \subseteq)$.

Jako obvykle následuje několik cvičení na závěr sekce.

**Cvičení 2.6.1.** Udělejte cvičení rozmístěná po sekci. Konkrétně,

(1) dokažte, že Hasseho diagram každého lineárního uspořádání má stejný tvar jako diagram na obrázku 13.

(2) dokažte, že relace dělitelnosti $|$ je uspořádání na každé podmnožině přirozených čísel.

**Cvičení 2.6.2.** Explicitně popište všechny relace (na libovolné množině), které jsou zároveň ekvivalencí a (částečným) uspořádáním.

**Cvičení 2.6.3.** Řekněme, že $R$ a $S$ jsou uspořádání na množině $A$. Které z následujících relací jsou také uspořádáními na $A$?

- $R \cap S$
- $R \cup S$
- $R \setminus S$
- $R \circ S$

## 2.7 Matematická indukce

Indukce je základní důkazovou technikou v diskrétní matematice. Je to jeden možný, ale zcela jistě nejoblíbenější, způsob, jak dokazovat libovolná tvrzení o přirozených číslech, která jsou vlastně právě tím číselným oborem, který studuje diskrétní matematika.

Princip indukce spočívá v tom, že přirozená čísla jsou definována v zásadě velmi jednoduše. Libovolná množina, která má nějaký „základní prvek“ (třeba jedničku) a spolu s každým prvkem má i jeho bezprostředního následníka (třeba to číslo o jedna větší), je automaticky „ta samá množina“ jako přirozená čísla.

Pokud byste měli chuť se podívat na formální definici přirozených čísel a dalších souvisejících věcí, doporučujeme vyhledat klíčová slova *Peanova aritmetika*, která je vlastně (možná kecám, ale myslím, že nejmenším možným) systémem axiomů (kategoricky platných výroků), jenž buduje ryze logický základ pro aritmetiku.

My si ale vystačíme s následujícím zjednodušením.

**Tvrzení 2.7.1** (Definice přirozených čísel)**.** Nechť $A$ je množina, která splňuje, že

- $1 \in A$,
- je-li $n \in A$, pak rovněž $n+1 \in A$.

Potom $A = \mathbb{N}$.

**Důkaz.** Nedokazuje se, je to axiom (konkrétně pátý) Peanovy aritmetiky. □

Žádáme, abyste si dali chvíli a zamysleli nad významem tvrzení. Zevrubně řečeno říká, že, pokud umím dokázat, že

- tvrzení platí pro první přirozené číslo a
- za předpokladu, že tvrzení platí pro $n$, platí pro $n+1$,

pak dané tvrzení platí pro všechna přirozená čísla. Tyhle dva důkazy totiž

dohromady dávají následující (nekonečný) řetězec důkazů:

(1) (Nějaké) tvrzení platí pro $n = 1$.

(2) Jestliže tvrzení platí pro $n = 1$, pak platí pro $n = 2$.

(3) Jestliže tvrzení platí pro $n = 2$, pak platí pro $n = 3$.

$$\vdots$$

Princip indukce asi není přehnaně složitý, ale získat dostatek zkušenosti, aby jej člověk uměl neomylně aplikovat, je výrazně obtížnější. Pár příkladů snad s tímto krokem pomůže. Budeme je záměrně formulovat jako lemmata či tvrzení, jelikož indukce je v prvé řadě důkazová technika. Doporučujeme, abyste důkazy četli se zvýšenou pozorností.

**Lemma 2.7.2.** Pro každé $n \in \mathbb{N}$ platí

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$$

**Důkaz.** Dokazujeme indukcí. Protože suma začíná od 0, je prvním prvkem, pro který musí tvrzení platit, v tomto případě právě $n = 0$. Dosazením zjistíme, že

$$\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^{0+1} - 1$$

tedy tvrzení platí pro $n = 0$. Předpokládáme, že tvrzení platí pro všechna přirozená čísla až do nějakého $n \in \mathbb{N}$ a z tohoto předpokladu odvodíme, že platí i pro $n + 1$. Počítáme

$$\sum_{i=0}^{n+1} 2^i = \sum_{i=0}^{n} 2^i + 2^{n+1}.$$

Ovšem, z předpokladu dostaneme

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1,$$

což dohromady s předchozím výpočtem dává

$$\sum_{i=0}^{n+1} 2^i = \sum_{i=0}^{n} 2^i + 2^{n+1} = 2^{n+1} - 1 + 2^{n+1} = 2 \cdot 2^{n+1} - 1 = 2^{n+2} - 1,$$

jak jsme chtěli ukázat. Důkaz je podle principu indukce ukončen. □

**Lemma 2.7.3.** Pro všechna $n \in \mathbb{N}$ platí, že

$$3 \mid n \Rightarrow 3 \mid n^2,$$

tedy, pokud 3 dělí $n$, pak 3 dělí $n^2$.

Tady by se jistě leckdo rád odvolal třeba na prvočíselné rozklady. Je ale dobré si uvědomit, že fakt, že každé přirozené číslo má jednoznačný rozklad na prvočísla, není samozřejmý. Ve skutečnosti zabere určitou práci toto dokázat. Či vy jste viděli nějaký přímočarý důkaz, že jdou čísla rozkládat na prvočísla? Opravdu lze *každé* číslo rozložit na prvočísla a opravdu to lze *pouze jediným způsobem*?

**Důkaz** (lemmatu 2.7.3)**.** Dokazujeme indukcí.

První přirozené číslo, pro které má smysl tvrzení dokázat, je $n = 3$. Pak vskutku $3 \mid n = 3$ a $3 \mid n^2 = 9$.

Předpokládejme, že výrok $3 \mid n \Rightarrow 3 \mid n^2$ platí pro nějaké $n \in \mathbb{N}$. Nejbližší další přirozené číslo po $n$, které je dělitelné 3, je $n+3$. Tedy, víme, že když $3 \mid n$, pak $3 \mid n+3$ a také $3 \mid n^2$. Z těchto dvou faktů odvodíme, že $3 \mid (n+3)^2$.

Máme $(n+3)^2 = n^2 + 6n + 9$. Protože $3 \mid 6$ a $3 \mid 9$, také $3 \mid 6n + 9$. Předpokládáme, že $3 \mid n^2$, dohromady tudíž $3 \mid n^2 + 6n + 9 = (n+3)^2$, jak jsme chtěli. □

Tři cvičení nakonec.

**Cvičení 2.7.1.** Dokažte indukcí, že

$$\sum_{i=1}^{n} i2^i = (n-1)2^{n+1} + 2.$$

**Cvičení 2.7.2** (Fibonacciho čísla a zlatý řez)**.** Tak zvaná *Fibonacciho* posloupnost je definována tak, že další člen dostanu jako součet dvou předchozích. Formálně, $F_0 = 0, F_1 = 1$ a $F_n = F_{n-1} + F_{n-2}$, kde $n \in \mathbb{N}$. Dokažte indukcí, že

$$F_n \leq \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}$$

pro všechna $n \geq 0$.

Číslu $(1+\sqrt{5})/2$ se někdy říká hodnota „zlatého řezu“ (protože je to v jistém smyslu *ideální* poměr mezi délkami dvěma bezprostředních úseček – internet poví víc). Jestli si někdy ukážeme limity, pak dokážeme tento výsledek zdokonalit v tom smyslu, že platí

$$\frac{F_n}{F_{n-1}} \xrightarrow{n\to\infty} \frac{1+\sqrt{5}}{2}.$$

**Cvičení 2.7.3.** Nakresleme $n$ přímek v rovině, a to tak, že

- žádné 2 nejsou rovnoběžné a
- žádné 3 se neprotínají v jednom bodě.

Dokažte indukcí, že takhle nakreslené přímky rozdělují rovinu na $n(n+1)/2+1$ částí.

# 3 | Počítání

V této kapitole se naučíme počítat; a ne, doteď jste to neuměli. Snad všechny potěšíme, když zvěstíme, že tahle je kapitola je již skutečným úvodem do problematiky *diskrétní matematiky*. Otázky typů „Kolik je čeho?“, „Kolika způsoby mohu něco udělat?“ skutečně nikam jinam patřit ani nemohou, protože analytici mají všeho nespočetně mnoho a lineárním algebraikům zas může vadit, že nad přirozenými čísly se žádná rozumná geometrie úplně dělat nedá.

Začneme snad jednoduchým počítáním daných typů zobrazení a podmnožin, poté se posuneme k počtu možností, jak za sebe skládat prvky. V neposlední řadě se budeme věnovat tzv. *principu inkluze a exkluze*, jenž umožňuje elegantně odpovídat třeba na otázky „Kolik je čísel menších než 100, které nejsou dělitelné 2 ani 3?“. Vše završíme notoricky známým *problémem šatnářky*, o kterém raději nic neprozradíme, bychom si udrželi alespoň přirozené číslo čtenářů.

Ty nejzákladnější způsoby, jak určovat počty věcí nebo způsobů, jak něco dělat, jsou obecně dva:

- **přímý** aneb „Vím, co dělám, a umím to spočítat pro libovolné přirozené číslo.“ a
- **indukcí** aneb „Vůbec to nechápu, ale zkusím si to pro pár malejch čísel a pak to nějak ukoulím i pro ty velký.“

Ačkoli by to kolegové z katedry kombinatoriky neradi slyšeli, druhý způsob je zcela jistě ten bohatě nejoblíbenější.

V trochu serióznějším duchu radíme vždy zkusit nejprve přímý důkaz, u kterého je zřejmé, jak jste na vzorec přišli a proč je správný. Důkaz indukcí je totiž z principu *nekonstruktivní*, tj. není z něj vůbec jasné, odkud se vzorec bere. Stačí totiž jen ukázat, že funguje pro jakési první číslo a že když funguje pro nějaké číslo, pak funguje i pro to další. Takový důkaz ale neposkytuje vůbec žádný vhled do problému.

## 3.1 Zobrazení a podmnožiny

Chvíli se budeme bavit počítáním zobrazení a podmnožin obvykle určených nějakou hezkou podmínkou. Začít právě tady je vhodné z páru důvodů. Zaprvé, není potřeba vymýšlet žádnou novou teorii a zadruhé – snad trochu překvapivě – umět počítat zobrazení a podmnožiny se hodí do spousty dalších matematických disciplín. Zmiňme Čínskou větu o zbytcích, v podstatě jeden ze základních stavebních kamenů teorie čísel, jejíž důkaz je založen právě na tom, že umíme počítat zobrazení mezi množinami. Dále je tu třeba Burnsideova věta z abstraktní algebry, na jejíž pravdivost spoléhá třeba otáčení obsahu obrazovky na mobilech a jejíž důkaz vyžaduje porovnávání velikostí systémů podmnožin. Konečně, patří sem i latinské čtverce – struktury, jejichž princip stojí za vznikem Sudoku.

Pojďme začít tím nejjednodušším možným tvrzením, tedy o počtu všech zobrazení mezi množinami. Ukážeme si dva důkazy: jeden přímý a jeden indukcí.

**Výstraha.** Pro stručnost budu v celé kapitole slovem zobrazení myslet **zobrazení definované všude**. Diskrétní matematiku totiž pravdaťúplně netrápí problémy definičních oborů, takže není žádná výhoda v tom uvažovat zobrazení, která nejsou definována pro všechny prvky svých domén.

**Tvrzení 3.1.1** (Počet všech zobrazení)**.** Mějme konečné množiny $A$ a $B$. Počet všech zobrazení $A \to B$ je $\#B^{\#A}$.

**Důkaz** (tvrzení 3.1.1 přímo)**.** Rozmysleme si nejprve, kdy se dvě zobrazení $f, g : A \to B$ liší. To je přeci tehdy, když existuje nějaký prvek $a \in A$ takový, že $f(a) \neq g(a)$.

Jinak řečeno, každé zobrazení $A \to B$ popíšu tak, že určím obrazy všech prvků z $A$. Kdykoli mám dvě zobrazení, jejichž obraz byť i jednoho prvku z $A$ se neshoduje, pak jsou to různá zobrazení. Pro každý prvek z $A$ mám přesně $\#B$ prvků, na které ho mohu zobrazit, tedy mám celkem přesně $\#B^{\#A}$ možností, jak zobrazit všechny prvky z $A$ na prvky z $B$. □

**Důkaz** (tvrzení 3.1.1 indukcí)**.** Dokážeme předchozí tvrzení užitím indukce podle velikosti množiny $A$.

Když je $A$ prázdná, čili $\#A = 0$, pak mám právě jedno zobrazení $A \to B$ – to, které nezobrazuje nic na nic. Čili mám vskutku $\#B^{\#A} = \#B^0 = 1$ různých zobrazení $A \to B$.

Předpokládejme, že platí, že zobrazení z $A$ do $B$ je právě $\#B^{\#A}$ a přidejme do množiny $A$ jeden prvek, třeba $x$. Chceme ukázat, že všech zobrazení $A \cup \{x\} \to B$ je $\#B^{\#A+1}$. Jeden způsob, jak to udělat, je podívat se kolika způsoby můžeme zobrazení $A \to B$ „dodefinovat“ v $x$.

No, $x$ přeci mohu zobrazit na jakýkoliv prvek z $B$ a každá volba obrazu mi dává jiné zobrazení. Čili, z jednoho zobrazení $A \to B$ mi vznikne právě $\#B$ různých zobrazení $A \cup \{x\} \to B$. To ale znamená, že všech zobrazení $A \cup \{x\} \to B$ je $\#B$-krát víc než zobrazení $A \to B$. Tedy jich je podle předpokladu

$$\#B^{\#A}\#B = \#B^{\#A+1}.$$ □

O něco těžší je počítat zobrazení $A \to B$ omezených vlastností. Samozřejmě bychom si mohli navymýšlet libovolné podmínky, které naše zobrazení musí splňovat; třeba, že musí na každý prvek $B$ zobrazit právě prvočíselný počet prvků z $A$. Zjistit počet všech takových zobrazení by jistě byla zajímavá úloha, ale asi ne příliš užitečná. Pojďme se soustředit na více obvyklé typy zobrazení.

**Tvrzení 3.1.2** (Počet prostých zobrazení)**.** Počet všech **prostých** zobrazení $A \to B$ je

$$\prod_{i=0}^{\#A-1} \#B - i,$$

**Důkaz.** Předvedeme přímý důkaz. Důkaz indukcí si zkusíte za cvičení.

Princip důkazu je podobný jako při počítání všech zobrazení $A \to B$. Zásadní rozdíl dlí v tom, že na každý prvek z $B$ lze zobrazit maximálně jeden prvek z $A$. Opět ale platí, že dva různé výběry obrazů prvků z $A$ nám dávají dvě různá zobrazení. Stačí tedy spočítat, kolika způsoby si můžeme zvolit, kam se prvky $A$ zobrazí.

Nu, první prvek z $A$ můžeme zobrazit na $\#B$ různých prvků z $B$. Pro ten druhý ovšem máme už jen $\#B-1$ možností, protože zobrazení musí být **prosté**, a tedy nelze druhý prvek zobrazit tam, kam ten první. Tenhle princip se opakuje. Pro třetí prvek už máme jen $\#B-2$ možných obrazů atd. Celkem, pro $i$-tý prvek z $A$ máme jen $\#B - i + 1$ míst, kam ho zobrazit.

Shrnuto, pro každý výběr obrazu prvního prvku máme už jen $\#B - 1$ možných obrazů pro druhý prvek. Pro každý výběr obrazů prvních dvou prvků máme už jen $\#B - 2$ možných obrazů pro třetí prvek. Takhle pokračujeme, dokud nedojdeme až k $\#A$-tému prvku, pro který nám zbývá $\#B-\#A+1$ nevyužitých prvků $B$. Sepsáno symbolicky, máme

$$\#B(\#B-1)(\#B-2)\cdots(\#B-\#A+1) = \prod_{i=0}^{\#A-1} \#B - i$$

možností, jak zvolit obrazy všech prvků z $A$ za daných podmínek. Tedy existuje právě tolik prostých zobrazení $A \to B$. □

Na konec sekce si ještě spočítáme nějaké podmnožiny. Už víme, že počet všech podmnožin $A$ je $2^{\#A}$. Je to Tvrzení 2.2.2. Asi nejjednodušší další úlohou je počet všech podmnožin liché a sudé velikosti. Čék by si řek', že jich je fifty-fifty a měl by recht. Ukážeme si důkaz.

**Tvrzení 3.1.3** (Počet podmnožin liché velikosti)**.** Všech podmnožin liché velikosti konečné množiny $A$ je $2^{\#A-1}$.

**Důkaz.** Půjdeme na to trochu jinak. Víme ze sekce o zobrazeních, že mezi konečnými množinami existuje bijekce jenom tehdy, když jsou stejně velké. Vyjměme z $A$ nějaký fixní prvek, třeba $a \in A$. Množinu $A \setminus \{a\}$ označíme $\tilde{A}$. Protože $\tilde{A}$ má $\#A - 1$ prvků, počet jejích podmnožin je $2^{\#A-1}$. Najdeme bijekci mezi všemi podmnožinami množiny $\tilde{A}$ a lichými podmnožinami množiny $A$.

Definujme zobrazení $f : 2^{\tilde{A}} \to 2^A$ následujícím způsobem. Pozor! Všimněte si, že zobrazení $f$ je definované na podmnožinách. Tedy zobrazuje množiny na množiny.

Každá lichá podmnožina $A$ buď obsahuje $a$, nebo je neobsahuje. Liché podmnožiny $A$, které obsahují $a$, jsou sudými podmnožinami $\tilde{A}$ (pro-

tože jsme $a$ odebrali), a ty, které $a$ neobsahují, zůstávají lichými i v $\tilde{A}$. Tedy, definujme

$$f(X) := \begin{cases} X, & \text{pokud } \#X \text{ je liché,} \\ X \cup \{a\}, & \text{pokud } \#X \text{ je sudé.} \end{cases}$$

pro každou podmnožinu $X \subseteq \tilde{A}$. Tím jsme sestrojili bijekci mezi všemi podmnožinami $\tilde{A}$ a lichými podmnožinami $A$. Odtud plyne, že lichých podmnožin $A$ je $2^{\#A-1} = 2^{\#A}/2$.

Pro sudé podmnožiny lze postupovat obdobně anebo si uvědomit, že všechny ostatní podmnožiny, které nejsou liché, musejí být sudé. Tedy jich je $2^{\#A} - 2^{\#A-1} = 2^{\#A-1}$. □

Předchozí důkaz ilustruje další běžný způsob, jak počítat prvky daných množin: konkrétně tak, že najdeme bijekci mezi množinou, jejíž počet prvků chceme spočítat, a množinou, jejíž počet prvků známe.

Dvě cvičení na závěr.

**Cvičení 3.1.1.** Dokažte Tvrzení 3.1.2 indukcí podle velikosti množiny $A$.

**Cvičení 3.1.2.** Určete počet všech uspořádaných dvojic $(A, B)$, kde $A \subseteq B \subseteq \{1, \ldots, n\}$.

*Uspořádaná dvojice* znamená, že $(A, B) \neq (B, A)$, tedy záleží na pořadí, v jakém podmnožiny zapíšu.

## 3.2 Permutace

Permutace jsou vlastně zobrazení, která prohazují prvky množin. Jejich asi hlavním účelem je formalizovat koncept, že „nezáleží na pořadí“ nebo naopak, že všechno dělám pro všechna možná přeuspořádání prvků. Člověk by měl dobrý důvod si myslet, že nejsou dobré k ničemu jinému, než ke zkrášlení zápisu. Opak je pravdou. Permutace mají velmi překvapivé aplikace v oblastech matematiky, kde by je jeden nehledal. Zmiňme tři příklady.

- Důkaz základní věty algebry – tvrzení, že každý komplexní polynom má komplexní kořen – silně využívá tzv. rozklad na symetrické polynomy, založený na vlastnostech permutací.
- Fakt, že kořeny obecných reálných (i komplexních) polynomů nelze zapsat v radikálech (tj. odmocninách), když je stupeň polynomu větší nebo roven 5 (tj. objevuje se v něm $x^5$), se opírá o tzv. „neřešitelnost“ permutačních grup (množin permutací na dané množině s binární operací skládání).
- Důkaz, že na každé Riemannově pseudovarietě dimenze 4 (kterou fyzikové používají jako model časoprostoru) existuje nekonečně mnoho neisomorfních Riemannových metrik (tj. v našem vesmíru mohu měřit vzdálenost nekonečně mnoha neekvivalentními způsoby) staví na symetrii tensorů definovaných pomocí permutací.

Takže, o co tu vlastně jde.

**Definice 3.2.1** (Permutace)**.** Bijekce $\sigma : X \to X$ konečné množiny $X$ na sebe samu se nazývá *permutace* množiny $X$.

Množinu všech permutací na $X$ značíme $S_X$ (jako grupa **s**ymetrií $X$).

Jelikož všichni rádi počítáme (xD), určíme si na začátek počet všech permutací na dané množině. Podle cvičení 2.5.3, které jste *všichni* dělali, zobrazení na konečné množině $X$ je prosté právě tehdy, když je na, tedy právě tehdy, když je bijektivní. Tento fakt nám pomůže s důkazem následujícího tvrzení. Jen ještě jedna definice usnadňující zápis.

**Definice 3.2.2** (Faktoriál)**.** Pro přirozené číslo $n \in \mathbb{N}$ definujeme

$$n! := \prod_{i=0}^{n-1} n - i.$$

Výraz $n!$ čteme *n faktoriál*.

**Tvrzení 3.2.3** (Počet permutací na množině)**.** Ať $X$ je konečná množina. Pak $\#S_X = (\#X)!$.

**Důkaz.** Podle tvrzení 3.1.2, počet prostých zobrazení $A \to B$ je

$$\prod_{i=0}^{\#A-1} \#B - i.$$

Když $A = B$, pak bijekce $A \to A$ jsou totéž, co prostá zobrazení $A \to A$. Tedy, všech bijekcí $A \to A$ (tj. všech permutací na $A$) je

$$\prod_{i=0}^{\#A-1} \#A - i = (\#A)!. \qquad \square$$

### 3.2.1 Zápis permutací

Budeme se chvíli bavit o tom, jak můžeme reprezentovat permutace. Samozřejmě, permutace jsou mimo jiné zobrazení, takže je lze kreslit, jak už jsme to dělali; tj. jako šipky mezi množinami teček.

Existují ale chytřejší a přehlednější způsoby, jak je znázornit. Jeden možný způsob je zápisem do řádku. Řekněme, že $X = \{1, 2, 3, 4\}$ a $\sigma \in S_X$. Když napíšeme, že

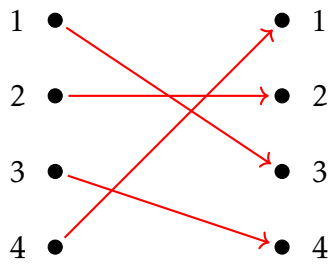
$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix},$$

myslíme tím, že $\sigma$ je zobrazení, které posílá 1 na 3, 2 na 2, 3 na 4 a 4 na 1. Můžeme navíc předpokládat, že vrchní řádek je vždycky v nějakém

předem dohodnutém pořadí a permutaci $\sigma$ zapsat prostě jako

$$\sigma = \begin{pmatrix} 3 & 2 & 4 & 1 \end{pmatrix}.$$

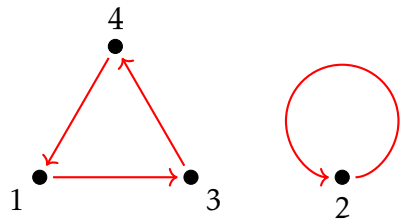
V obvyklém kreslení zobrazení bychom $\sigma$ znázornili, kterak vidíte na obrázku níže.




Obrázek 17: Permutace $\sigma$ zakreslená šipkami.

Ačkoli je způsob zápisu do řádku intuitivní a podporuje představu permutace jako „proházení“ prvků na množině, mnohem více se používá tzv. zápis v cyklech. Důvody jsou primárně formální; z cyklického zápisu se velmi snadno totiž pozná jak „řád“ permutace, tak její rozložení na „transpozice“. Oba pojmy definujeme a vysvětlíme později.

Jelikož permutace jsou bijekce z množiny do téže množiny, můžeme vždy začít v nějakém libovolné prvku a pokračovat po šipkách, dokud se nedostaneme opět na ten samý prvek. Tento přístup formalizuje právě zápis do cyklů. Například zápis permutace $\sigma = (3\ 2\ 4\ 1)$ do cyklů by vypadal takto:




Obrázek 18: Zápis permutace $\sigma$ do cyklů.

Jak si asi dovedete představit, tento zápis znamená, že permutace $\sigma$ pošle 1 na 3, pak 3 na 4 a nakonec 4 na 1. Tedy, po třech „iteracích“ permutace

$\sigma$ se dostaneme z prvku 1 opět do prvku 1. Smyčka nad 2 samozřejmě znamená, že 2 se zobrazuje opět na 2.

Zápis na obrázku výše je evidentně dosti neúsporný, a v textu tudíž těžko použitelný. Obvykle se taková permutace zapíše jako $\sigma = (134)(2)$. Tedy, jednotlivé cykly jsou odděleny závorkami a šipky v cyklech vedou zleva doprava, případně z posledního prvku zpět na první.

Konečně, smyčky (tj. zobrazení prvku na sebe sama) se z cyklického zápisu běžně vynechávají. Svoji oblíbenou permutaci $\sigma = (134)(2)$ můžeme proto úplně nejúsporněji zapsat jako $\sigma = (134)$. Zápis permutací v cyklech budeme odteď využívat výhradně.

**Výstraha.** Zápis permutace pomocí cyklů **není jednoznačný**. Je to proto, že u daného cyklu nelze říct, kde „končí“ a kde „začíná“. Důležité je pouze vzájemné pořadí prvků. Permutace $(134), (341)$ a $(413)$ jsou tudíž jedna a ta samá.

### 3.2.2 Skládání permutací

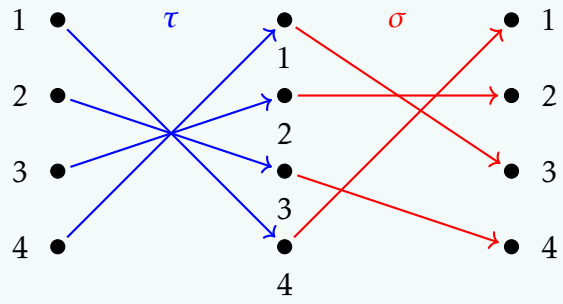
Jelikož permutace jsou speciálně zobrazení, dají se pochopitelně skládat. Navíc, protože doména i kodoména každé permutace na $X$ je právě množina $X$, mohu je skládat v libovolném pořadí a libovolném množství.

Na výpočet složení dvou permutací $\sigma, \tau \in S_X$ není myslím žádný vyloženě snadný postup. Člověk se musí zkrátka v cyklickém zápisu dočíst, kam posílá první permutace daný prvek a kam zase druhá permutace posílá obraz tohoto prvku.

**Příklad.** Ať $X = \{1, 2, 3, 4\}$ a $\sigma, \tau \in S_X, \sigma = (134), \tau = (14)(23)$. Pak

$$\sigma\tau = (243),$$

protože $\tau$ pošle 1 na 4 a $\sigma$ pošle 4 na 1, tedy $\sigma\tau$ pošle 1 na 1. Podobně pro ostatní prvky. V šipkách složení $\sigma\tau$ vypadá takto:




Obrázek 19: Složení permutací $\sigma\tau$ znázorněno v šipkách.

Všimněte si ale, že

$$\tau\sigma = (123).$$

**Výstraha.** Jak bylo vidno z předchozího příkladu, skládání permutací (jako obecně i relací) **není komutativní**. Dokonce platí, že pouze skládání permutace se sebou samou je komutativní, čili jsou-li $\sigma, \tau \in S_X$, pak

$$\sigma\tau = \tau\sigma \Rightarrow \tau = \sigma$$

za předpokladu, že $\#X \geq 3$.

Nyní si konečně povíme, co znamená „řád“ permutace a že každou permutaci lze rozložit na „transpozice“.

Když $\sigma = c_1 c_2 \cdots c_n$ je rozklad permutace $\sigma \in S_X$ na cykly $c_1, c_2, \ldots, c_n$, pak *délkou cyklu* $c_i$ myslíme počet prvků množiny $X$, které se v něm vyskytují. Tedy, např. délka cyklu $(1324)$ je 4 a délka cyklu $(457)$ je 3.

**Definice 3.2.4** (Transpozice). Permutace $\sigma \in S_X$ se nazývá *transpozice*, když obsahuje právě jeden cyklus délky 2. Lidsky řečeno, transpozice jsou přesně ty permutace, které prohazují dva prvky.

Možná trochu překvapivý výsledek ohledně permutací je, že každou permutaci lze napsat jako složení transpozic. Navíc je algoritmus velmi přímočarý – stačí zkrátka každý cyklus délky $k$ rozložit na $k-1$ transpozic tak, že každý „vnitřní“ prvek cyklu zdvojíme. Zformulujeme si tvrzení a ukážeme si algoritmus na příkladě.

**Tvrzení 3.2.5** (Rozklad na transpozice). Ať $\sigma \in S_X$ a $\#X \geq 2$ (jinak bychom neměli dva prvky k prohození). Pak existují transpozice $\tau_1, \ldots, \tau_n$ takové, že

$$\sigma = \tau_1 \tau_2 \cdots \tau_n.$$

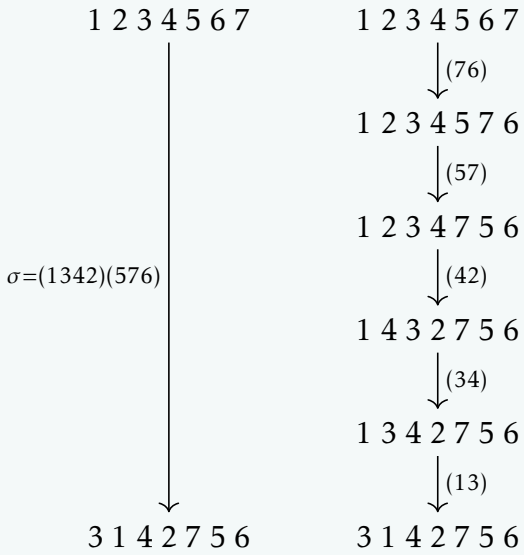
Navíc, počet transpozic v rozkladu $\sigma$ je určen jednoznačně.

**Důkaz.** Formální. Vynechám. Ideu si ukážeme na příkladě. □

**Příklad.** Rozložíme permutaci $\sigma = (1342)(576)$ na transpozice. Každý cyklus vlastně rozdělíme na cykly délky dva (což jsou vlastně transpozice) zdvojením každého vnitřního prvku. Tedy, cyklus (576) se rozdělí na transpozice (57) a (76) a cyklus (1342) se rozdělí na (13), (34) a (42). Pak dostaneme (**pozor na pořadí!**)

$$\sigma = (13) \circ (34) \circ (42) \circ (57) \circ (76),$$

což je rozklad $\sigma$ na transpozice. Na obrázku vidíte znázornění aplikace permutace $\sigma$ na množinu $\{1, \ldots, 7\}$ a postupnou aplikaci příslušných transpozic.




Obrázek 20: Znázornění rozkladu permutace $\sigma$ na transpozice.

**Definice 3.2.6** (Sudá/lichá permutace)**.** Permutaci $\sigma \in S_X$ nazveme *sudou*, když její rozklad na transpozice obsahuje sudý počet transpozic. Jinak ji nazveme *lichou*.

Poslední zajímavý výsledek o permutacích, který zmíníme, říká, že když jednu permutaci složím samu se sebou dostatečněkrát, dostanu identické zobrazení na $X$. Počtu složení se formálně říká „řád“ permutace.

**Definice 3.2.7** (Řád permutace)**.** Mějme $\sigma \in S_X$. Přirozené nenulové číslo $k \in \mathbb{N}$ nazveme *řádem* permutace $\sigma$, když

$$\sigma^k = \mathbb{1}_X,$$

kde výrazem $\sigma^k$ myslíme složení $\sigma$ se sebou samou $k$-krát, tj.

$$\sigma^k := \underbrace{\sigma \circ \sigma \circ \cdots \circ \sigma}_{k\text{-krát}}.$$

Řád permutace $\sigma$ značíme $\operatorname{ord}\sigma$ (z angl. **ord**er).

Na konec podsekce si rozmyslíme, že každá permutace má konečný řád a jak ho počítat. Uvažme třeba permutaci $\sigma = (143)$. Tahle permutace posílá 1 na 4, 4 na 3 a 3 zpět na 1. To ovšem znamená, že když ji „zopakuji“ třikrát za sebou, dostanu se rovnou z 1 na 1. Vskutku, můžete si ověřit, že

$$\sigma^3 = (143) \circ (143) \circ (143) = \mathbb{1}_{\{1,2,3,4\}}.$$

Tento pohled napovídá, že když cyklus délky $k$ zopakuji $k$-krát, zobrazím všechny prvky v tomto cyklu na ony samé.

Co když mám ale permutaci složenou z cyklů různých délek, jako třeba $(143)(25)$? No, cyklus $(143)$ musím zopakovat třikrát a cyklus $(25)$ dvakrát. Když ale tuto permutaci třikrát zopakuji, nedostanu identické zobrazení, protože cyklus $(25)$ zopakovaný třikrát je zase $(25)$. Když se zamyslíme, zjistíme, že abych dostal z permutace identické zobrazení, musím ji opakovat právě tolikrát, kolik je nejmenší společný násobek délek jejích cyklů, aby se každý cyklus zopakoval nějakým násobkem své délky. Zformulujeme si tento fakt jako tvrzení, ale dokazovat ho nebudeme, protože důkaz je otravně formální a ideu jsme si právě řekli.

**Tvrzení 3.2.8** (O řádu permutace)**.** Ať $\sigma \in S_X$ a

$$\sigma = c_1 c_2 \cdots c_n$$

je zápis $\sigma$ v cyklech $c_1, \ldots, c_n$. Označme $d_i$ délku cyklu $c_i$ pro každé $i \leq n$. Pak

$$\operatorname{ord} \sigma = \operatorname{lcm}(d_1, \ldots, d_n),$$

kde lcm (z angl. **l**east **c**ommon **m**ultiple) značí nejmenší společný násobek.

Myslím, že tohle tvrzení je pěkným příkladem přirozeného avšak poměrně silného tvrzení. Kdybyste nevěděli nic o permutacích a řekl bych vám, že máte dokázat fakt, že když bijekci na konečné množině složím samu se sebou hodněkrát, dostanu identické zobrazení, asi byste se zapotili.

### 3.2.3 Problém sta vězňů

Za sedmero horami a sedmero řekami, byla nebyla kdysi jedna věznice, ve které bylo žilo a živořilo sto vězňů. Její dozorce, chor a zchřadl, matematik řemeslem a krutovládce povahou, jednoho jitra rozhodl, že vězně propustí – buď do světa, nebo až do toho příštího.

Vyklidiv vězeňskou jídelnu, postavil zde sto dřevěných stolic nesoucích sto papírových obálek. Mezi obálky rozloučil sto kamení, v každém toporně vyryto různé číslo od jedné do sta.

Rozkázav vězně seřaditi před jídelnu, přičlenil každému rovněž různé číslo od jedné do sta, v pořadí, kterak stáli. Vězni měli po řadě postoupit do jídelny a otevřít padesát obálek. Našli-liž mezi nimi kámen zračící jejich číslo, vrátili jej zpět a směli opustit jídelnu opačným východem.

Podmínka propuštění děla, že každý vězeň musí v jedné z padesáti obálek, které otevřel, uzřít své číslo. Pokud byť i jeden vězeň své číslo neobjevil, celé těleso vězňů bylo by kvapně popraveno.

Asi není těžké si rozmyslet, že tohle je úloha na permutace. Pořadí, v jakém jsou umístěny stolice, můžeme vnímat jako množinu přirozených čísel $\{1, \ldots, 100\}$ a rozmístění kamenů do obálek jako jednu její náhodně zvolenou permutaci. Tedy to, že v obálce číslo 6 je kámen s číslem 23, zna-

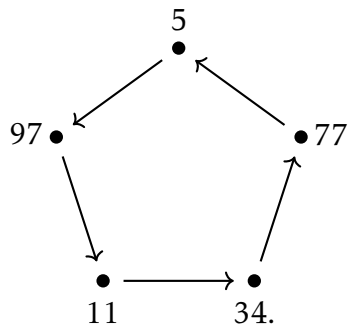
mená, že tato permutace posílá prvek 6 na prvek 23.

Pokud by vězni vybírali obálky k otevření náhodně, měl by každý z nich na nalezení kamene se svým číslem šanci přesně 1/2. Protože tahle situace se má opakovat stokrát po sobě, šance, že všichni najdou tímto způsobem své číslo, je $(1/2)^{100}$, tedy 1 ku 1 267 650 600 228 229 401 496 703 205 376 – čili pořád větší, než že si MŠMT všimne, že existují počítače – ale přesto malá.

My tvrdíme, že ve skutečnosti existuje strategie, která umožní vězňům uniknout poslednímu dechu s šancí větší než 30 %. Navíc je velmi přímočará. Stačí, aby každý vězeň otevíral obálky v pořadí, které určují kameny v nich obsažené. Konkrétně, každý vězeň otevře jako první obálku na stolici se stejným pořadím, jako je to jeho. Dál půjde otevřít obálku na stolici s pořadím, které odpovídá číslu na kameni v obálce, kterou právě otevřel. Takhle pokračuje, dokud nenajde své číslo, nebo dokud neprojde padesát různých obálek.

Ilustrujme si postup na příkladě. Do jídelny vstoupí vězeň s číslem 11. Otevře obálku číslo 11 a najde v ní kámen s číslem 34. Dále otevře obálku s číslem 34, v níž je číslo 77. Ještě jednou, v obálce číslo 77 najde číslo 5. V obálce číslo 5 najde kámen s číslem 97. A nakonec – Ejhle! – v obálce číslo 97 najde kámen se svým číslem – 11.

Označme permutaci určenou čísly na kamenech v obálkách třeba $\kappa$ (podle slova **k**ámen). Situace v předchozím odstavci znamená, že v $\kappa$ existuje cyklus




Nyní už si můžeme rozmyslet, za jaké podmínky objeví všichni vězni obálku s kamenem svého čísla. V moment, kdy vězeň otevře první obálku, dostane se tím do jednoho konkrétního cyklu permutace $\kappa$. Podle dané

strategie bude tento cyklus sledovat až do konce (když „začátkem“ cyklu myslíme číslo vězně). Pokud je délka daného cyklu třeba $d$, pak v moment, kdy otevře tento vězeň $d$-tou obálku, bude v ní kámen s jeho číslem. Samozřejmě, podmínka propuštění říkala, že každý vězeň smí otevřít maximálně 50 obálek. Protože každý vězeň má přiřazeno jiné, určitě čísla všech vězňů dohromady vyčerpají všechny cykly permutace $\kappa$. To znamená, že všichni vězni najdou svoje číslo tímto postupem jedině v případě, **kdy permutace $\kappa$ neobsahuje cyklus délky větší než 50**. Zformulujeme si to jako pozorování.

> **Pozorování.** Problém sta vězňů má řešení (tedy všech sto vězňů bude propuštěno) právě tehdy, když permutace $\kappa$, určená čísly na kamenech v obálkách, neobsahuje cyklus délky větší než 50.

Abychom spočítali šanci vězňů na úspěch, zbývá nám umět spočítat počet všech takových permutací, tj. permutací s cykly délky maximálně 50.

Ať $X := \{1,\dots,100\}$. Bude výhodnější řešit opačný problém, tedy hledat počet permutací s cyklem délky aspoň 51, protože (vzhledem k tomu, že množina $X$ má 100 prvků), takový cyklus tam může být nejvýše jeden.

Ať $C_n(S_X)$ značí počet všech permutací na $X$ s aspoň jedním cyklem délky $n$. Jak jsme právě řekli, pro $n \geq 51$ může mít libovolná permutace takový cyklus jenom jeden. Dále je zřejmé, že počet všech permutací s cyklem délky *aspoň* 51 spočtu tak, že sečtu počty permutací s cyklem délky $n$ pro všechna $n$ od 51 do 100. Když $C_{\geq 51}(S_X)$ značí počet permutací s cyklem délky aspoň 51, máme

$$C_{\geq 51}(S_X) = \sum_{n=51}^{100} C_n(S_X).$$

Spočítáme $C_n(S_X)$ pro $n \geq 51$. Abychom určili permutace s cyklem délky $n$, musíme zvolit $n$ z těch 100 čísel, která se v cyklu objeví. Počet způsobů, jak zvolit $n$ čísel ze 100 je $\binom{100}{n}$ (vizte sekci o kombinačních číslech). Čísla v tomto cyklu mohou být uspořádána $n!$ způsoby. **Ovšem, pozor!** Nezapomeňte, že uvnitř cyklu mohu čísla posouvat a cyklus tím nechat stejný. Protože mám v cyklu $n$ čísel, dělá to dohromady $n$ možných posunutí. Tedy mám

$$\frac{\binom{100}{n} n!}{n} = \binom{100}{n} (n-1)!$$

různých způsobů, jak zvolit cyklus délky $n$.

Konečně, zbývající čísla (tedy ta mimo ten cyklus) můžu přeuspořádat $(100-n)!$ způsoby. Celkem, počet všech permutací s cyklem délky $n$ pro $n \geq 51$ je

$$C_n(S_X) = \binom{100}{n}(n-1)!(100-n)!.$$

Je na čase završit výpočet. Všech možných permutací na 100 číslech je 100!. Počet všech permutací s cyklem délky aspoň 51 je $C_{\geq 51}(S_X)$. Čili šance, že náhodně vybraná permutace na 100 číslech **obsahuje** cyklus délky větší než 50 je

$$\frac{C_{\geq 51}(S_X)}{100!} = \frac{\sum_{n=51}^{100} C_n(S_X)}{100!} = \frac{\sum_{n=51}^{100} \binom{100}{n}(n-1)!(100-n)!}{100!} \approx 0.688,$$

čili 68,8 %. To ovšem znamená, že šance, že náhodná permutace **neobsahuje** cyklus délky větší než 50 je přibližně $1 - 0.688 = 0.312$. Takže při využití této strategie mají vězni přibližně 31.2% šanci na přežití. O něco lepší než náhodné zkoušení.

**Cvičení 3.2.1.** Spočtěte složení $\sigma\tau$ a $\tau\sigma$, když

(1) $\sigma = (143)(26), \tau = (146)(253)$,

(2) $\sigma = (14)(25)(36), \tau = (123456)$,

(3) $\sigma = (145)(263), \tau = (154)(236)$.

**Cvičení 3.2.2.** Určete řád permutace $\sigma$, kde

(1) $\sigma = (1345)$,

(2) $\sigma = (1346)(28)(579)$.

**Cvičení 3.2.3** (těžké)**.** Určete číslo $C_n(S_X)$, kde $\#X = 100$ a $n \leq 50$, tedy počet všech permutací na 100 číslech s aspoň jedním cyklem délky menší nebo rovné 50.

Samozřejmě jich je $100! - C_{\geq 51}(S_X)$, ale cílem úlohy je spočítat je nějak chytře, aby člověk dostal hezčí vzoreček.

**Cvičení 3.2.4.** Ať $\sigma \in S_n$, tedy $\sigma$ je permutace množiny $\{1,\dots,n\}$. Řekneme, že $\sigma$ *invertuje* dvojici $(i,j)$, kde $i,j \in \{1,\dots,n\}$, když $i < j$, ale $\sigma(i) > \sigma(j)$.

Definujme

$$I(\sigma) := \{(i,j) \in \{1,\dots,n\}^2 \mid \sigma \text{ invertuje } (i,j)\},$$

čili $I(\sigma)$ je množina všech dvojic $(i,j)$, které $\sigma$ invertuje. Uvědomme si, že $I(\sigma)$ je podmnožinou $\{1,\dots,n\}^2$, čili relací na $\{1,\dots,n\}$.

(1) Dokažte, že $I(\sigma)$ je transitivní relace na $\{1,\dots,n\}$ pro každou permutaci $\sigma \in S_n$.

(2) * Navrhněte algoritmus, který pro danou permutaci $\sigma \in S_n$ spočte $\#I(\sigma)$.

(3) Spočtěte počet invertovaných dvojic, čili $\#I(\sigma)$, permutací $\sigma = (134)(579)(26)$.

## 3.3 Kombinační čísla

V této sekci se budeme zabývat asi poměrně přirozenou otázkou – kolik má množina $X$ podmnožin velikosti $k$, kde $k$ může být libovolné číslo od 0 do $\#X$. Pro $k = 0$ i $k = \#X$ je odpověď jednoduchá: přesně jednu. Pro $k = 1$ člověku hádám taky dojde, že jednoprvková podmnožina je vlastně totéž, co její jediný prvek, takže takových máme $\#X$. Od $k = 2$ nám ale začína, borcovia, prituhovať. Nejspíš bychom pořád zvládli počet dvouprvkových množin nějak zpatlat, ale co třeba $k = \#X/2$ (když je $\#X$ sudé) a podobné takřka nekřesťanské výmysly? To už chce nějaké udělátko.

Nejdřív si to ale, jakožto slušní a spořádaní matematikové, definujeme.

**Definice 3.3.1** (Počet $k$-prvkových podmnožin)**.** Ať $X$ je množina a $0 \leq k \leq \#X$ je přirozené číslo. Definujeme množinu

$$\binom{X}{k} := \{A \subseteq X \mid \#A = k\}$$

všech $k$-prvkových podmnožin množiny $X$. Výraz $\binom{X}{k}$ čteme „$X$ nad $k$“.

Chvilku se budeme bavit přemítáním o způsobu, jak spočítat $\#\binom{X}{k}$ pro libovolné $k$ mezi 0 a $\#X$.

Použijeme kombinatorickou metodu důkazu zvanou *počítání dvěma způsoby*. Jde o užitečný (a podle mého velmi elegantní) přístup ve chvíli, kdy neumím spočítat rovnou konkrétní množství, ale umím spočítat něco velmi podobného. Technika počítání dvěma způsoby spočívá v tom, že tu kvantitu, kterou spočítat *umím*, vyjádřím na jedné straně pomocí kvantity, kterou spočítat *neumím*, a na druhé straně pomocí vzorečku, který znám. To mi dá rovnici, ze které pak vyjádřím to číslo, které chci určit.

Takto abstraktně vám asi *počítání dvěma způsoby* nic neřeklo, takže je raději pojďme aplikovat na zpytovaný problém. Počet $k$-prvkových podmnožin $X$ spočítat neumím; což takhle začít tím, že si nějakou náhodnou podmnožinu $\{x_1, \ldots, x_k\} \subseteq X$ zvolím. Jeden z důvodů, proč neumím počet takovýchhle podmnožin spočítat, je, že mi chybí nějaké *uspořádání*.

Zatím všechny věci, které jsme počítali, byly v jistém smyslu *uspořádané*.

Počet všech zobrazení $A \to B$ jsme počítali tak, že jsme prvky $A$ **jeden po druhém** zobrazovali na prvky $B$. Vlastně nevědomky jsme si tak nějakým náhodným způsobem *uspořádali* množinu $A$, aby se nám dobře počítalo. Permutace jsou taky přímo definované tak, že mění *uspořádání* prvků na množině.

Vybrat si nějaké uspořádání na $X$ a všechny její podmnožiny pak považovat za uspořádané podle stejného uspořádání je chytrý nápad, který přinese ovoce. Na konci výpočtu je však potřeba zanedbat všechny možné způsoby, kterými jsme $k$-prvkové množiny mohli uspořádat. Uvidíte, že nám nakonec opravdu vyjde, že počet všech $k$-prvkových podmnožin $X$ je vlastně počet všech $k$-prvkových podmnožin $X$ s nějakým konkrétním uspořádáním dělen počtem způsobů, kolika jsme takové uspořádání mohli zvolit.

Pojďme tedy místo množiny $\{x_1, \ldots, x_k\}$ uvažovat její „uspořádanou verzi“, tím míním $k$-tici $(x_1, \ldots, x_k)$. Rozdíl je samozřejmě v tom, že (třeba pro $k = 3$) je množina $\{x_1, x_2, x_3\}$ ta samá, co $\{x_2, x_1, x_3\}$, ale trojice $(x_1, x_2, x_3)$ je různá od trojice $(x_2, x_1, x_3)$. Záleží na pořadí, v jakém prvky za sebe umisťuji, na *uspořádání*.

Na první straně rovnice vzniklé *počítáním dvěma způsoby* si určíme, kolik různých takových $k$-tic mi z jedné množiny může vzniknout. No přeci tolik, kolika způsoby mohu mezi sebou proházet (nebo třeba cizeji „propermutovat“ ← hint btw) její prvky. Každá permutace na $\{x_1, \ldots, x_k\}$ mi určuje přesně jedno možné uspořádání. Těch je, podle tvrzení 3.2.3, $k!$. Z definice máme $\#\binom{X}{k}$ různých $k$-prvkových podmnožin $X$ a každá určuje $k!$ uspořádaných $k$-tic. Celkem těchto tedy máme $k! \cdot \#\binom{X}{k}$. To činí jednu stranu naší rovnice.

Na druhé straně, vybrat $k$ prvků z $X$ a nějak je uspořádat je přeci to samé, jako jim nějak přiřadit čísla od 1 do $k$. Takovéhle přiřazení mi určí přesně, v jakém pořadí prvky $x_1, \ldots, x_k$ jsou. Tedy, ten prvek, který dostane 1, jde první, ten s 2 jde druhý atd.

Uvědomme si, že výběr přesně $k$ prvků z množiny $X$ a jejich následné uspořádání (tedy přiřazení čísel od 1 do $k$) je vlastně prosté zobrazení $f : \{1, \ldots, k\} \to X$. Každému takovému zobrazení odpovídá uspořádaná $k$-tice $(f(1), \ldots, f(k))$ prvků z $X$ a naopak každou $k$-tici $(x_1, \ldots, x_k)$ lze považovat za prosté zobrazení $g : \{1, \ldots, k\} \to X$ takové, že $g(i) = x_i$ pro všechna $i \leq k$. Čili, pro každou podmnožinu $\{x_1, \ldots, x_k\} \subseteq X$ mám tolik uspořáda-

ných $k$-tic $(x_1,\dots,x_k)$ jako mám prostých zobrazení $\{1,\dots,k\} \to X$. Podle tvrzení 3.1.2 je tento počet roven $\prod_{i=0}^{k-1} \#X - i$.

Na závěr již stačí obě množství vzniklá dvěma různými způsoby počítání uspořádaných $k$-prvkových podmnožin $X$ porovnat. Dostaneme

$$\prod_{i=0}^{k-1} \#X - i = k! \cdot \#\binom{X}{k},$$

odkud okamžitě plyne

$$\#\binom{X}{k} = \frac{\prod_{i=0}^{k-1} \#X - i}{k!}.$$

Diskuse tvořící obsah předchozích dvou stránek je velmi obšírným důkazem následujícího tvrzení.

**Tvrzení 3.3.2** (Počet $k$-prvkových podmnožin)**.** Ať $X$ je konečná množina. Pak počet všech $k$-prvkových podmnožin $X$ je roven

$$\frac{\prod_{i=0}^{k-1} \#X - i}{k!}.$$

K tomuto tvrzení se víže definice tzv. *kombinačního čísla*.

**Definice 3.3.3** (Kombinační číslo)**.** Ať $k, n \in \mathbb{N}$ a $k \leq n$. Pak definujeme

$$\binom{n}{k} := \frac{\prod_{i=0}^{k-1} n - i}{k!}.$$

Výraz $\binom{n}{k}$ čteme $n$ nad $k$.

V závěsu definice 3.3.3 můžeme tvrzení 3.3.2 přepsat jako rovnost

$$\#\binom{X}{k} = \binom{\#X}{k}.$$

Ve škole se často učí interpretace kombinačního čísla $\binom{n}{k}$ jako „počet způsobů, jak volit $k$ předmětů z $n$ bez závislosti na pořadí“. Ta je plně v souladu s naší definicí, interpretujeme-li $X$ jako množinu předmětů a jednu

její $k$-prvkovou podmnožinu jako výběr $k$ předmětů, kde však pochopitelně (jedná se o **podmnožinu**) nezáleží na jejich uspořádání.

### 3.3.1 Problém rozkladu na sčítance

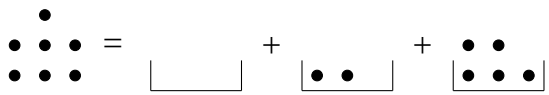
Oblíbený problém, který lze řešit počítáním způsobů, jak volit počet sourodých předmětů z většího množství, ale nikdo by to na první pohled nečekal, je *problém rozkladu na sčítance*.

Volme číslo $m \in \mathbb{N}$. *Rozkladem čísla $m$ na $r$ sčítanců* myslíme rovnost

$$m = x_1 + x_2 + \cdots + x_r = \sum_{k=1}^{r} x_k,$$

kde $x_i \geq 0$ jsou nezáporná celá čísla. Zajímá nás, kolika způsoby lze dané číslo $m$ rozložit na $r$ (ne nutně různých) sčítanců. Na první pohled nejde vůbec o kombinace (tedy o výběr bez závislosti na pořadí), protože pořadí sčítanců je zde důležité: rozklad $7 = 0 + 2 + 5$ **je různý** od rozkladu $7 = 2 + 0 + 5$. Situace se navíc v tomto směru zdá zcela beznadějná, neboť když jsou dvě a více čísel v rozkladu stejná, pak jejich prohození samozřejmě neurčuje jiný rozklad. Například, v rozkladu $7 = 2 + 2 + 3$ mohu prohodit první 2 s druhou 2 a nedostat tak odlišný rozklad.

První průlomovou myšlenkou je náhled, že číslo $m$ vlastně „rozhazuji“ mezi $r$ čísel. V tomto smyslu si lze představit přirozené číslo $m$ jako $m$ nerozlišitelných míčků, které rozděluji do $r$ košíků. Třeba rozklad $7 = 0 + 2 + 5$ by vypadal takto.



Obrázek 21: Vizualizace rozkladu $7 = 0 + 2 + 5$ jako rozdělení míčků do košíků.

Samozřejmě, v takovémto rozkladu určuje pravá strana jednoznačně tu levou, takže není třeba 7 míčků na levé straně znázorňovat. Podobně, když se dohodneme, že počty míčků v košících vždy sčítáme, lze i symboly + vynechat. Trochu složitější rozklad, třeba $9 = 1+0+3+5$ bychom nakreslili jak vidno na obrázku 22.

Obrázek 22: Zjednodušený nákres rozdělení míčků do košíků.

Nakonec si uvědomíme, že přeci není ani třeba kreslit jednotlivé košíky. Stačí, když všechny míčky vysypeme na zem a jenom mezi ně dáme nějaká hradla, abychom věděli, které míčky patřily do stejného košíku. Názorná ukázka, pro stejný rozklad, tedy $9 = 1 + 0 + 3 + 5$.

Obrázek 23: Ještě jednodušší nákres rozkladu $9 = 1 + 0 + 3 + 5$.

Jeden možný způsob, jak si snadno představit spojitost mezi rozkladem a touto poslední verzí jeho vizualizace je ten, že stěny či hradla odpovídají symbolům + a počet míčku mezi dvěma hradly odpovídá číslu mezi příslušnými symboly. Protože sčítanců je $r$, a tedy symbolů + je $r-1$, je hradel též $r-1$.

Jsme připraveni řešení úlohy završit. Rozmysleli jsme si, že každé rozmístění $r-1$ hradel mezi $m$ za sebou v řadě ležících míčků mi definuje jeden konkrétní rozklad čísla $m$ na $r$ sčítanců. Stačí tedy umět určit počet takových rozmístění.

To ale není těžké. Míčky a hradla činí dohromady $m+r-1$ objektů, z kterých přesně $r-1$ jsou hradla. Řečeno jinak, když z $m+r-1$ objektů vyberu těch $r-1$, která se stanou hradly, a zbytek přetvořím v míčky, pak určím rozklad čísla $m$ na $r$ sčítanců. Počet všech možných výběrů $r-1$ prvků z množiny o $m+r-1$ prvcích je $\binom{m+r-1}{r-1}$, kteréžto číslo je tudíž i počet způsobů, jak rozložit číslo $m$ na $r$ sčítanců.

### 3.3.2 Pár vlastností kombinačních čísel

Tahle sekce si neklade za cíl objevit zatím neznámý kontinent ani čtenáře naučit životu v afrických pralesích. Baže naopak, jedná se o prostou přílohu k již známému. Kombinační čísla se objevují, kdykoli člověk počítá s podobjekty konečných objektů, tedy v podstatě pořád. Věnujeme chvilku času prozkoumání způsobů, jak s nimi zacházet.

Nejprve jeden výpočetně užitečný vzoreček.

**Lemma 3.3.4.** Ať $k, n \in \mathbb{N}$, $k \leq n$. Platí

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

**Důkaz.** Z definice kombinačního čísla máme

$$\binom{n}{k} = \frac{\prod_{i=0}^{k-1} n-i}{k!},$$

stačí tedy ukázat, že

$$\prod_{i=0}^{k-1} n-i = \frac{n!}{(n-k)!}.$$

To je však zřejmé, neboť

$$\begin{aligned} n! &= n(n-1)\cdots(n-k+1)(n-k)(n-k-1)\cdots 1 \\ &= n(n-1)\cdots(n-k+1)(n-k)! = \left(\prod_{i=0}^{k-1} n-i\right)(n-k)!. \end{aligned}$$

Důkaz plyne z vydělení poslední rovnice číslem $(n-k)!$. □

Dále si povíme o tzv. *Pascalově trojúhelníku*. Tím se obvykle míní následující struktura.

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & 1 & & & & \\ & & & 1 & & 2 & & 1 & & & \\ & & 1 & & 3 & & 3 & & 1 & & \\ & 1 & & 4 & & 6 & & 4 & & 1 & \\ 1 & & 5 & & 10 & & 10 & & 5 & & 1 \\ & & & & & \vdots & & & & & \end{array}$$

Obrázek 24: Pascalův trojúhelník.

Jde vlastně o posloupnost řad, kde na začátku a na konci každé řady je číslo 1 a prostřední čísla dostanu tak, že sečtu ta dvě čísla z předchozí řady těsně nad ním.

Formálně můžeme říci, že Pascalův trojúhelník je posloupnost uspořáda-

ných $n$-tic $(p_1^n,\ldots,p_n^n) \in \mathbb{N}^n$ ($n$ jsou **indexy** řádků, nikoli mocniny), taková, že $p_i^n = p_{i-1}^{n-1} + p_i^{n-1}$ pro každé $n \geq 1$ a každé $1 \leq i \leq n$. Pro začátek položíme $p_1^1 = 1$ a v každém řádku dodefinujeme $p_0^n = p_{n+1}^n = 0$.

Skutečně, když $p_1^1 = 1$, tedy první číslo prvního řádku je 1, pak $p_1^2 = p_0^1 + p_1^1 = 0 + 1 = 1$ a $p_2^2 = p_1^1 + p_2^1 = 1 + 0 = 1$, čili druhý řádek je dvojice $(1,1)$. Zde vidíte důvod, proč jsme dodefinovali též 0-tý a $(n+1)$-ní prvek $n$-tého řádku. Museli bychom totiž jinak psát speciální pravidlo pro určení prvního a posledního prvku každého řádku. Místo toho předstíráme, že je Pascalův trojúhelník ještě z obou stran obklopen nulami.

Pro pořádek si ještě v tomto formálním pohledu spočteme třetí řádek, tj. trojici $(p_1^3, p_2^3, p_3^3)$. Máme

$$\begin{aligned} p_1^3 &= p_0^2 + p_1^2 = 0 + 1 = 1, \\ p_2^3 &= p_1^2 + p_2^2 = 1 + 1 = 2, \\ p_3^3 &= p_2^2 + p_3^2 = 1 + 0 = 1. \end{aligned}$$

Vše je, jak má být.

Budeme chtít ukázat, že $n$-tý řádek Pascalova trojúhelníku tvoří přesně čísla $\binom{n-1}{0}, \binom{n-1}{1}, \ldots, \binom{n-1}{n-1}$. Pro první řádky je to jistě pravda, neboť $\binom{0}{0} = 1$ (neboť 0! se tradičně definuje jako 1) a dále $\binom{1}{0} = \binom{1}{1} = 1$.

Rozepíšeme si, co naše tvrzení vlastně znamená z pohledu kombinačních čísel. Prvky $p_i^{n+1}$ v $(n+1)$-ním řádku Pascalova trojúhelníku jsou definovány pomocí prvků v předchozím řádku vzorcem $p_{i+1}^{n+1} = p_i^n + p_{i+1}^n$. A my tvrdíme, že $p_i^n = \binom{n-1}{i-1}$. (Ověřte si, že to je **opravdu** to, co říkáme!) Přepíšeme-li tuto rovnost v kombinačních číslech, potřebujeme dokázat, že

$$\binom{n}{i} = \binom{n-1}{i-1} + \binom{n-1}{i} \qquad (*)$$

pro každé $n \geq 1$ každé $1 \leq i \leq n$.

I když by to jistě nějak šlo upočítat, my zvolíme elegantnější způsob, který zůstává věrný tomu, co kombinační číslo vlastně **vyjadřuje**. Nezapomeňte, že $\binom{n}{i}$ je počet $i$-prvkových podmnožin $n$-prvkové množiny. Je jisté, že množina $(i-1)$-prvkových podmnožin je disjunktní (má prázdný průnik)

s množinou $i$-prvkových podmnožin. To ovšem znamená, že

$$\#\left(\binom{\{1,\dots,n-1\}}{i-1}\cup\binom{\{1,\dots,n-1\}}{i}\right)=\#\binom{\{1,\dots,n-1\}}{i-1}+\#\binom{\{1,\dots,n-1\}}{i}$$
$$=\binom{n-1}{i-1}+\binom{n-1}{i}.$$

Řečeno selsky, když vezmu množinu obsahující všechny $i$-prvkové i $(i-1)$-prvkové podmnožiny, pak její velikost je počet všech $i$-prvkových podmnožin plus počet všech $(i-1)$-prvkových podmnožin. No shit.

Čili, abychom dokázali rovnost (∗), najdeme bijekci mezi množinou všech $i$-prvkových a $(i-1)$-prvkových podmnožin $(n-1)$-prvkové množiny a množinou všech $i$-prvkových podmnožin $n$-prvkové množiny.

**Tvrzení 3.3.5** (Pascalova rovnost). Ať $1 \le i, n \in \mathbb{N}$ a $i \le n$. Pak platí

$$\binom{n}{i}=\binom{n-1}{i-1}+\binom{n-1}{i}.$$

**Důkaz.** Definujeme bijekci

$$f:\binom{\{1,\dots,n-1\}}{i-1}\cup\binom{\{1,\dots,n-1\}}{i}\to\binom{\{1,\dots,n\}}{i}.$$

Ať nejprve $A \in \binom{\{1,\dots,n-1\}}{i}$, čili $A$ je $i$-prvková podmnožina $\{1,\dots,n-1\}$. Pak je $A$ též $i$-prvková podmnožina $\{1,\dots,n\}$, neboli $A \in \binom{\{1,\dots,n\}}{i}$ a stačí definovat $f(A) := A$. Stručně řečeno, $f$ je identické zobrazení na $i$-prvkových podmnožinách.

Teď ať $B \in \binom{\{1,\dots,n-1\}}{i-1}$. Pak $B \cup \{n\}$ je $i$-prvková podmnožina $\{1,\dots,n\}$ a tedy můžeme definovat $f(B) := B \cup \{n\}$.

Je zřejmé, že $f$ je bijekce. Když $A \subseteq \{1,\dots,n\}$ neobsahuje $n$, pak je jejím vzorem při $f$ ta samá množina, tj. $A$. Když $A \subseteq \{1,\dots,n\}$ obsahuje $n$, pak je jejím vzorem množina $A \setminus \{n\} \subseteq \{1,\dots,n-1\}$.

Tím je důkaz dokončen. □

Předchozí tvrzení ukazuje, že pro $n$-tý řádek Pascalova trojúhelníka oprav-

du platí rovnost

$$(p_1^n,\dots,p_n^n)=\left(\binom{n-1}{0},\binom{n-1}{1},\dots,\binom{n-1}{n-1}\right).$$

Na závěr celé sekce o kombinačních číslech si ukážeme ještě poslední snadno dokazatelnou rovnost, která je však výpočetně též užitečná.

**Lemma 3.3.6.** Ať $k,n\in\mathbb{N}$ a $k\le n$. Pak

$$\binom{n}{k}=\binom{n}{n-k}.$$

**Důkaz.** Nalezneme bijekci

$$\binom{\{1,\dots,n\}}{k}\to\binom{\{1,\dots,n\}}{n-k}.$$

Uvědomme si, že když $A\subseteq\{1,\dots,n\}$ a $\#A=k$, pak $\#(\{1,\dots,n\}\setminus A)=n-k$. Kýžená bijekce je tudíž zobrazení $A\mapsto\{1,\dots,n\}\setminus A$. □

Ještě několik úloh pro bystré hlavy.

**Cvičení 3.3.1.** Dokažte, že

$$\sum_{i=0}^{n}\binom{n}{i}^2=\binom{2n}{n}.$$

**Hint**: použijte lemma 3.3.6.

**Cvičení 3.3.2.** Dokažte vzorec

$$\sum_{k=r}^{n}\binom{k}{r}=\binom{n+1}{r+1}$$

pro pevné $r\in\mathbb{N}$ indukcí podle $n\in\mathbb{N}$.

**Cvičení 3.3.3** (těžké)**.** Kolik existuje podmnožin $\{1,\dots,n\}$, které neobsahují žádná dvě po sobě jdoucí čísla. Formálně, určete velikost množiny

$$\{A \subseteq \{1,\dots,n\} \mid \{i,j\} \not\subseteq A, \text{ kdykoli } |i-j|=1\}.$$

**Cvičení 3.3.4** (trocha teorie čísel)**.** Ať $p$ je prvočíslo a $k,n$ přirozená čísla.

(a) Dokažte, že pro $k < p$ je $\binom{p}{k}$ dělitelné $p$.

(b) Dokažte, že $\binom{n}{p}$ je dělitelné $p$ právě tehdy, když $\lfloor n/p \rfloor$ je dělitelné $p$, kde $\lfloor\cdot\rfloor$ značí *dolní celou část*.

**Cvičení 3.3.5.** Budeme vybírat $k$-tice předmětů z $n$ druhů předmětů. Budeme uvažovat různé typy výběru podle toho, jestli vybíráme $k$-tice uspořádané, nebo neuspořádané (tj. podmnožiny) a též podle toho, zda každého druhu je vždy jen jeden předmět, či nikoli. Doplňte následující tabulku:

| | Jen 1 předmět každého druhu | Libovolně mnoho předmětů každého druhu |
|---|---|---|
| Uspořádané $k$-tice | | |
| Neuspořádané $k$-tice | | |

Tabulka 1: Výběr $k$-tic předmětů z $n$ druhů předmětů.

**Cvičení 3.3.6** (těžké)**.** Kolika způsoby můžeme postavit 7 čarodějnic a 5 vodníků do řady tak, aby 2 vodníci nikdy nestáli vedle sebe?

## 3.4 Princip inkluze a exkluze

Další základní úlohou, která se objevuje napříč diskrétní matematikou, je problém velikosti sjednocení množin.

Velikost průniku se dá spočítat snadno. Člověk se zkrátka podívá, které prvky leží v každé množině, a spočítá je. Vlastně stačí vzít tu nejmenší množinu a započítat jen ty prvky, které leží i ve všech ostatních.

Se sjednocením je to však těžší, neboť některé prvky mohou ležet v jedné množině, ve všech množinách nebo v jakémkoli počtu množin mezi těmito extrémy. Měl-li by člověk počítat velikost sjednocení manuálně, musel by množiny procházet jednu po druhé a navíc si u každého prvku pamatovat, zda ho už započetl, či ještě ne. Tento způsob je sice algoritmicky přímočarý, ale neefektivní i z hlediska výpočetního. Navíc často není ani použitelný, protože se může stát, že znám jen velikosti jednotlivých množin, ale ne přesný výčet jejich prvků. Lepší způsob, který si teď ukážeme a vysvětlíme, počítá, možná i trochu překvapivě, velikost sjednocení vlastně jako součet přes všechny možné průniky.

Nejprve jakýsi „motivační“ příklad. Jeho praktické využití je za účelem zachování jednoduchosti pravdaže poněkud omezené.

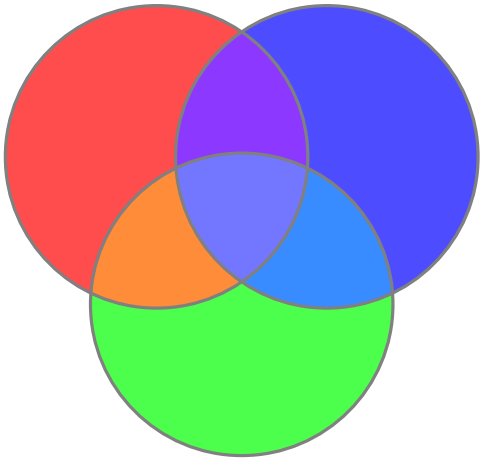
**Příklad.** Na Gymnáziu Growth Severní Obec je možné se učit třem cizím jazykům – němčině, španělštině a francouzštině. Německy se učí 30 studentů, španělsky 25 studentů a 2 velmi nešťastné osoby se učí francouzsky. Přitom, mezi němčináři jsou 4 španělštináři a jeden bageťák. Exkluzivně románským jazykům neholduje nikdo. Jeden nolifer se ale dokonce učí všem třem jazykům naráz.

Kolik celkem studentů se učí aspoň jednomu cizímu jazyku?

Můžete si zkusit úlohu vyřešit manuálně. Chvíli vám to zabere a na konci navíc zjistíte, že jste vynalezli tzv. *princip inkluze a exkluze*, což je honosný název pro vzoreček, který říká, že velikost sjednocení množin dostanu tak, že sečtu velikosti všech průniků lichého počtu množin a odečtu velikosti průniků sudého počtu množin.

Protože vám předchozí věta vůbec nic neřekla, ukážeme si nejprve na třech množinách (a pomocí obrázků), jak *princip inkluze a exkluze* funguje.

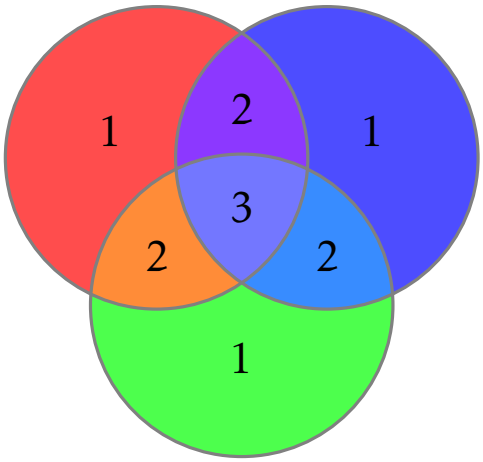
Uvažujme tři množiny $A$, $B$ a $C$ s neprázdným průnikem, tedy $A \cap B \cap C \neq \emptyset$. Nakreslenou vidíte tuto situaci na obrázku 25.



Obrázek 25: Množiny $A, B, C$ s neprázdným průnikem.

Budeme nyní počítat velikost sjednocení $A \cup B \cup C$.

Za předpokladu, že by množiny byly disjunktní, stačilo by zkrátka sečíst velikosti jednotlivých množin. V této situaci však započteme některé prvky vícekrát. Stane se to proto, že když sečtu třeba $\#A + \#B$, tak prvky, které leží i v $A$ i v $B$ (tedy ty v průniku $A \cap B$) započtu dvakrát. Na obrázku níže vidíte, kolikrát započteme který „díl“ svého Vennova diagramu.

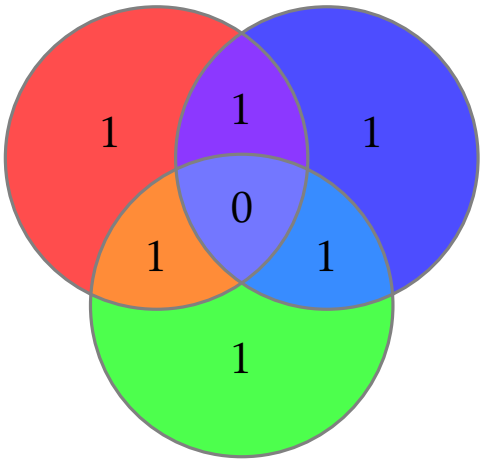



Obrázek 26: Znázornění součtu $\#A + \#B + \#C$.

Jak můžete vyčíst z obrázku, některé průniky jsme započítali mockrát. Černá čísla značí, kolikrát jsme každý kousek sjednocení započetli, když

jsme sečetli $\#A + \#B + \#C$. Například jsme tedy započetli každý prvek ležící v průniku $A \cap C$ dvakrát a každý prvek v průniku $A \cap B \cap C$ dokonce třikrát.

Abychom situaci napravili, a započetli průnik každého páru množin jen jednou, odečteme od součtu $\#A + \#B + \#C$ velikosti všech průniků dvou množin. Odpovídající diagram vidíte níže.




Obrázek 27: Znázornění výrazu $\#A + \#B + \#C - \#A \cap B - \#A \cap C - \#B \cap C$.

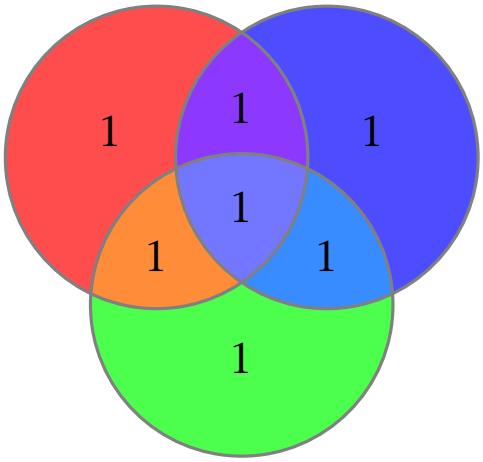
Ovšem, nyní jsme zase odečetli třikrát průnik $A \cap B \cap C$, protože každý prvek v $A \cap B \cap C$ leží zároveň v $A \cap B$, v $A \cap C$ i v $B \cap C$, a ty jsme všechny odečetli. Tedy nám z průniku $A \cap B \cap C$ žádné prvky nezůstaly. Situaci napravíme tím, že jeho velikost přičteme zpátky. Celkově dostaneme pro velikost sjednocení $\#A \cup B \cup C$ vzorec

$$\#A \cup B \cup C = \#A + \#B + \#C - \#A \cap B - \#A \cap C - \#B \cap C + \#A \cap B \cap C.$$

Finální znázornění vidíte na obrázku 28.

Formální znění principu inkluze a exkluze dí, že tento postup funguje zcela obecně, tedy že pokud průniky sudého počtu množin odečítám a lichého počtu přičítám, dostanu tím velikost sjednocení.

Ještě předtím, než ho uvedeme a dokážeme, si ale určíme počet studentů z úvodního příkladu. Množiny němčinářů, španělštinářů a francouzštinářů označíme po řadě $N$, $Š$ a $Ž$ (jako **ž**abožrout). Ze zadání víme, že $\#N = 30, \#Š = 25$ a $\#Ž = 2$. Dále víme, že $\#N \cap Š = 4$, protože mezi němčináři jsou čtyři španělštináři; podobně $\#N \cap Ž = 1$ a $\#Š \cap Ž = 0$. Konečně,




Obrázek 28: Velikost sjednocení $\#A \cup B \cup C$ podle principu inkluze a exkluze.

$\#N \cap \check{S} \cap \check{Z} = 1$. Podle principu inkluze a exkluze máme

$$\begin{aligned}\#N \cup \check{S} \cup \check{Z} &= \#N + \#\check{S} + \#\check{Z} - \#N \cap \check{S} - \#N \cap \check{Z} - \#\check{S} \cap \check{Z} + \#N \cap \check{S} \cap \check{Z} \\ &= 30 + 25 + 2 - 4 - 1 - 0 + 1 = 53.\end{aligned}$$

Celkem se tedy na Gymnáziu Growth Severní Obec učí 53 studentů cizím jazykům.

Posledním krokem, který zbývá, je umět princip inkluze a exkluze nějak rozumně zapsat. Tohle je asi první situace, na kterou jsme narazili, kdy bez rozumného zápisu daného tvrzení bychom se ze symbolů doslova zbláznili (hlavně v jeho důkazu). Natvrdo napsaný říká princip inkluze a exkluze, že

$$\begin{aligned}\#\bigcup_{i=1}^{n} A_i &= \#A_1 + \#A_2 + \cdots + \#A_n \\ &\quad - \#A_1 \cap A_2 - \#A_1 \cap A_3 - \cdots - \#A_1 \cap A_n - \#A_2 \cap A_3 - \cdots - \#A_{n-1} \cap A_n \\ &\quad + \#A_1 \cap A_2 \cap A_3 + \cdots + \#A_{n-2} \cap A_{n-1} \cap A_n + \cdots + (-1)^{n-1} \bigcap_{i=1}^{n} A_i.\end{aligned}$$

Jak sami vidíte, z tohoto zápisu je sotva (pokud vůbec) poznat, čemu se vlastně velikost sjednocení $\bigcup_{i=1}^{n} A_i$ rovná. Musíme tento zápis zjednodušit. Nejprve, pro každou $I \subseteq \{1, \ldots, n\}$ znamená zápis

$$\bigcap_{i \in I} A_i$$

průnik přesně těch množin $A_i$, jejichž indexy leží v množině $I$. Je-li tedy např. $I = \{1, 4, 5\}$, pak

$$\bigcap_{i \in I} A_i = A_1 \cap A_4 \cap A_5.$$

Pro extrémní případ $I = \emptyset$ dodefinujeme $\bigcap_{i \in I} A_i := \emptyset$, tedy prázdný průnik je prázdná množina.

Dále si rozmyslíme jednoduchý způsob, jak zapsat znaménko u každého z průniků. Víme, že průnik sudého počtu množin má záporné znaménko a průnik lichého počtu má kladné. Ještě jinak, když $\#I$ je sudé číslo, pak $\bigcap_{i \in I} A_i$ musí dostat záporné znaménko, a když $\#I$ je liché číslo, tak kladné. V matematice se pro tyto účely obvykle používá $(-1)^k$ pro přirozené číslo $k \in \mathbb{N}$, protože $(-1)^k = 1$, když $k$ je sudé, a $(-1)^k = -1$, když $k$ je liché. To ale znamená, že výraz $\#\bigcap_{i \in I} A_i$ musí být vynásobený číslem $(-1)^{\#I-1}$.

Už jsme skoro hotovi. Pro vyjádření $\#\bigcup_{i=1}^{n} A_i$ pomocí průniků musíme sečíst velikosti všech průniků $k$ různých množin pro $k$ od 1 až do $n$ se správným znaménkem. Čili, musíme sčítat všechny výrazy $(-1)^{\#I-1}\#\bigcap_{i \in I} A_i$ pro všechny podmnožiny $I \subseteq \{1, \ldots, n\}$.

Předchozí diskuse nám umožňuje zapsat princip inkluze a exkluze jako rovnost

$$\#\bigcup_{i=1}^{n} A_i = \sum_{I \subseteq \{1,\ldots,n\}} (-1)^{\#I-1} \#\bigcap_{i \in I} A_i,$$

kde symbol $\sum_{I \subseteq \{1,\ldots,n\}}$ značí součet přes všechny **podmnožiny** množiny $\{1, \ldots, n\}$.

Ještě jednou tedy princip inkluze a exkluze formulujeme jako větu.

**Věta 3.4.1** (Princip inkluze a exkluze)**.** Ať $A_1, A_2, \ldots, A_n$ jsou množiny. Pak platí rovnost

$$\#\bigcup_{i=1}^{n} A_i = \sum_{I \subseteq \{1,\ldots,n\}} (-1)^{\#I-1} \#\bigcap_{i \in I} A_i. \qquad (\triangle)$$

**Důkaz.** Důkaz povedeme indukcí podle $n$.

Když $n = 1$, pak na levé straně rovnosti ($\triangle$) máme

$$\#\bigcup_{i=1}^{1} A_i = \#A_1$$

a na pravé straně

$$\begin{aligned}\sum_{I\subseteq\{1\}} (-1)^{\#I-1} \#\bigcap_{i\in I} A_i &= (-1)^{\#\emptyset-1} \#\bigcap_{i\in\emptyset} A_i + (-1)^{\#\{1\}-1} \#\bigcap_{i\in\{1\}} A_i \\ &= (-1)^{-1}\cdot 0 + (-1)^0 \cdot \#A_1 = \#A_1,\end{aligned}$$

protože jediné podmnožiny $\{1\}$ jsou $\emptyset$ a $\{1\}$. Čili pro $n = 1$ rovnost ($\triangle$) platí.

Předpokládejme nyní, že rovnost platí pro všechna čísla menší než $n$ a dokazujme rovnost pro $n$. Sjednocení na levé straně ($\triangle$) můžeme napsat jako

$$\bigcup_{i=1}^{n} A_i = \bigcup_{i=1}^{n-1} A_i \cup A_n.$$

Označíme si $B_1 := \bigcup_{i=1}^{n-1} A_i$ a $B_2 := A_n$. Podle principu inkluze a exkluze pro 2 množiny (ten můžeme použít z indukčního předpokladu) máme

$$\#B_1 \cup B_2 = \#B_1 + \#B_2 - \#B_1 \cap B_2.$$

Přepsání zpět do množin $A_i$ nám dá

$$\#\bigcup_{i=1}^{n} A_i = \#\bigcup_{i=1}^{n-1} A_i \cup A_n = \#\bigcup_{i=1}^{n-1} A_i + \#A_n - \#\left(\bigcup_{i=1}^{n-1} A_i \cap A_n\right).$$

Nejprve si uvědomíme, že

$$\left(\bigcup_{i=1}^{n-1} A_i\right) \cap A_n = \bigcup_{i=1}^{n-1} (A_i \cap A_n),$$

tedy, že je to samé sjednotit množiny $A_i$ a pak je proniknout s množinou $A_n$ jako nejprve proniknout každou množinu $A_i$ s množinou $A_n$ a pak všechny ty průniky sjednotit.

Teď můžeme (opět z indukčního předpokladu) použít princip inkluze a exkluze na sjednocení $\bigcup_{i=1}^{n-1} A_i$ a zároveň na $\bigcup_{i=1}^{n-1}(A_i \cap A_n)$.

Dostaneme

$$\#\bigcup_{i=1}^{n-1} A_i = \sum_{I\subseteq\{1,\dots,n-1\}} (-1)^{\#I-1} \#\bigcap_{i\in I} A_i$$

a

$$\#\bigcup_{i=1}^{n-1}(A_i \cap A_n) = \sum_{I\subseteq\{1,\dots,n-1\}} (-1)^{\#I-1} \#\bigcap_{i\in I}(A_i \cap A_n).$$

Na jedné straně tedy máme vyjádření

$$\begin{aligned}\#\bigcup_{i=1}^{n} A_i = &\sum_{I\subseteq\{1,\dots,n-1\}} (-1)^{\#I-1} \#\bigcap_{i\in I} A_i + \#A_n \\ &- \sum_{I\subseteq\{1,\dots,n-1\}} (-1)^{\#I-1} \#\bigcap_{i\in I}(A_i \cap A_n).\end{aligned} \tag{1}$$

Na druhé straně potřebujeme dokázat (**ještě to nevíme!**), že

$$\#\bigcup_{i=1}^{n} A_i \stackrel{?}{=} \sum_{I\subseteq\{1,\dots,n\}} (-1)^{\#I-1} \#\bigcap_{i\in I} A_i. \tag{2}$$

Porovnáváme tedy součty na pravých stranách rovností (1) a (2). Jediný rozumný způsob, jak to udělat, je podívat se, že každý sčítanec v (2) je rovněž sčítancem v (1).

Rozlišíme dva případy podle toho, zda podmnožina $I$ obsahuje $n$, či ne.

(a) Ať nejprve $n \notin I$. Pak v sumě (2) je sčítanec

$$(-1)^{\#I-1} \#\bigcap_{i\in I} A_i,$$

kde ten průnik neobsahuje $A_n$, protože $n \notin I$. Jenže pak je $I \subseteq \{1,\dots,n-1\}$, a tedy i součet (1) obsahuje ten samý sčítanec (v té úplně první sumě nahoře) se stejným znaménkem.

(b) Teď předpokládáme, že $n \in I$. Musíme rozlišit zase dva případy (xD).

($\alpha$) $I = \{n\}$. Pak součet (2) obsahuje sčítanec

$$(-1)^{\#\{n\}-1} \# \bigcap_{i\in\{n\}} A_i = \#A_n,$$

který je ale i v součtu (1) (mezi těmi dvěma sumami).

($\beta$) $I$ obsahuje $n$ a $\#I \geq 2$. V tomto případě máme v součtu (2) sčítanec

$$(-1)^{\#I-1} \# \bigcap_{i\in I} A_i,$$

jejž si však můžeme přepsat jako

$$\begin{aligned}(-1)^{\#I-1} \# \bigcap_{i\in I} A_i &= (-1)^{\#I-1} \# \left( \bigcap_{i\in I\setminus\{n\}} A_i \right) \cap A_n \\ &= (-1)^{\#I-1} \# \bigcap_{i\in I\setminus\{n\}} (A_i \cap A_n),\end{aligned}$$

jelikož předpokládáme, že $n \in I$. Protože $I\setminus\{n\} \subseteq \{1,\dots,n-1\}$, je v součtu (1) (v té sumě dole) sčítanec

$$(-1)^{\#(I\setminus\{n\})-1} \# \bigcap_{i\in I\setminus\{n\}} (A_i \cap A_n),$$

což je v podstatě tentýž, až na znaménko. To však není problém, neboť si můžete všimnout, že v součtu (1) tu druhou sumu obsahující všechny průniky s množinou $A_n$ odečítám. Tedy, odpovídající sčítanec má ve skutečnosti znaménko $-(-1)^{\#(I\setminus\{n\})-1} = (-1)^{\#I-1}$.

Tím jsme ověřili, že součet ve vyjádření (1) obsahuje přesně tytéž sčítance jako součet ve vyjádření (2), a tudíž se jedná o stejný součet. Tím je podle principu matematické indukce důkaz dokončen. □

Nakonec se podíváme, že věta 3.4.1 souhlasí plně s naší intuicí v již prozkoumaném případě tří množin s neprázdným průnikem.

**Příklad.** Uvažme množiny $A_1, A_2, A_3$ takové, že $A_1 \cap A_2 \cap A_3 \neq \emptyset$. Podle principu inkluze a exkluze počítáme

$$\#A_1 \cup A_2 \cup A_3 = \sum_{I \subseteq \{1,2,3\}} (-1)^{\#I-1} \# \bigcap_{i \in I} A_i.$$

Jednoprvkové podmnožiny $\{1,2,3\}$ jsou $\{1\},\{2\}$ a $\{3\}$. Tedy započteme

$$(-1)^{1-1}\#A_1 + (-1)^{1-1}\#A_2 + (-1)^{1-1}\#A_3 = \#A_1 + \#A_2 + \#A_3.$$

Ty dvouprvkové jsou $\{1,2\},\{1,3\}$ a $\{2,3\}$. Dávají nám po řadě sčítance

$$\begin{aligned}&(-1)^{2-1}\#A_1 \cap A_2 + (-1)^{2-1}\#A_1 \cap A_3 + (-1)^{2-1}\#A_2 \cap A_3\\ &= -\#A_1 \cap A_2 - \#A_1 \cap A_3 - \#A_2 \cap A_3.\end{aligned}$$

Konečně, tříprvkovou podmnožinu má $\{1,2,3\}$ jenom jednu – sebe samu. Za tu si započteme

$$(-1)^{3-1}\#A_1 \cap A_2 \cap A_3 = \#A_1 \cap A_2 \cap A_3.$$

Když sečteme průniky přes všechny podmnožiny $\{1,2,3\}$, dostaneme

$$\begin{aligned}\#A_1 \cup A_2 \cup A_3 = \#A_1 + \#A_2 + \#A_3 \\ - \#A_1 \cap A_2 - \#A_1 \cap A_3 - \#A_2 \cap A_3 \\ + \#A_1 \cap A_2 \cap A_3,\end{aligned}$$

jak jsme očekávali.

Kterak káže obyčej, pár cvičení závěrem.

**Cvičení 3.4.1** (zase trocha teorie čísel)**.** Jeden z oblíbených a celkem rychlých rozkladů čísla na prvočísla je tzv. *Eratosthenovo síto*.

Funguje na principu vyškrtávání násobků čísel. Konkrétně, algoritmus prochází seznam čísel až do nějaké hranice, a kdykoli narazí na ještě neodškrtnuté číslo, odškrtne z tohoto seznamu všechny jeho násobky kromě něho samotného. Sami si rozmyslete, že tímto způsobem zůstanou ve výsledném seznamu pouze prvočísla.

My si tady rozmyslíme pouze zjednodušenou verzi. Spočtěte, kolik zůstane čísel mezi 1 a 1000 potom, co vyškrtáme všechny násobky čísel 2, 3, 5 a 7.

**Cvičení 3.4.2.** Určete počet přirozených čísel menších než 100, které nedělí druhá mocnina žádného přirozeného čísla (kromě 1).

**Cvičení 3.4.3.** Kolika způsoby lze seřadit do řady 5 Čechů, 4 Maďary a 3 Rusy, aby všichni příslušníci jednoho národa nikdy nestáli hned za sebou?

### 3.4.1 Problém šatnářky

Kapitolu o kombinatorickém počítání završíme klasickým *problémem šatnářky*. Jeho tradiční znění je následující.

V jeden chladný podvečer přijde skupina pánů do luxusního podniku. Před vstupem do lokálu si každý z nich odloží svoji černou buřinku (neb jiné se nenosí) v pánské šatně. Večírek se však neobyčejně prodlužuje a šatnářka, navíc ten den dosti nevyspalá, počne únavou podřimovati. Když ctihodní pánové konečně doflámují, sotva bdělá šatnářka jim při odchodu z šatny náhodně rozdá – jejíma zaslepenýma očima nerozlišitelné – černé buřinky. S jakou pravděpodobností odchází každý pán s cizí buřinkou?

Očíslujeme si pány přirozenými čísly od 1 do $n$ a jejich buřinky taktéž; čili pánovi číslo $i$ připadá buřinka číslo $i$. Náhodné rozdání buřinek odpovídá výběru permutace $\sigma \in S_n$, neboli bijektivního zobrazení $\sigma : \{1,\dots,n\} \to \{1,\dots,n\}$. Podle tvrzení 3.2.3 je počet těchto roven $n!$. Jednu konkrétní volbu rozloučení buřinek učiní tedy šatnářka ze zadání problému s pravděpodobností $1/n!$. Označíme-li si počet permutací, které žádné číslo nezobrazují na totéž, jako $š(n)$, pak řešením problému šatnářky je číslo $š(n)/n!$.

Tou netriviální otázkou zůstává, jak $š(n)$ určit. Postup, který se zde jmeme popsat, využívá „trik“ častý především v teorii pravděpodobnosti. Bývá totiž obvykle výrazně jednodušší spočítat počet objektů, které **splňují** nějakou vlastnost, raději než objektů, které onu vlastnost **nesplňují**. Oba postupy jsou logicky ekvivalentní, neboť počet objektů nesplňujících vlastnost $V$ je roven rozdílu počtu všech objektů a počtu těch, které vlastnost

$V$ splňují. Výpočetně se však postupy svou obtížností často výrazně liší.

Místo toho, abychom určovali počet permutací $\sigma \in S_n$, jež nezobrazí žádný prvek na sebe, budeme počítat permutace, které zobrazí aspoň jeden prvek na sebe. Takovému prvku se pak obvykle říká *pevný bod* dané permutace.

**Definice 3.4.2** (Pevný bod). Ať $\sigma \in S_n$. Prvek $i \in \{1,\dots,n\}$ nazveme *pevným bodem* permutace $\sigma$, pokud $\sigma(i) = i$.

Pro dané $i \in \{1,\dots,n\}$ si ještě označíme

$$\mathcal{P}_i := \{\sigma \in S_n \mid \sigma(i) = i\},$$

čili $\mathcal{P}_i$ je množina permutací, jejichž pevným bodem (**možná ne jediným!**) je $i$. Snadno si rozmyslíme, že množina všech permutací, které mají **aspoň jeden pevný bod**, je právě

$$\mathcal{P}_1 \cup \mathcal{P}_2 \cup \ldots \cup \mathcal{P}_n = \bigcup_{i=1}^{n} \mathcal{P}_i.$$

Vskutku, permutace $\sigma$ leží v $\bigcup_{i=1}^n \mathcal{P}_i$ právě tehdy, když existuje nějaké $i \in \{1,\dots,n\}$ takové, že $\sigma(i) = i$, což je z definice právě tehdy, když existuje nějaký její pevný bod. Čili, $\#\bigcup_{i=1}^n \mathcal{P}_i$ vyjadřuje počet permutací, které zobrazí aspoň jeden prvek na ten samý. Z diskuse výše plyne, že

$$š(n) = n! - \#\bigcup_{i=1}^{n} \mathcal{P}_i.$$

Problém šatnářky vyřešíme tím, že určíme $\#\bigcup_{i=1}^n \mathcal{P}_i$. Z věty 3.4.1 můžeme počítat

$$\#\bigcup_{i=1}^{n} \mathcal{P}_i = \sum_{I \subseteq \{1,\dots,n\}} (-1)^{\#I-1} \#\bigcap_{i \in I} \mathcal{P}_i. \qquad (\square)$$

Dalším krokem ke zdárnému řešení problému je uvědomění, že výraz $\#\bigcap_{i\in I} \mathcal{P}_i$ záleží pouze na $\#I$, nikoli na konkrétních indexech, které se v $I$ vyskytují. Přeci, $\#\bigcap_{i\in I} \mathcal{P}_i$ značí počet permutací s přesně $\#I$ pevnými body. Všimněme si, že ten je určen pouze samotným počtem pevných bodů, a ne jejich konkrétní hodnotou.

Toto pozorování napovídá, že v součtu na pravé straně ($\square$) můžeme sloučit ty sčítance, pro něž je $\#I$ stabilní. Řečeno přímočařeji, stačí, když budeme

sčítat pouze přes **velikosti** všech podmnožin, a nemusíme sčítat přes individuální podmnožiny, neboť na jejich konkrétních prvcích nezáleží (pouze na jejich velikostech).

Provedeme pro přehlednost ještě poslední přeznačení. Výraz $\mathsf{P}(k)$ bude značit *počet* permutací s přesně $k$ pevnými body. Díky předchozímu odstavci bychom též mohli být formální a psát, že

$$\mathsf{P}(k) = \sum_{\substack{I\subseteq\{1,\dots,n\}\\ \#I=k}} \#\bigcap_{i\in I}\mathcal{P}_i,$$

čili (dokonce správně) tvrdit, že počet permutací s $k$ pevnými body dostaneme tak, že sečteme počty permutací s pevnými body v podmnožině $I$ přes všechny $k$-prvkové podmnožiny $I$.

Nyní můžeme rovnost (□) přepsat do mnohem jednodušší podoby

$$\#\bigcup_{i=1}^{n}\mathcal{P}_i = \sum_{k=1}^{n}(-1)^{k-1}\mathsf{P}(k).$$

Opět přeloženo do lidské řeči říkáme, že počet permutací s aspoň jedním pevným bodem můžeme určit jako součet (až na znaménko) všech permutací s přesně $k$ pevnými body přes všechna $k$ od 1 do $n$. To dává smysl.

Dokážeme-li určit $\mathsf{P}(k)$, jsme hotovi. K tomu slouží následující lemma.

**Lemma 3.4.3.** Ať $k \in \mathbb{N}, k \leq n$. Platí

$$\mathsf{P}(k) = \binom{n}{k}(n-k)! = \frac{n!}{k!}. \qquad (\bullet)$$

**Důkaz.** Za důkaz vřele děkujeme laureátovi Fieldsovy medaile, prof. Mgr. et Ing. Jáchymovi Löwenhöfferu, RNDr. et Ph. D., CSc., DSc. Neodvažujeme se však přímo parafrázovat, tedy jej s odpuštěním mírně přeformulujeme.

Permutace s přesně $k$ libovolnými pevnými body je určena výběrem těchto $k$ pevných bodů z $n$ čísel a poté výběrem uspořádání zbývajících $n-k$ čísel. Výběr $k$ prvků z $n$ mohu podle tvrzení 3.3.2 učinit $\binom{n}{k}$ způsoby. Zvolit permutaci zbývajících $n-k$ prvků mohu podle tvrzení 3.2.3 $(n-k)!$ způsoby. Celkem tedy mohu určit jednu permutaci

na $n$ prvcích s $k$ pevnými body přesně $\binom{n}{k}(n-k)!$ způsoby. To dokazuje první rovnost v (•).

Ta druhá plyne z lemmatu 3.3.4, neboť

$$\binom{n}{k}(n-k)! = \frac{n!}{k!(n-k)!}(n-k)! = \frac{n!}{k!}. \qquad \square$$

Dopracovali jsme se k závěru problému. Díky předchozímu lemmatu máme po dosazení do (□) vzorec

$$\#\bigcup_{i=1}^{n}\mathcal{P}_i = \sum_{k=1}^{n}(-1)^{k-1}\frac{n!}{k!}.$$

Čili,

$$\check{s}(n) = n! - \#\bigcup_{i=1}^{n}\mathcal{P}_i = n! - \sum_{k=1}^{n}(-1)^{k-1}\frac{n!}{k!}.$$

Vytknutí čísla $n!$ dá snad hezčí vyjádření

$$\check{s}(n) = n!\sum_{k=0}^{n}(-1)^{k}\frac{1}{k!}.$$

Je triviální dokázat, že součet výše konverguje k $1/e$, kde písmeno $e$ značí tzv. Eulerovo číslo. Formálně zapsáno,

$$\sum_{k=0}^{n}(-1)^{k}\frac{1}{k!} \xrightarrow{n\to\infty} \frac{1}{e}.$$

Neformálně tento vztah říká, že čím je vyšší počet přišedších pánů do podniku, tím víc se pravděpodobnost, že šatnářka vrátí každému cizí buřinku, blíží k číslu $1/e$. Je tomu tak pro to, že

$$\frac{\check{s}(n)}{n!} = \frac{n!\sum_{k=0}^{n}(-1)^{k}\frac{1}{k!}}{n!} = \sum_{k=0}^{n}(-1)^{k}\frac{1}{k!} \xrightarrow{n\to\infty} \frac{1}{e},$$

kde, pamatujte, $\check{s}(n)/n!$ je přesně řešení problému šatnářky.

# 4 | Teorie grafů

Velkou část moderní matematiky (zcela jistě topologii, geometrii i algebru) tvoří studium „struktur“. Toto obecně nedefinované slovo obvykle značí množinu s nějakou další informací o vztahu mezi jejími prvky – tím obvykle bývá operace nebo třeba, jako v případě grafů, relace.

Tato kapitola zároveň značí jakýsi milník ve vývoji matematického myšlení, především algebraickým směrem. Můžeme se totiž začít bavit o speciálních zobrazeních, které zachovávají strukturu na množinách, mezi kterými vedou, tzv. *homomorfismech*; pochopit, že je dobré mít více popisů stejné struktury ekvivalentních v tom smyslu, že poskytují stejné množství informací, přestože se o žádné bijekci nedá formálně hovořit; uvidět, že je užitečné dva různé grafy (či obecně dvě různé struktury) považovat za stejné, když se liší pouze zanedbatelně.

Jednou, avšak zdaleka ne *jedinou*, motivací pro teorii grafů je schopnost analyzovat struktury tvořené množinou „uzlů“, mezi některými vedou „spojnice“. Takováto struktura úspěšně modeluje až neuvěřitelné množství přírodních i společenských úkazů. Mezi nimi jmenujmež

- návrhy elektrických obvodů, kde uzly jsou elektrická zařízení a spojnice jsou kabely mezi nimi vedoucí;
- lingvistické modely, kde uzly jsou slova a spojnice vede mezi těmi syntakticky souvisejícími;
- studium molekul, kde uzly jsou atomy a spojnice vazby mezi nimi;
- analýza šíření fámy v sociologii, kde uzly jsou lidské komunity a spojnice vede mezi komunitami s bezprostředním kontaktem.

Snad pro to, že uzly a spojnice grafu se obvykle kreslí jako body a úsečky v prostoru, ujaly se pro ně názvy *vrcholy* a *hrany* (jako v mnohoúhelnících), respektive. Struktura zvaná *graf* tedy sestává ze dvou údajů:

(1) množiny (obvykle konečné) vrcholů značené $V$ a

(2) množiny hran $E$, která je spjata s množinou vrcholů; toto „sepětí“ se však definuje různě, v závislosti na vkusu a aplikaci. My si ukážeme tři z jistě většího množství různých definic.

Asi prvním přirozeným kandidátem pro „strukturu“ je množina s relací. To je také první způsob, jak si budeme definovat pojem *graf*. Je to také ten nejobecnější v tom smyslu, že jeho pouze drobné modifikace nám umožní definovat i obdobné struktury, jež také zmateně slují *grafy*, byť s připojeným atributem.

Abychom úsečky mezi body mohli popsat jako relaci, čili pomocí uspořádaných dvojic bodů, zcela jistě budeme požadovat, aby nevedly úsečky z bodu do něho samého. Úsečku délky 0 lze totiž triviálně ztotožnit s bodem. Dále, úsečka z bodu $A$ do bodu $B$ je jistě tatáž, kterak úsečka z bodu $B$ do bodu $A$. Tento fakt musí rovněž relace $E$ odrážet.

První vlastností relace se, snad nepřekvapivě, říká *antireflexivita*. Čili, relace $E$ na $V$ je *antireflexivní*, když hrana $(v,v) \notin E$ pro každý vrchol $v \in V$.

Druhou vlastnost už jsme potkali a nazvali ji symetrií. Požadujeme, aby s hranou $(v,w) \in E$ obsahovala $E$ též hranu $(w,v) \in E$ pro každé dva vrcholy $v,w \in V$.

**Definice 4.0.1** (Graf poprvé)**.** Dvojici $G := (V,E)$, kde $V$ je konečná množina a $E$ je relace na $V$, nazveme *grafem*, pokud je $E$ **antireflexivní** a **symetrická**.

**Poznámka.** Z hlediska ryze formálního neodpovídá tato definice dokonale naší geometrické představě. My jsme totiž pouze požadovali, aby $E$ obsahovala jak úsečku z $v$ do $w$, tak úsečku z $w$ do $v$, ale nikoli, aby se jednalo o *tutéž* úsečku. Tedy, jedna úsečka mezi body je v množině $E$ reprezentována dvěma dvojicemi.

Nápravou by bylo definovat navíc ještě relaci $R$ na $E$, kde $(v,v')$ je v relaci $R$ s $(w,w')$ právě tehdy, když $(w,w') = (v',v)$ nebo $(w,w') = (v,v')$. Jinak řečeno, úsečka z bodu $v$ do bodu $v'$ je v relaci sama se sebou a s úsečkou z bodu $v'$ do bodu $v$.

Uvážíme-li pak jako hrany grafu $G$ nikoli množinu $E$, ale její třídy ekvivalence podle $R$ (**Ověřte, že $R$ je ekvivalence!**), dostaneme již přesnou množinovou paralelu bodů a úseček.

My však budeme v zájmu přehlednosti tento nedostatek ignorovat, protože není pro pochopení ani rozvoj teorie relevantní.

Cesta k druhé možné definici grafu není od první daleká. Stačí vlastně relaci $E$ interpretovat trochu jinak. Přece, antireflexivní a symetrická relace je „totéž“ jako množina dvouprvkových podmnožin $V$.

Vskutku, vezměme nějakou $E' \subseteq \binom{V}{2}$. Relaci $E$ z definice 4.0.1 sestrojíme tak, že z množiny $\{v,w\} \in E'$ vyrobíme dvojice $(v,w)$ a $(w,v)$. Protože prvky v množině nejsou uspořádané a nemohou se opakovat, dává tato konstrukce opravdu antireflexivní a symetrickou relaci. Vizuálně odpovídá rozdělení úsečky mezi $v$ a $w$ na šipku z $v$ do $w$ a šipku z $w$ do $v$.

Z druhé strany, mějme nějakou antireflexivní a symetrickou relaci $E$ na $V$. Protože s dvojicí $(v,w)$ obsahuje $E$ i dvojici $(w,v)$, můžeme z těchto dvojic ztvárnit množinu $\{v,w\}$. Relace $E$ je antireflexivní, čili se nemůže stát, že $v = w$, a množina $\{v,w\}$ je pročež vždy dvouprvková. Posbíráme-li všechny množiny $\{v,w\}$ do jedné velké množiny $E'$, bude platit $E' \subseteq \binom{V}{2}$. Vizuálně odpovídá tato konstrukce slepení šipky z $v$ do $w$ a šipky z $w$ do $v$ do jedné úsečky mezi $v$ a $w$.

**Výstraha.** Mezi množinami $E$ a $E'$ **nemůže existovat bijekce**, třeba jen pro to, že $\#E = 2\#E'$. Co konstrukce v předchozích dvou odstavcích ukazují, je pouze fakt, že pro naše účely definují $E$ a $E'$ stejnou strukturu na množině $V$.

Ovšem, uvážili-li bychom místo $E$ pouze třídy ekvivalence jejích prvků podle relace $R$ popsané v poznámce pod definicí 4.0.1, pak bychom skutečně tímto způsobem našli bijekci s množinou $E'$.

**Definice 4.0.2** (Graf podruhé)**.** Dvojici $G := (V, E')$, kde $V$ je konečná množina a $E' \subseteq \binom{V}{2}$, nazveme *grafem*.

Třetí pohled na hrany v grafu je více „kategoriální“. Zatímco množiny $E$ a $E'$ jsou závislé ve své definici na množině $V$, třetí množina hran $E''$, kterou si zde definujeme, bude libovolná konečná množina.

Tento popis grafové struktury bude odpovídat trochu jiné představě; konkrétně takové, kdy začínáme s množinou bodů $V$ a s množinou šipek $E$ (jež jsou od sebe naprosto odděleny) a oba konce každé šipky zapíchneme do dvou různých bodů z $V$. Toto „zapíchnutí“ realizují zobrazení $s, t : E'' \to V$ (z angl. **s**ource a **t**arget), která zobrazují šipky z $E''$ do bodů z $V$. Přičemž

budeme trvat na tom, aby $s(e) \neq t(e)$ pro všechny šipky $e \in E''$ a navíc, aby pro každou $e \in E''$ existovala šipka $e' \in E''$ taková, že $s(e) = t(e')$, $t(e) = s(e')$. Lidsky řečeno, nesmíme zapíchnout konce šipky do téhož vrcholu a, když zapíchneme začátek šipky do bodu $v$ a její konec do bodu $w$, pak musíme vzít další šipku, jejíž začátek zapíchneme do $w$ a konec do $v$.

Je snadné si rozmyslet, že z množiny šipek $E''$ zrekonstruujeme množinu $E$ z definice 4.0.1 tak, že z šipky $e \in E''$ vytvoříme dvojici $(s(e), t(e)) \in E$. Podmínky kladené na zobrazení $s$ a $t$ zaručují, že vzniklá množina $E$ je relace na $V$, která je antireflexivní a symetrická. V tomto případě dává uvedená konstrukce dokonce bijekci $E \cong E''$, čili jsme opět definovali tutéž strukturu na $V$.

Tato struktura bude zvlášť užitečná, až budeme probírat *toky v síti*.

**Definice 4.0.3** (Graf potřetí)**.** Čtveřici $G := (V, E'', s, t)$, kde $V$ a $E''$ jsou konečné množiny a $s$ a $t$ jsou zobrazení $E'' \to V$ nazveme *grafem*, pokud

(a) $s(e) \neq t(e)\ \forall e \in E''$ a

(b) $\forall e \in E''\ \exists e' \in E'' : s(e) = t(e') \wedge t(e) = s(e')$.

**Poznámka.** Opět, aby definice 4.0.3 odpovídala představě bodů a úseček (nebo oboustranných šipek), museli bychom definovat relaci $R$ na $E$ tak, aby $e$ a $e'$ byly v $R$, právě když $s(e) = s(e')$ a $t(e) = t(e')$ nebo $s(e) = t(e')$ a $t(e) = s(e')$. V takovém případě bychom mohli sestrojit bijekci mezi $E''$ a $E'$ z definice 4.0.2.

**Příklad.** Ať $V := \{1, 2, 3, 4, 5\}$ a

(1) $E := \{(1,2), (2,1), (1,3), (3,1), (1,4), (4,1), (1,5), (5,1), (2,3), (3,2), (4,5), (5,4)\}$;

(2) $E' := \{\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{4,5\}\}$;

(3) $E'' := \{e_1, e_2, e_3, e_4, e_5, e_6, e_1', e_2', e_3', e_4', e_5', e_6'\}$, kde

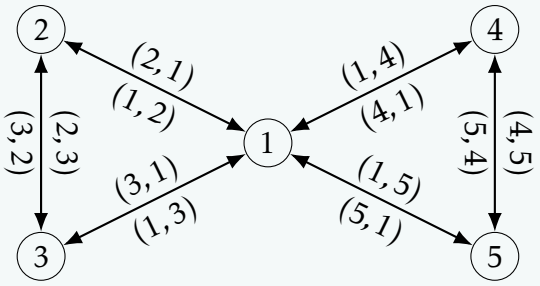
(a) $s(e_1) = s(e_2) = s(e_3) = s(e_4) = 1$, $s(e_5) = 2$, $s(e_6) = 4$,

(b) $t(e_1) = 2$, $t(e_2) = t(e_5) = 3$, $t(e_3) = 4$, $t(e_4) = t(e_6) = 5$ a

(c) $(s(e_i), t(e_i)) = (t(e'_i), s(e'_i))$ pro všechna $i \leq 6$.

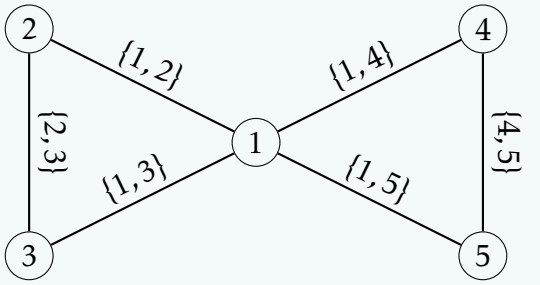
Není těžké nahlédnout, že $E$, $E'$ i $(E'', s, t)$ definují tutéž strukturu na $V$. Nakreslíme si grafy $G = (V, E), G' = (V, E')$ a $G'' = (V, E'', s, t)$. Přičemž hrany z $E$ budeme kreslit jako oboustranné šipky, ty z $E'$ jako prosté úsečky a ty z $E''$ rozdělíme na dvě protichůdné šipky, abychom vyjádřili rozdíly v interpretaci těchto grafových struktur.

Graf $G = (V, E)$ vypadá například takto. Pomněte, že například oboustranná šipka mezi vrcholy 1 a 2 představuje ve skutečnosti **dvě** dvojice – $(1, 2)$ a $(2, 1)$ z relace $E$.



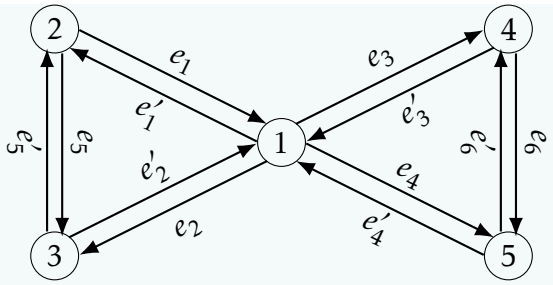

Obrázek 29: Graf jako množina $V$ s relací $E$.

Zcela stejně vypadá i graf $G' = (V, E')$.



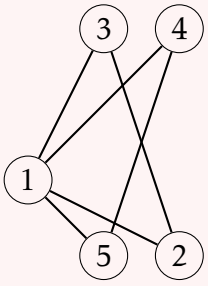

Obrázek 30: Graf jako množiny $V$ a $E' \subseteq \binom{V}{2}$.

Konečně, $G'' = (V, E'', s, t)$ můžeme načrtnout taktéž velmi podobně.




Obrázek 31: Graf jako množina $V$ s trojicí $(E'', s, t)$.

**Výstraha.** Grafová struktura je obecně zcela nezávislá na jejím nakreslení. Například graf $G = (V, E')$ z předchozího příkladu lze ekvivalentně vyobrazit třeba následovně.




Obrázek 32: Graf $G = (V, E')$ nakreslený jinak.

V následujícím textu spojíme všechny tři interpretace dohromady a pro $v, w \in V$ budeme hranu mezi $v$ a $w$ značit zjednodušeně jako $vw$. Pokud nehrozí nedorozumění, budeme pod tímto zápisem rozumět hranu, jejíž začátek je $v$ a konec $w$, čili $s(vw) = v$ a $t(vw) = w$. Avšak, kdykoli se nám to bude hodit, ztotožníme ji bez okolků s hranou $wv$ s obrácenými konci.

Tento neformální přístup k popisu hran se může zdát jako nebezpečný, ale jak uvidíme, ve skutečnosti velmi zjednodušuje zápis a újma na rigorozitě je obecně minimální. Kompletněji řečeno, hranou mezi dvěma vrcholy $v, w \in V$ myslíme buď dvojici $(v, w) \in E$ nebo dvojici $(w, v) \in E$ nebo množinu $\{v, w\} \in E'$ nebo prvek $e \in E''$ takový, že $s(e) = v$ a $t(e) = w$, nebo prvek $e' \in E''$ takový, že $s(e') = w$ a $t(e') = v$, a je nám to u ...

Obecně, v teorii grafů se velmi často pracuje s konečnými posloupnostmi (či $n$-ticemi, chcete-li) vrcholů a hran. Zavedeme proto zjednodušené zna-

čení $x_1x_2\cdots x_n$ pro uspořádanou $n$-tici $(x_1,\dots,x_n)$. Kdyby hrozil konflikt se zápisem součinu prvků $x_1,\dots,x_n$, samozřejmě tento úzus dočasně opustíme.

**Cvičení 4.0.1.** Nakreslete graf $G=(V,E)$, kde

- $V=\{1,\dots,5\}, E=\{\{1,2\},\{1,3\},\{1,5\},\{2,3\},\{3,4\},\{4,5\}\}$.
- $V=\{1,\dots,5\}$, $E=\binom{V}{2}$.
- $V=\{1,\dots,8\}$, $E=\{e_1,\dots,e_8\}$ a
  - $t(e_i)=s(e_{i+1})=i+1$ pro všechna $i\leq 7$,
  - $t(e_8)=s(e_1)=1$.

**Cvičení 4.0.2.** Popište všechny grafy $G=(V,E)$, kde $E$ je relace na $V$, která je antireflexivní, symetrická (to je součástí definice grafu) a navíc **transitivní**.

**Cvičení 4.0.3.** Ať $V$ je konečná množina a $E$ je relace na $V$, která je antireflexivní a symetrická. Definujme navíc na $E$ další relaci $\sim$ předpisem

$$(v,v')\sim(w,w')\Leftrightarrow(v,v')=(w,w')\vee(v,v')=(w',w).$$

Dokažte, že pak existuje bijekce mezi $[E]_\sim$ a množinou

$$E':=\{\{v,v'\}\mid(v,v')\in E\},$$

čili mezi množinou tříd ekvivalence $E$ podle $\sim$ a množinou, kterou dostanu tak, že z uspořádaných dvojic v $E$ udělám neuspořádané dvojice, tj. dvouprvkové podmnožiny. Pro intuici vizte poznámku pod definicí 4.0.1.

**Cvičení 4.0.4.** Spočtěte, kolik existuje grafů na $n$ vrcholech.

## 4.1 Pohyb v grafu

Jak jsme zmínili na začátku kapitoly, mnoho aplikací teorie grafů využívá interpretace této struktury jako množiny „uzlů“ se „spojnicemi“, po kterých se dá mezi uzly pohybovat. V této podsekci dáme pohybu po spojnicích formální tvář.

Nejzákladnějším typem pohybu v grafu je libovolná posloupnost vrcholů se zcela přirozenou podmínkou, že mezi vrcholy v posloupnosti bezprostředně za sebou musí vést hrana. Takové posloupnosti se říká *sled* (angl. *walk*).

**Definice 4.1.1** (Sled poprvé). Ať $G = (V, E)$ je graf. Posloupnost vrcholů $v_1 v_2 \cdots v_n$ nazveme *sledem* v grafu $G$, pokud $v_i v_{i+1} \in E$ pro každý index $i \in \{1, \ldots, n-1\}$.

Je však užitečné si uvědomit, že každý sled lze ekvivalentně definovat jako posloupnost navazujících hran. Tento pohled (jak uvidíme později v kapitole) má své aplikace – často nás totiž zajímá, po kterých spojnicích chodíme, spíše než které uzly procházíme.

Máme-li sled $v_1 v_2 \cdots v_n$, tak každá dvojice $v_i v_{i+1}$ musí být hranou v $G$. To ovšem znamená, že zcela identickou informaci poskytuje i posloupnost hran $e_1 e_2 \cdots e_{n-1}$, kde $e_i = v_i v_{i+1}$.

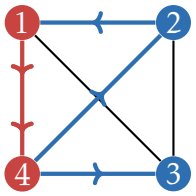
**Poznámka.** Tento princip, který prostupuje matematické struktury, je až překvapivě zásadního významu. Zobecníme-li trochu předchozí pozorování, uvědomíme si, že množina hran vlastně už v sobě obsahuje informaci o všech prvcích množiny $V$. Tedy bychom dokonce mohli definovat graf pouze jako množinu hran a množinu vrcholů bychom takto získali automaticky. Opak samozřejmě není pravdou.

Situace, kdy struktura na množině (či něčem složitějším) je dostatečně „hustá“, aby obsahovala kompletní informaci o této množině, je obecně velmi žádaná, jelikož struktura je často konstruována systematicky (a tedy jí rozumíme lépe) ve srovnání s náhodnou volbou její bázové množiny.

Pro nedostatek představivosti uvedeme příklad z homologické alge-

bry, kde injektivní a projektivní rezolventy modulů obsahují již kompletní informaci o daném modulu. Tedy člověku pro pochopení teorie modulů z homologického hlediska „stačí“ studovat injektivní a projektivní moduly, které jsou z definice více omezené než moduly obecné, pročež snadněji popsatelné.

**Definice 4.1.2** (Sled podruhé)**.** Ať $G = (V, E)$ je graf. Posloupnost hran $e_1 e_2 \cdots e_n$ nazveme *sledem* v grafu $G$, pokud $t(e_i) = s(e_{i+1})$ pro všechna $i \in \{1, 2, \ldots, n-1\}$.




Obrázek 33: Příklad sledu 142143 v grafu. Jednou navštívené hrany a vrcholy jsou značené modře, dvakrát navštívené červeně.

Méně obecným pohybem v grafu je sled, ve kterém se nesmějí opakovat hrany. Takové sledy lze najít často třeba v běžné situaci, kdy si jako rozvůzce jídla plánujete cestu městem. Jako uzly si označíte například místa, která musíte objet, a hrany budou nejkratší trasy mezi nimi. Projet přes některá místa vícekrát vám příliš vadit nemusí, ale ztrácet čas cestováním po stejné trase sem a tam byste neradi.

Takovému sledu se říká *tah* (angl. *trail*). Jeho definice je velmi přirozená.

**Definice 4.1.3** (Tah)**.** Ať $G = (V, E)$. *Tahem* v grafu $G$ nazveme buď

(1) sled (vrcholů) $v_1 v_2 \cdots v_n$ takový, že $v_i v_{i+1} \neq v_j v_{j+1}$ pro všechna $i \neq j$, nebo

(2) sled (hran) $e_1 e_2 \cdots e_{n-1}$ takový, že $e_i \neq e_j$ pro všechna $i \neq j$.

**Výstraha.** Obě uvedené definice tahu jsou v našem (částečně neformálním) pojetí hran skutečně ekvivalentní. Napíšeme-li totiž $v_i v_{i+1} \neq$

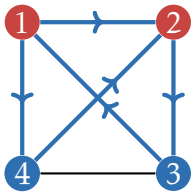
$v_j v_{j+1}$ myslíme tím vlastně dvě nerovnosti:

$$(v_i, v_{i+1}) \neq (v_j, v_{j+1}) \wedge (v_i, v_{i+1}) \neq (v_{j+1}, v_j);$$

nebo zápis můžeme též chápat jako

$$\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\},$$

čili každou hranu chceme projít (kterýmkoli směrem) maximálně jednou.




Obrázek 34: Příklad tahu 142312 v grafu. Jednou navštívené hrany a vrcholy jsou značené modře, dvakrát navštívené červeně.

Posledním, a asi nejčastěji zkoumaným, typem pohybu grafem je *cesta* (angl. *path*), což je sled, ve kterém se nesmějí opakovat ani hrany ani vrcholy. Pro jednoduchost je užitečné si uvědomit, že z podmínky neopakování vrcholů automaticky plyne podmínka neopakování hran. Pokud bychom totiž chtěli po nějaké hraně přejít dvakrát, tak zároveň dvakrát projdeme její koncové vrcholy. Definovat cestu pomocí sledu vrcholů je tudíž přímočaré.

U sledu hran je to však horší. Protože každá hrana definuje dva vrcholy sledu, je třeba zařídit, aby se oba koncové body každé hrany lišily od obou koncových bodů jiné hrany, ale jenom tehdy, když hrany nejdou bezprostředně za sebou. To bohužel vede na trochu neintuitivní definici cesty přes hrany. Totiž, cesta obsahující $n$ hran nutně obsahuje $n+1$ vrcholů, protože po sobě jdoucí hrany vždy sdílejí jeden vrchol. Rafinovaně se tedy cesta přes hrany dá vyjádřit jako tah, kde sjednocení přes všechny hrany (vnímané tentokrát jako množiny) má velikost právě $n+1$.

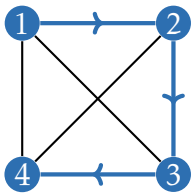
Studium cest má aplikace například právě v návrhu elektrických obvodů, kdy zcela jistě nechcete, aby do připojených zařízení šel proud z více, než jednoho místa.

**Definice 4.1.4** (Cesta). Ať $G = (V, E)$ je graf. *Cestou* v grafu $G$ nazveme buď

(1) sled (vrcholů) $v_1 v_2 \cdots v_n$ takový, že $v_i \neq v_j$ pro všechna $i \neq j$, nebo

(2) tah (hran) $e_1 e_2 \cdots e_{n-1}$ takový, že

$$\#\bigcup_{i=1}^{n-1} e_i = n.$$




Obrázek 35: Příklad cesty 1234 v grafu. Navštívené hrany a vrcholy jsou značené modře.

Posledním důležitým konceptem v grafu je tzv. *cyklus* (též *kružnice*, angl. *cycle*). Jedná se vlastně o „téměř cestu“, která končí tam, kde začala. Konkrétně je to tedy cesta prodloužená o svůj první vrchol (samozřejmě automaticky předpokládáme, že existuje hrana z posledního vrcholu do prvního).

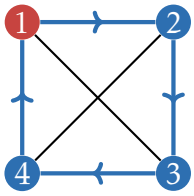
**Definice 4.1.5** (Cyklus). Ať $G = (V, E)$ je graf. *Cyklem* v grafu nazveme buď

(1) sled (vrcholů) $v_1 v_2 \cdots v_n v_1$, pokud $v_1 v_2 \cdots v_n$ je cesta v $G$, nebo

(2) tah (hran) $e_1 e_2 \cdots e_{n-1} e_n$, kde $t(e_n) = s(e_1)$ a $e_1 e_2 \cdots e_{n-1}$ je cesta v $G$.

Příklad cyklu vidíte na obrázku 36. Jejich význam zatím necháme zahalen tajemstvím, jež má být odkryto v nejvíce dramatickou chvíli.

**Cvičení 4.1.1.** Ať $\mathcal{P}_1 = e_1^1 e_2^1 \cdots e_n^1$ a $\mathcal{P}_2 = e_1^2 e_2^2 \cdots e_n^2$ jsou cesty. Za předpokladu, že $t(e_n^1) = s(e_1^2)$, definujeme jejich *sloučení*, které zapíšeme třeba




Obrázek 36: Příklad cyklu 12341 v grafu. Jednou navštívené hrany a vrcholy jsou značené modře, dvakrát navštívené červeně.

jako $\mathcal{P}_1 \oplus \mathcal{P}_2$, přirozeně jako posloupnost hran

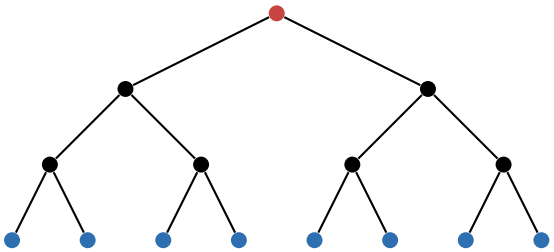
$$\mathcal{P}_1 \oplus \mathcal{P}_2 := e_1^1 e_2^1 \cdots e_n^1 e_1^2 e_2^2 \cdots e_n^2.$$

Určete, pro jaké cesty (v obecném grafu) $\mathcal{P}_1, \mathcal{P}_2$ platí, že

- $\mathcal{P}_1 \oplus \mathcal{P}_2 = \mathcal{P}_2 \oplus \mathcal{P}_1$;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ cesta;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ tah;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ sled;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ cyklus.

## 4.2 Stromy

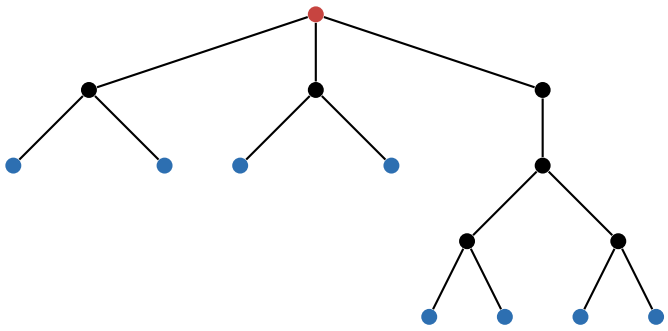
Chvíli se budeme bavit o stromech – ano, těch s listy a kořenem. Stromy jsou speciální typy grafů, které se takto nazývají ne nadarmo. Jsou to totiž grafy, u kterých si člověk může zvolit jakýsi „počáteční“ vrchol (zvaný *kořen*), z nějž se po cestě (ve smyslu definice 4.1.4) vždy dostane do jednoho z „koncových“ vrcholů, tzv. *listů*. Příklad stromu je na obrázku 37.



Obrázek 37: Příklad stromu. Kořen je značen červeně a listy modře.

**Výstraha.** Listy stromu jsou určeny jednoznačně jeho strukturou (jsou to ty jediné vrcholy, do nichž cesty od kořene mohou pouze vést a nikoli jimi procházet). Za kořen lze však volit libovolný vrchol, klidně i jeden z listů. Strom z obrázku 37 může proto vypadat i jak ukazuje obrázek 38.

Často není nutno o kořenu a listech stromu hovořit, pokud je v konkrétní situaci irelevantní rozlišovat jednotlivé vrcholy. Součástí definice stromu (kterou si záhy odvodíme) kořen ani listy nejsou.



Obrázek 38: Strom z obrázku 37 s jinou volbou kořene.

Nyní si rozmyslíme dvě ekvivalentní definice stromu.

Za první podmínku, abychom mohli graf nazvat stromem, budeme považovat fakt, že od kořene se dá dostat po hranách do každého z listů. Ekvivalentně, že z každého vrcholu se dá cestou dostat do každého vrcholu, protože za kořen lze, jak jsme nahlédli, volit kterýkoli vrchol, a cestu z kořene do vrcholu můžeme zkrátit tak, aby končila v nějakém vrcholu, jímž původně procházela. Grafy splňující tuto podmínku slují *souvislé*.

**Definice 4.2.1** (Souvislý graf)**.** Graf $G = (V, E)$ nazveme *souvislým*, pokud pro každé dva vrcholy $v, w \in V$ existuje cesta z $v$ do $w$, tedy cesta $v_1 v_2 \cdots v_n$, kde $v_1 = v$ a $v_n = w$.

Samotný název „strom“ plyne z faktu, že se jako graf pouze „větví“, čímž míníme, že při cestě směrem od (libovolného) kořene se jeden může pouze přibližovat k listům, ale nikoli se dostat zpět blíže ke kořeni. To lze snadno zařídit tak, že zakážeme cykly. Totiž, neexistuje-li v grafu cyklus, pak se po libovolné cestě ze zvoleného vrcholu můžeme od tohoto vrcholu pouze vzdalovat. Takové grafy nazveme, přirozeně, *acyklické*.

**Definice 4.2.2** (Acyklický graf)**.** Graf $G = (V, E)$ nazveme *acyklický*, pokud neobsahuje cyklus o aspoň třech vrcholech (samotné vrcholy jsou totiž z definice vždy cykly).

**Definice 4.2.3** (Strom)**.** Graf $G = (V, E)$ nazveme *stromem*, je-li souvislý a acyklický.

Na začátku sekce jsme slíbili ještě ekvivalentní definici stromu; ta činí začnou část důvodu užitečnosti stromů, především v informatice.

Ukazuje se totiž, že neexistence cyklů spolu se souvislostí způsobují, že mezi dvěma vrcholy vede vždy **přesně jedna cesta**. Po chvíli zamyšlení snad toto nepřichází jako nijak divoké tvrzení. Přeci, pokud by mezi vrcholy vedly cesty dvě, pak vrchol, kde se rozpojují, a vrchol, kde se opět spojují, by byly součástí cyklu uvnitř stromu, který jsme výslovně zakázali. Třeba překvapivější je fakt, že platí i opačná implikace.

**Tvrzení 4.2.4** (Ekvivalentní definice stromu)**.** Graf $G = (V, E)$ je stromem ve smyslu definice 4.2.3 právě tehdy, když mezi každými dvěma

vrcholy $G$ vede přesně jedna cesta.

**Důkaz.** Dokazujeme dvě implikace. Obě budeme dokazovat v jejich *kontrapozitivní* formě, tedy jako obrácenou implikaci mezi negacemi výroků. Lidsky, dokážeme, že (1) když existují vrcholy, mezi kterými nevede žádná nebo vede více než jedna cesta, pak $G$ není strom, a (2) když $G$ není strom, tak existují vrcholy, mezi kterými nevede žádná cesta nebo vede více než jedna.

(1) Pokud existují vrcholy, mezi kterými nevede cesta, pak $G$ není souvislý, což odporuje definici stromu. Budeme tedy předpokládat, že existují vrcholy $v, w$, mezi kterými vedou různé cesty $v_1 \cdots v_n$ a $v'_1 \cdots v'_m$, kde $v_1 = v'_1 = v$ a $v_n = v'_m = w$. Myšlenka důkazu je najít cyklus obsahující vrchol, kde se cesty rozdělují, a vrchol, kde se opět spojují. Vizte obrázek 39a.

Ať $r$ (od **r**ozpojení) je **největší** index takový, že $v_i = v'_i$ pro všechna $i \leq r$ (čili $v_r = v'_r$ je vrchol, ve kterém se cesty rozpojují). Ten určitě existuje, protože cesty se v nejhorším případě dělí už ve vrcholu $v = v_1$.

Podobně, ať $s$ (od **s**pojení) je **nejmenší** index takový, že existuje $k \in \mathbb{Z}$ splňující $v_j = v'_{j+k}$ pro všechna $j \geq s$. Čili, vrchol $v_s = v'_{s+k}$ je vrchol, ve kterém se cesty opět spojily. Ovšem, mohlo se tak stát v okamžiku, kdy jsme po jedné cestě prošli více nebo méně vrcholů než po druhé – tento počet vyjadřuje ono číslo $k$. Takový vrchol jistě existuje, v nejhorším je to přímo koncový vrchol $w = v_n$.

Zřejmě platí $r < s$, jinak by cesty nebyly různé. Potom je ovšem například posloupnost vrcholů

$$(v_r, v_{r+1}, \ldots, v_s = v'_{s+k}, v'_{s+k-1}, \ldots, v'_r = v_r)$$

cyklem v $G$. Tedy ani v tomto případě $G$ není strom.

(2) Když $G$ není strom, tak není souvislý nebo obsahuje cyklus. Když $G$ není souvislý, tak existují vrcholy, mezi nimiž nevede v $G$ cesta, což protiřečí podmínce, aby mezi každým párem vrcholů vedla přesně jedna.

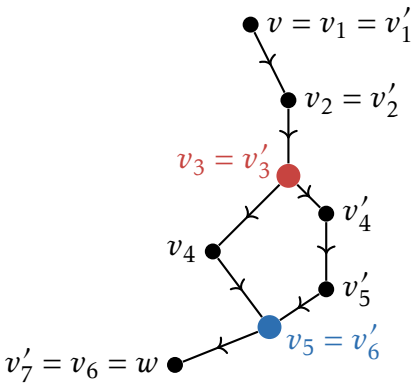
Budeme tedy předpokládat, že $G$ obsahuje cyklus $v_1 v_2 \cdots v_n$ (tedy $v_n = v_1$ a $n \geq 3$). Pak ovšem pro libovolné indexy $i < j \leq n$ jsou

posloupnosti

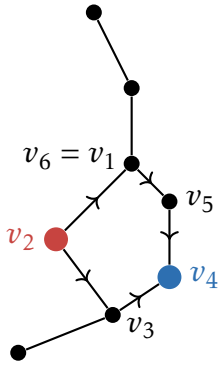
$$(v_i, v_{i+1}, \ldots, v_j) \quad \text{a} \quad (v_i, v_{i-1}, \ldots, v_1 = v_n, v_{n-1}, \ldots, v_j)$$

dvě různé cesty mezi $v_i$ a $v_j$. Vizte obrázek 39b.

Tím je důkaz dokončen. □




(a) Část (1) důkazu tvrzení 4.2.4. Zde $r = 3, s = 5$ a $k = 1$. Sestrojený cyklus je $v_3 v_4 v_5 v_5' v_4' v_3'$.




(b) Část (2) důkazu tvrzení 4.2.4. Zde $i = 2, j = 4$ a sestrojené cesty jsou $v_2 v_3 v_4$ a $v_2 v_1 v_5 v_4$.

Obrázek 39: Ilustrace k důkazu tvrzení 4.2.4.

**Cvičení 4.2.1.** Dokažte, že je-li $T = (V, E)$ strom, pak $\#E = \#V - 1$.

**Cvičení 4.2.2.** Spočtěte, kolik existuje stromů na $n$ vrcholech.

### 4.2.1 Minimální kostra

Ne všechny hrany jsou si rovny. Kterési se rodí krátké, jiné dlouhé; kterési štíhlé, jiné otylé; kterési racionální, jiné iracionální.

Často nastávají situace, kdy jeden potřebuje hranám grafu přiřadit nějakou hodnotu, obvykle číselnou, která charakterizuje klíčovou vlastnost této hrany. Při reprezentaci dopravní sítě grafem to může být délka silnice či její vytížení, při reprezentaci elektrických obvodů pak například odpor. V teorii grafů takové přiřazení hodnoty hranám grafu sluje *ohodnocení*.

**Definice 4.2.5** (Ohodnocený graf)**.** Ať $G = (V, E)$ je graf. Libovolné zobrazení $w : E \to \mathbb{R}^+ = (0, \infty)$ nazveme *ohodnocením* grafu $G$. Trojici $(V, E, w)$, kde $w$ je ohodnocení $G$, nazveme *ohodnoceným grafem*.

**Poznámka.** Každý graf $G = (V, E)$ lze triviálně ztotožnit s ohodnoceným grafem $(V, E, w)$, kde $w : E \to \mathbb{R}^+$ je konstantní zobrazení. Obvykle se volí konkrétně $w \equiv 1$, tedy zobrazení $w$ takové, že $w(e) = 1$ pro každou $e \in E$.

Porozumění struktuře ohodnocených grafů může odpovědět na spoustu zajímavých (jak prakticky tak teoreticky) otázek. Můžeme se kupříkladu ptát, jak se nejlépe (vzhledem k danému ohodnocení) dostaneme cestou z jednoho vrcholu do druhého. Slovo „nejlépe“ zde chápeme pouze intuitivně. V závislosti na zpytovaném problému můžeme požadovat, aby cesta třeba minimalizovala či maximalizovala součet hodnot všech svých hran přes všechny možné cesty mezi danými vrcholy. Jsou však i případy, kdy člověk hledá cestu, která je nejblíže „průměru“.

Abychom pořád neříkali „součet přes všechny hrany cesty“, zavedeme si pro toto často zkoumané množství název *váha cesty*. Čili, je-li $\mathcal{P} := e_1 \cdots e_n$ cesta v nějakém ohodnoceném grafu $G$, pak její vahou rozumíme výraz

$$w(\mathcal{P}) := \sum_{i=1}^{n} w(e_i).$$

Zápis $w(\mathcal{P})$ můžeme vnímat buď jako zneužití zavedeného značení, nebo jako fakt, že jsme zobrazení $w$ rozšířili z množiny všech hran na množinu všech cest v grafu $G$ (kde samotné hrany jsou z definice též cesty).

**Poznámka.** Záměrně jsme užili slovního spojení *váha* cesty místo snad přirozenějšího *délka* cesty. V teorii grafů se totiž délkou cesty myslí obyčejně počet hran (nebo vrcholů), které obsahuje. Délka cesty $\mathcal{P} = e_1 \cdots e_n$ je tudíž $n$ (resp. $n+1$), bo obsahuje $n$ hran (resp. $n+1$ vrcholů).

Tento úzus svědčí účelu předchozí poznámky. Pokud totiž každý graf bez ohodnocení vnímáme vlastně jako ohodnocený graf, kde každá hrana má váhu přesně 1, pak váha každé cesty je rovna její délce.

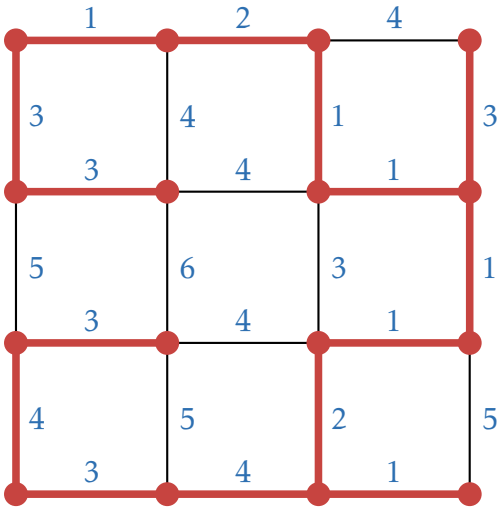
První (a nejjednodušší) problém, kterým se budeme zabývat, je nalezení

*minimální kostry* (angl. *spanning tree*).

**Definice 4.2.6** (Minimální kostra)**.** Ať $G = (V, E, w)$ je **souvislý** ohodnocený graf. Ohodnocený graf $K = (V', E', w)$ nazveme *minimální kostrou* grafu $G$, pokud je souvislý, $V' = V$ (tedy $K$ obsahuje všechny vrcholy $G$), $E' \subseteq E$ a

$$\sum_{e \in E'} w(e)$$

je minimální vzhledem ke všem možným volbám podmnožiny $E' \subseteq E$. Lidsky řečeno, graf $K$ spojuje všechny vrcholy $G$ tím „nejlevnějším“ způsobem vzhledem k ohodnocení $w$.




Obrázek 40: Minimální kostra grafu s ohodnocením $w$.

**Pozorování.** Minimální kostra ohodnoceného grafu je strom.

**Důkaz.** Kdyby minimální kostra nebyla strom, pak buď není souvislá, což jsme výslovně zakázali, nebo obsahuje cyklus. Tudíž se mezi nějakými dvěma vrcholy dá jít po více než jedné cestě, a proto můžeme přinejmenším jednu hranu z kostry odebrat. Protože každá hrana má kladné ohodnocení, snížili jsme tím součet hodnot všech hran. To je spor. □

**Důsledek 4.2.7.** Minimální kostra ohodnoceného stromu je s ním totožná.

Minimální kostra je zvlášť užitečná právě při návrhů elektrických obvodů, kdy je potřeba zařídit, aby všechna připojená zařízení čerpala co nejmenší množství energie. Protože elektřina proudí rychlostí světla, délka kabelu (pokud není zrovna mezigalaktický) nás příliš netrápí, ale právě odpor či kvalita/vodivost konkrétních spojů by mohly.

Další praktickou grafovou úlohou vedoucí na problém nalezení minimální kostry je potřeba spojit vzdálené servery. Korporace mají obvykle mnoho různých serverů rozmístěných po světě, jež spolu ale musejí sdílet data. Problém je v tom, že propojení mezi servery by nejen mělo vést k nejmenší možné prodlevě při přenosu dat (od toho **minimální**), ale nesmí ani obsahovat cykly (od toho **kostra**). Kdyby totiž cykly obsahovalo, pak by se při přenosu dat stalo, že by aspoň jeden server v tomto cyklu dostal aspoň dvakrát stejnou informaci z dvou různých zdrojů, ale v odlišný čas. Taková situace vede nezbytně dříve nebo později ke korupci dat; řekněme, když daný server už s obdrženou informací začal po přijetí provádět výpočet. Pro více detailů k tomuto využití minimálních koster vizte Spanning Tree Protocol.

### 4.2.2 Kruskalův algoritmus

Na problém nalezení minimální kostry souvislého ohodnoceného grafu existuje skoro až zázračně přímočarý algoritmus, pojmenovaný po americkém matematiku, Josephu B. Kruskalovi. Jeho základní myšlenkou je prostě začít s grafem $K$ obsahujícím všechny vrcholy z $V$ a přidávat hrany od těch s nejnižším ohodnocením po ty s nejvyšším tak dlouho, dokud nevznikne souvislý graf. Je potřeba pouze dávat pozor na cykly. K tomu stačí si pamatovat stromy (jako množiny vrcholů), které přidáváním hran vytvářím, a povolit přidání hrany jedině v případě, že spojuje dva různé stromy.

Pro zápis v pseudokódu vizte algoritmus 1. Manim s průběhem algoritmu s náhodně vygenerovaným hodnocením je k dispozici zde.

**Algoritmus 1:** Kruskalův algoritmus.

**input :** souvislý ohodnocený graf $G = (V, E, w)$, kde $V = \{v_1, \ldots, v_n\}$
**output:** množina hran $E'$ minimální kostry grafu $G$

*Inicializace*;
$E' \leftarrow \emptyset$;
*Množina hran, které nelze přidat (jinak by vznikl cyklus)*;
$X \leftarrow \emptyset$;
**for** $i \leftarrow 1$ **to** $n$ **do**
    *Každý strom nejprve obsahuje pouze jediný vrchol*;
    $T_i \leftarrow \{v_i\}$;
*Množina indexů pro pamatování sloučených stromů*;
$I \leftarrow \{1, \ldots, n\}$;
*Přidávám hrany, dokud mám pořád víc než jeden strom*;
**while** $\#I > 1$ **do**
    $e \leftarrow$ libovolná hrana s minimální $w(e)$, která není v $E'$ ani v $X$;
    $i \leftarrow$ index v $I$ takový, že $s(e) \in T_i$;
    $j \leftarrow$ index v $I$ takový, že $t(e) \in T_j$;
    **if** $i = j$ **then**
        *Hrana spojuje vrcholy ve stejném stromě, jejím přidáním by vznikl cyklus*;
        $X \leftarrow X \cup \{e\}$;
    **else**
        *Hrana spojuje různé stromy. Přidávám ji do kostry*;
        $E' \leftarrow E' \cup \{e\}$;
        $T_i \leftarrow T_i \cup T_j$;
        $I \leftarrow I \setminus \{j\}$;
**return** $E'$;

**Tvrzení 4.2.8.** Kruskalův algoritmus je korektní.

**Důkaz.** Potřebujeme ověřit, že

(1) algoritmus provede pouze konečný počet kroků;

(2) algoritmus vrátí správnou odpověď.

Případ (1) je zřejmý, protože $\#E < \infty$, čili algoritmus přidá do $E'$ pouze

konečně mnoho hran. Navíc, graf $G$ je z předpokladu souvislý, a tedy vždy existuje hrana $e$ spojující dva různé stromy $T_i$ a $T_j$.

V případě (2) uvažme, že $K = (V, E', w)$ není minimální kostra $G$. V takovém případě se mohlo stát, že

(a) $K$ není strom – tedy buď není souvislý nebo obsahuje cyklus;

(b) existuje podmnožina $E'' \subseteq E$ taková, že $K' = (V, E'', w)$ je strom a

$$\sum_{e \in E''} w(e) < \sum_{e \in E'} w(e).$$

Případ (a) lze vyloučit snadno, neboť souvislost $K$ plyne ihned ze souvislosti $G$, tedy, jak již bylo řečeno v bodě (1), vždy lze nalézt hranu spojující do té doby dva různé stromy. Pokud $K$ obsahuje cyklus, tak algoritmus musel v jednom kroku spojit hranou dva vrcholy ze stejného stromu, což je spor.

Pokud nastal případ (b), pak musejí existovat hrany $e'' \in E'' \setminus E'$ a $e' \in E' \setminus E''$ takové, že $w(e'') < w(e')$. Protože algoritmus zkouší přidávat hrany vždy počínaje těmi s nejmenší vahou, musel v nějakém kroku narazit na hranu $e''$ a zavrhnout ji. To ovšem znamená, že hrana $e''$ spojila dva různé vrcholy téhož stromu a graf $K' = (V, E'', w)$ obsahuje cyklus. To je spor s předpokladem, že $K'$ je strom. Tedy, taková podmnožina $E'' \subseteq E$ nemůže existovat a $K$ je vskutku minimální kostra $G$. □

**Výstraha.** Minimální kostra grafu **není jednoznačně určena**! Všimněte si, že na řádku 12 Kruskalova algoritmu volím **libovolnou** hranu, která má ze všech zatím nepřidaných nejnižší váhu a jejímž přidáním nevznikne cyklus.

**Poznámka.** V definici minimální kostry požadujeme, aby $G$ byl souvislý graf. Pokud tomu však tak není, je pořád možné zkonstruovat minimální kostru pro každou část $G$, která souvislá je (pro každou jeho tzv. *komponentu souvislosti*), zkrátka tím, že Kruskalův algoritmus spouštíme opakovaně.

**Cvičení 4.2.3** (Těžké ale fun)**.** Ať $V$ je množina $n$ bodů v rovině. Pro každé dva body $x, y \in V$ definujme váhu hrany $xy$ jako vzdálenost bodů $x$ a $y$. Čili, pokud $x = (x_1, x_2)$ a $y = (y_1, y_2)$, pak

$$w(xy) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

Vzniklý graf označme obyčejně $G$.

(a) Ukažte, že v každé minimální kostře $G$ vede z každého vrcholu maximálně 6 hran.

(b) Ukažte, že existuje minimální kostra $G$, jejíž hrany se (jakožto úsečky v rovině) nekříží.

**Cvičení 4.2.4.** *Úplným grafem* na $n$ vrcholech myslíme graf $G = (V, \binom{V}{2})$, tedy graf, mezi každým párem jehož vrcholů vede hrana. Takový graf se obvykle značí $K_n$. Najděte minimální kostru $K_n$ a spočtěte její váhu (tj, součet vah všech jejích hran), je-li $V = \{1, \ldots, n\}$ a

(a) $w(ij) = \max(i, j)$,

(b) $w(ij) = i + j$,

pro všechny páry $i, j \leq n$.

**Cvičení 4.2.5.** Dokažte, že když $w$ (tj. ohodnocení $G$) je prosté zobrazení, pak je minimální kostra $G$ určena jednoznačně.

## 4.3 Jordanovo centrum

V návaznosti na sekci o minimální kostře se rozhovoříme o jednom dalším optimalizačním problému – konkrétně hledání „centra“ ohodnoceného grafu.

Motivační úlohou je tzv. *facility location problem*, v přibližném překladu *úloha umístění střediska*. Jde o úlohu, kdy máte dánu dopravní síť sídlišť (obecně obydlených zón) a význačných uzlů, přes které se chtě nechtě musí jezdit (například velké křižovatky, Nuselák apod.). Hrany vedou mezi sídlišti či uzly, když od jednoho k druhému vede bezprostřední cesta (tedy cesta neprocházející žádným jiným sídlištěm nebo uzlem).

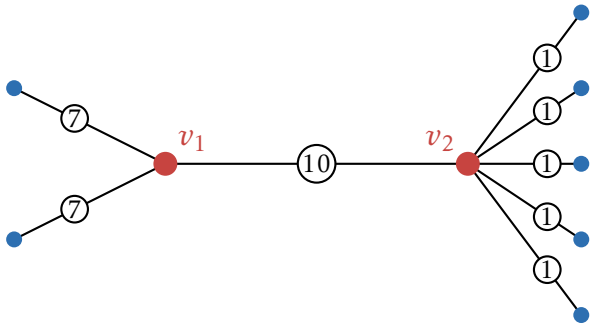
Pojďme si úlohu rozmyslet podrobně. Máme u nějakého uzlu v síti dopravních křižovatek postavit středisko. Bez ohledu na typ střediska nás pravděpodobně zajímá, aby se všichni z obydlených zón, které má toto středisko pokrýt, dostali k němu co nejsnáze. Ovšem, výraz „co nejsnáze“ je matematicky příliš vágní. Jistě hledáme optimální řešení, ale v jakém smyslu konkrétně?

Snad bude užitečné kýžený smysl optimality vyzkoumat na příkladě. Řekněme, že střediskem, jež potřebujeme umístit, je nemocnice. Co znamená, že je vzhledem k dané síti nemocnice *optimálně* umístěna? První, co by nás mohlo napadnout, je chtít, aby průměr vzdáleností od obydlených zón k nemocnici byl co nejmenší. To je přirozená myšlenka, ale jak si ihned rozmyslíme, vcelku morbidní.

U průměru totiž často nastává situace, že značný počet nízkých hodnot převáží nad zanedbatelným počtem hodnot vysokých. Uvažte síť uzlů a sídlišť danou grafem na obrázku 41, kde sídliště jsou značena modře, dopravní uzly červeně a ohodnocení hran, jak je vlastně zvykem, značí průměrnou dobu jízdy po těchto cestách.

V tomto případě by uzel $v_2$ jistě nabízel mnohem lepší průměrné řešení, neboť můžeme snadno spočítat, že průměrná doba jízdy od libovolného sídliště k němu je asi pět a půl minuty. Naopak, průměrná doba jízdy k uzlu $v_1$ činí těsně pod deset minut.

My se ale přesto rozhodneme postavit nemocnici v uzlu $v_1$. Proč? Totiž, při stavbě nemocnice nám nejde ani tolik o to, aby se do ní dostalo co nejvíce lidí co nejrychleji, ale **aby to nikdo neměl příliš daleko**. Kdybychom




Obrázek 41: Špatně vyvážená síť sídlišť a dopravních uzlů.

učinili opak a postavili nemocnici v uzlu $v_2$, pak by sice lidé ze sídlišť na pravé straně to měli do nejbližší nemocnice pouhou minutu, ale lidé z levých sídlišť by cestovali průměrně až sedmnáct minut. Do uzlu $v_1$ se dostane každý člověk nejpozději za jedenáct minut.

Tento způsob měření vhodnosti umístění nemocnice v dopravních sítích je skutečně v praxi používaný a je dozajista nepěkným příkladem volby menšího ze dvou zel. Tedy, není prioritou, aby se někomu dostalo pomoci velmi brzy, ale aby se nikomu nedostalo pomoci příliš pozdě.

Problém, jenž jsme právě zformulovali, je variantou výše zmíněného *facility location problem* (dále jen FLP), která sluje EFLP, tedy ***emergency*** *facility location problem*. Úlohou je nalézt takový uzel, jehož **maximální** vzdálenost ke všem ostatním uzlům je minimální, čili uzel pro umístění středisek jako jsou právě nemocnice a polikliniky, či obecněji „pohotovostní“ střediska.

Druhou známou variantou je SFLP, čili ***service*** *facility location problem* – úloha nalézt vhodný uzel pro umístění střediska „služeb“, kde je naopak žádoucí, aby nastala druhá z výše diskutovaných situací, aby se k němu přiblížilo co nejvíce lidí co nejrychleji. Ti, již cestují dlouho, budou na nejméně libý pád kalé nálady sedíce rozčileně ve voze a shrbení klejíce ustavičně v kolena, však nezhynou.

Formálně tedy hledáme uzel, který minimalizuje průměr všech vzdáleností od něj ke všem ostatním uzlům. Pro usnadnění výpočtu je dobré si uvědomit, že minimalizovat **průměr** všech vzdáleností je totéž, co minimalizovat **součet** všech vzdáleností, neboť počet uzlů se nemění (**Rozmyslete si to!**).

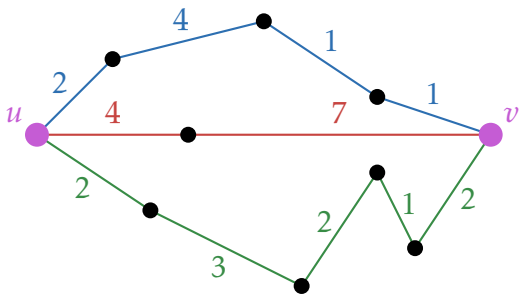
K formulaci obou úloh potřebujeme zavést základní pojem *vzdálenosti* mezi vrcholy v ohodnoceném grafu.

V zájmu strohosti vyjádření označíme pro libovolné dva vrcholy $u, v \in V$ grafu $(V, E, w)$ symbolem $\mathcal{P}(u, v)$ množinu všech cest mezi $u$ a $v$. Speciálně, $\mathcal{P}(v, v) = \{v\}$, čili cesta z vrcholu do něj samého obsahuje pouze tento jeden vrchol, a $\mathcal{P}(u, v) = \emptyset$, pokud mezi $u$ a $v$ nevede v $G$ cesta.

**Definice 4.3.1** (Vzdálenost v grafu)**.** Ať $G = (V, E, w)$ je ohodnocený graf a $v, w \in V$. *Vzdálenost* mezi $u$ a $v$ v grafu $G$, značenou $d_G(u, v)$ (z angl. **d**istance), definujeme jako

$$d_G(u, v) := \begin{cases} \min_{\mathcal{P} \in \mathcal{P}(u,v)} w(\mathcal{P}), & \text{pokud } \mathcal{P}(u, v) \neq \emptyset; \\ \infty, & \text{pokud } \mathcal{P}(u, v) = \emptyset. \end{cases}$$

Lidsky řečeno, vzdáleností mezi vrcholy je váha nejkratší cesty mezi nimi vedoucí, pokud taková existuje.




Obrázek 42: Zde $d_G(u, v)$ je váha nejkratší, to jest modré, cesty.

Abychom našli pro daný graf $G = (V, E, w)$ řešení EFLP, musíme najít takový vrchol, který minimalizuje největší možnou vzdálenost od něj ke všem ostatní vrcholům $G$. Takový vrchol (nebo vrchol**y**?) nazveme *Jordanovým centrem* grafu $G$, po žabožroutím počtáři, Marie E. C. Jordanovi.

Samozřejmě, nezajímá-li nás vzdálenost ke **všem** vrcholům, jako je tomu v příkladě umístění nemocnice, uvážíme pouze maximum vzdáleností k relevantním vrcholům (tedy k sídlištím). Na principu úlohy to nic nemění.

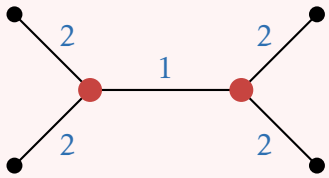
Formálně, *excentricita* vrcholu $v \in V$ je kvantita $e(v) := max_{u \in V} d_G(v, u)$, tedy maximum přes všechny vzdálenosti od něj k ostatním vrcholům. Toto číslo vyjadřuje, jak moc je vrchol vzdálen od „ideálního centra“ grafu,

tedy od bodu, od kterého by každý vrchol byl stejně daleko. Samozřejmě, toto ideální centrum málokdy existuje, takže hledáme pouze vrchol s nejmenší excentricitou, s nejmenší *odchylkou* od centra.

**Definice 4.3.2** (Emergency Facility Location Problem). Ať $G = (V, E, w)$ je **souvislý** ohodnocený graf. Úlohu nalézt vrchol s minimální excentricitou nazveme EFLP. Jejím *řešením* je vrchol s touto vlastností, tedy vrchol $c \in V$ splňující

$$e(c) = \min_{v \in V} e(v).$$

**Výstraha.** Řešení EFLP **není jednoznačně určeno**! Vizte např. graf na obrázku 43.




Obrázek 43: Vrcholy s minimální excentricitou v grafu $G = (V, E, w)$.

**Definice 4.3.3** (Jordanovo centrum). Množinu všech řešení EFLP pro graf $G = (V, E, w)$ nazýváme *Jordanovým centrem* grafu $G$.

**Definice 4.3.4** (Poloměr grafu). Je-li $c$ vrchol v Jordanově centru grafu $G = (V, E, w)$, pak hodnotu $e(c)$ nazýváme *poloměrem* grafu $G$ a značíme ji $\rho(G)$.

**Poznámka.** V definici EFLP jsme požadovali, aby byl graf souvislý. To z ryze technického hlediska není nutné, protože excentricita vrcholu je definována i pro nesouvislý graf. Uvědomme si ale, že pro nesouvislý graf je excentricita každého vrcholu rovna $\infty$, tedy Jordanovým centrem je celý graf a úloha poněkud pozbývá smyslu.

Obdobným způsobem si formalizujeme i SFLP.

Nyní chceme nalézt vrchol, který minimalizuje průměr (nebo ekvivalent-

ně součet) vzdáleností od něj k, buď všem nebo pouze zajímavým, vrcholům. Názvosloví zde poněkud selhává a tomuto součtu přes vzdálenosti ke všem vrcholům se rovněž často přezdívá *excentricita*. Abychom pojmy odlišili, slovo „excentricita“ přeložíme a budeme říkat „výstřednost“. Tedy, *výstředností* vrcholu $v \in V$ v ohodnoceném grafu $G = (V, E, w)$ myslíme číslo

$$e'(v) := \sum_{u \in V} d_G(v, u).$$

**Definice 4.3.5** (Service Facility Location Problem)**.** Ať $G = (V, E, w)$ je souvislý ohodnocený graf. Úlohu nalézt vrchol s minimální výstředností nazveme SFLP. Jejím řešením je vrchol $c' \in V$ s touto vlastností, tedy takový, že

$$e'(c') = \min_{v \in V} e'(v).$$

Pochopitelně, stejně jako EFLP, i SFLP může mít více řešení. Avšak, podle našeho nejlepšího vědomí se množině řešení SFLP nijak význačně neříká. Minimální výstřednost se občas nazývá *status* grafu $G$. Etymologie tohoto názvosloví je spjata s novodobým využitím SFLP při vyvažování nervových sítí a její zevrubné objasnění je nad rámec tohoto textu.

**Definice 4.3.6** (Status grafu)**.** Ať $G = (V, E, w)$ je souvislý ohodnocený graf. Když $c' \in V$ je řešení SFLP, pak se hodnota $e'(c)$ nazývá *status* grafu $G$ a značí se $\sigma(G)$.

**Výstraha.** Řešení EFLP a řešení SFLP mohou (ale **nemusí**) být disjunktní. Vrátíme-li se ke grafu na obrázku 41, pak řešením EFLP je pouze vrchol $v_1$, zatímco řešením SFLP je pouze vrchol $v_2$.

Následující podsekci dedikujeme obecnému algoritmu, pomocí nějž lze také hledat řešení jak EFLP, tak SFLP. Poznamenáme však, že existují mnohem efektivnější algoritmy, jež naleznou řešení těchto úloh; tyto ale vyžadují o poznání hlubší poznatky teorie grafů.

### 4.3.1 Floydův-Warshallův algoritmus

Vlastně jedinou výzvou na cestě k vyřešení FLP je umět aspoň rámcově efektivně najít vzdálenost mezi všemi dvojicemi vrcholů. Bohužel, žádná pěkná věta jako Pythagorova, která umožňuje okamžitě počítat vzdálenosti bodů v Eukleidovských prostorech, v teorii grafů neexistuje a existovat nemůže.

Připomeňme, že v ohodnoceném grafu $G = (V, E, w)$ je vzdálenost vrcholů $u, v \in V$ definována jako váha nejkratší cesty. Poznamenejme, že zde slovo „nejkratší“ chápeme ve smyslu *váhy* cesty (tedy součtu vah všech jejích hran), nikoli ve smyslu *délky* cesty (tedy počtu jejích hran). Asi bychom měli říkat „nejlehčí“ cesta, to je ale poněkud nepřirozené...

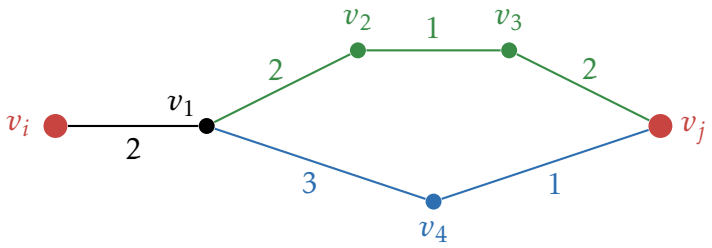
Floydův-Warshallův algoritmus nesouvisí přímo s FLP. Je to algoritmus, který nalezne vzdálenost (tedy váhu nejkratší cesty) mezi všemi dvojicemi vrcholů. Je však zřejmé, jak znalost této informace vede okamžitě k řešení EFLP, resp. SFLP. V moment, kdy známe vzdálenost každého vrcholu od každého, stačí pouze spočítat excentricitu, resp. výstřednost, každého vrcholu a vybrat pouze ty s minimální.

Jistě není překvapením, že zkoušet z každého vrcholu všechny možné cesty do všech ostatních vrcholů a z nich vybírat ty nejkratší, není dvakrát efektivní. Floydův-Warshallův algoritmus stojí na dvou principech, jimž se především v programování říká *rekurze* a *dynamické programování*. Jejich úplné pochopení a nabytí schopnosti využívat může být časově náročné, ale Floydův-Warshallův algoritmus jich využívá velmi přímočaře. Postupně si rozebereme, že ze znalosti váhy nejkratší cesty mezi dvěma vrcholy, **která využívá jen nějakou podmnožinu ostatních vrcholů**, lze zvětšováním této podmnožiny získat nakonec váhu nejkratší cesty mezi těmito vrcholy v celém grafu (odtud *rekurze*). Dále, ze znalosti vzdálenosti mezi určitými dvojicemi vrcholů můžeme rychle určit vzdálenost mezi párem, u kterého jsme ji zatím neznali (odtud *dynamické programování*).

Naší prací v této podsekci je dát předchozímu odstavci formální podobu. Ať $G = (V, E, w)$ je souvislý ohodnocený graf, kde $V = \{v_1, \ldots, v_n\}$. Definujme zobrazení $\delta : \{1, \ldots, n\}^3 \to \mathbb{R}^+$ následovně. Ať $\delta(i, j, k)$ je váha nejkratší cesty mezi $v_i$ a $v_j$ **využívající pouze vrcholy z podmnožiny vrcholů** $\{v_1, \ldots, v_k\}$ (samozřejmě, kromě počátečního $v_i$ a koncového $v_j$). I když je $G$ souvislý, tak taková cesta nemusí vždy existovat; v takovém případě je $\delta(i, j, k) = \infty$. Je snadné si uvědomit, že $d_G(v_i, v_j) = \delta(i, j, n)$, čili vzdále-

nost mezi vrcholy v grafu $G$ je váha nejkratší cesty, která smí využít jeho všechny vrcholy.

Je zřejmé, že $\delta(i,j,k) \leq \delta(i,j,k-1)$, neboť máme jeden vrchol navíc, a přes ten může vést nějaká kratší cesta. Základní, a vlastně jedinou, myšlenkou Floydova-Warshallova algoritmu je pozorování, že když nastane situace, kdy $\delta(i,j,k) < \delta(i,j,k-1)$, pak ta kratší cesta musí využívat vrchol $v_k$. Ovšem, takovou cestu lze rozdělit na dvě – na cestu z $v_i$ do $v_k$ a cestu $v_k$ do $v_j$. Původní cesta $v_i \cdots v_k \cdots v_j$ využívala pouze vrcholy z $\{v_1,\ldots,v_k\}$, takže obě její části, $v_1 \cdots v_k$ i $v_k \cdots v_j$ využívají pouze vrcholy z $\{v_1,\ldots,v_{k-1}\}$. Zároveň to musejí být právě ty nejkratší cesty mezi $v_i$ a $v_k$ a $v_k$ a $v_j$, jinak by celková cesta $v_1 \cdots v_k \cdots v_j$ nebyla ta nejkratší. Mrkněte na obrázek 44.




Obrázek 44: Zde $\delta(i,j,3) = 7$, ale $\delta(i,j,4) = 6$.

Odtud plyne, že $\delta(i,j,k)$ je buď

- rovna $\delta(i,j,k-1)$, pokud přes vrchol $v_k$ nevede žádná kratší cesta z $v_i$ do $v_j$, nebo
- rovna součtu vah nejkratší cesty z $v_i$ do $v_k$ a nejkratší cesty z $v_k$ do $v_j$, pokud přes vrchol $v_k$ nějaká kratší cesta z $v_i$ do $v_j$ vede. V symbolech je tento součet roven $\delta(i,k,k-1) + \delta(k,j,k-1)$.

Předchozí body se dají shrnout do jednoho zápisu, neboť $\delta(i,j,k)$, jakožto váha **nejkratší** cesty, je vždy menším z obou množství. Konkrétně, vzoreček stojící za celým Floydovým-Warshallovým algoritmem je tento:

$$\delta(i,j,k) = \min(\delta(i,j,k-1), \delta(i,k,k-1) + \delta(k,j,k-1)). \qquad (\triangle)$$

Možná už vás napadá, jak samotný algoritmus bude fungovat. Přeci, čísla $\delta(i,j,k)$ mohu spočítat pro všechny dvojice $(i,j) \in \{1,\ldots,n\}^2$ v moment, kdy znám $\delta(i,j,k-1)$ opět pro všechny dvojice $(i,j)$. Jenže, $\delta(i,j,0)$ je váha hrany

mezi $v_i$ a $v_j$, pokud existuje, jinak $\infty$. Symbolicky,

$$\delta(i,j,0) = \begin{cases} w(v_iv_j), & \text{pokud } v_iv_j \in E, \\ \infty, & \text{jinak.} \end{cases}$$

To znamená, že znám hodnotu $\delta(i,j,0)$ pro všechny dvojice $(i,j)$, a tedy postupným zvyšováním $k$ umím spočítat i všechny hodnoty $\delta(i,j,k)$, a tím i nakonec, pro $k = n$, vzdálenosti mezi všemi páry vrcholů.

Nyní, když jsme myšlenku algoritmu doufám objasnili, zbývá jeho postup nějak rozumně zapsat. Naštěstí to lze velmi přímočaře, dokonce výrazně snadněji než například Kruskalův algoritmus.

Vlastně nám jde o to postupně nacházet kratší a kratší cesty mezi všemi dvojicemi vrcholů. Ten fakt, že hledáme vzdálenosti mezi vskutku **všemi** dvojicemi, nám umožňuje díky závěrům v předchozích odstavcích toto dělat efektivně.

Algoritmus vytváří zobrazení $D : V^2 \to [0,\infty]$, kterýmť si vlastně „pamatuje“ váhu zatím nejkratší nalezené cesty mezi každými dvěma vrcholy. Na začátku algoritmu je tudíž $D(v,v) := 0$ pro všechny $v \in V$ a $D(u,v) := w(uv)$ pro všechny dvojice $(u,v) \in V^2$, mezi kterými vede hrana. Pro všechny ostatní dvojice $(u,v)$ pokládáme $D(u,v) := \infty$.

Opírajíce sebe o předšedší úvahy víme, že budeme muset učinit v algoritmu $n$ kroků (to je počet vrcholů $G$) a v $k$-tém kroku vždy počítat hodnoty $\delta(i,j,k)$ pro všechny dvojice $(i,j)$. Přejeme si tudíž, aby hodnoty z předchozího kroku, tj. $\delta(i,j,k-1)$, byly před spuštěním $k$-tého kroku všechny již zakódovány v zobrazení $D$, jakožto čísla $D(v_i,v_j)$, představující **zatím** nejkratší známé cesty mezi všemi dvojicemi vrcholů. Rovnost ($\triangle$) se pročež překládá (**v rámci $k$-tého kroku algoritmu**) do rovnosti

$$D(v_i,v_j) = \min(D(v_i,v_j), D(v_i,v_k) + D(v_k,v_j)).$$

Po skončení $n$-tého kroku bude tedy zobrazení $D$ přesně odpovídat zobrazení $d_G$. Konečně můžeme přikročit k samotnému pseudokódu, jenž si prohlédněte níže.

**Cvičení 4.3.1.** Dokažte, že Floydův-Warshallův algoritmus selže, když připustíme i záporná ohodnocení hran.

**Algoritmus 2:** Floydův-Warshallův algoritmus.

**input :** souvislý ohodnocený graf $G = (V, E, w)$, kde $V = \{v_1, \ldots, v_n\}$
**output:** zobrazení $D : V^2 \to [0, \infty]$ takové, že $D \equiv d_G$

```
1  Inicializace;
2  for i ← 1 to n do
3      for j ← 1 to n do
4          if i = j then
5              Vzdálenost od vrcholu k němu samému je 0;
6              D(v_i, v_i) ← 0;
7          else
8              if v_i v_j ∈ E then
9                  Vzdálenost mezi různými vrcholy je váha hrany, pokud
                    existuje;
10                 D(v_i, v_j) ← w(v_i v_j);
11             else
12                 Jinak nastavíme hodnotu na ∞;
13                 D(v_i, v_j) ← ∞;

14 Postupně pro každé k od 1 do n budeme zmenšovat hodnoty D(v_i, v_j)
    pro všechny dvojice (i, j);
15 for k ← 1 to n do
16     for i ← 1 to n do
17         for j ← 1 to n do
18             D(v_i, v_j) ← min(D(v_i, v_j), D(v_i, v_k) + D(v_k, v_j));
19 return D;
```

## 4.4 Vzdálenost vrcholů

Sekci motivujeme úlohou „jako ze života“, která ve mně budí jistou míru nostalgie, jelikož jsem ji řešil na konci prvního ročníku v rámci zkoušky z programování. Dovolil jsem si ji ovšem literárně obohatit.

**Úloha** (Rodinný výlet)**.** Je prodloužený víkend a slezská rodina Koláčků plánuje cyklistický výlet oblastí Karviná. Jedou děda Koláček se svou chotí, babičkou Koláčkovou, a jejich čtyři uřvaná rozmazlená vnoučata – Matouš, Marek, Lukáš a Jan.

Z Třince do Orlové dánť jest směr a, i přes relativní nenáročnost terénu, vnoučata ustavičně fňukají, že chtějí jet tou nejkratší trasou. Děda Koláček, dobrodinec ten od kosti, snaží se vnoučatům vyhovět a nejkratší trasu úpěnlivě hledá. Do vřavy se přidává babička Koláčková, která ví, že trasa z Třince do Orlové vede přes mnoho malých vesnic, mnohože z nich hostí aspoň jednu hospodu.

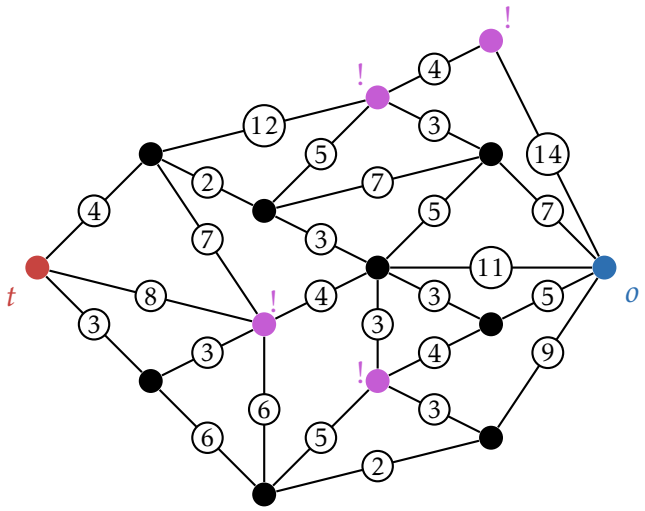
Vědouc velmi dobře, že každá hospoda zbrzdí cestu na nejméně půl hodiny a že i při mimoděčné zmínce o blízké hospodě si děda Koláček počíná mnouti pivní pupek, trvá babička Koláčková na tom, aby vybraná cesta z Třince do Orlové vedla přes vesnice, které jsou tak daleko od hospod, jak je to jen možné. Křik vnoučat brzy ji však přesvědčí, že délka trasy převažuje nad množstvím hospod, ke kterým se po cestě přiblíží.

Koláčkovic rodina tudíž stojí před nelehkým úkolem vybrat ze všech nejkratších cest z Třince do Orlové tu, která je hospodám co nejvíce vzdálena.

Brzy znaven, děda Koláček posílá nezkrotná vnoučata s konečně neosedlanými bicykly domů a výlet do Orlové se odkládá na příští školní prázdniny. Pomozme mu jej zorganizovat předem.

Úlohu si modelujeme ohodnoceným grafem $G = (V, E, w)$. Vrcholy představují vesnice, případně města, mezi Třincem a Orlovou, z nichž některé jsou označeny vykřičníkem mínícím přítomnost jedné či více hospod.

Nejprve se soustředíme na nejpodstatnější část úlohy, tou jest nalezení nejkratší cest mezi Třincem a Orlovou, kterážto města si v zájmu strohosti




Obrázek 45: Graf obcí mezi Třincem a Orlovou.

označíme písmeny $t, o \in V$.

První, co by člověka mohlo napadnout je spustit Floydův-Warshallův algoritmus. Zde je však nutno si uvědomit, že ten není možné modifikovat tak, aby hledal vzdálenost mezi dvěma konkrétními vrcholy. Totiž, z popisu jeho průběhu plyne, že váhu nejkratší cesty mezi dvěma vrcholy, která vede pouze přes vrcholy z množiny $\{v_1, \ldots, v_k\}$, umím spočítat teprve tehdy, když znám váhy nejkratších cest používajících pouze vrcholy $v_1, \ldots, v_{k-1}$ mezi **všemi dvojicemi vrcholů** v grafu $G = (\{v_1, \ldots, v_n\}, E, w)$. Odtud plyne, že Floydův-Warshallův algoritmus potřebuje znát zatím nejkratší nalezenou cestu mezi každými dvěma vrcholy, aspoň dokud $k$ není $n$.

Samozřejmě je též možné najít **všechny** cesty mezi danými dvěma vrcholy a z nich vybrat tu nejkratší. Tento postup je však ještě výrazně méně efektivní než Floydův-Warshallův algoritmus.

Není však radno zoufat, bo algoritmus pro relativně efektivní nalezení vzdálenosti mezi určenými dvěma vrcholy v grafu existuje. Nosí jméno svého tvůrce, holandského informatika a programátora Edsgera W. Dijkstry (čteno „dajkstry“). Není složitý, ale zcela jistě je méně přímočarý než Kruskalův i Floydův-Warshallův algoritmus.

Pozastavme se na moment a oceňme tu ironii, že najít efektivní algoritmus na výpočet vzdálenosti mezi dvěma vrcholy je výrazně obtížnější než na

výpočet vzdálenosti mezi všemi dvojicemi vrcholů.

### 4.4.1 Dijkstrův algoritmus

Dijkstrův algoritmus má mnoho společného s Floydovým-Warshallovým algoritmem, například rovněž v každém kroku zlepšuje zatím nejlepší známou vzdálenost od *zdrojového* vrcholu k ostatním. Jeho základní myšlenka je však jiná. Dijkstrův algoritmus využívá tzv. „průchod do šířky“, což znamená, že se „šíří“ od jednoho zvoleného *zdrojového vrcholu* nejprve do jeho sousedů (tj. do vrcholů s ním spojených hranou). Při průchodu si u každého vrcholu pamatuje zatím nejlepší nalezenou cestu ze zdrojového vrcholu do něj, a kdykoli vstoupí do dalšího vrcholu, ověří, zda cesta, po které se do něj dostal, není kratší než dosavad nejlepší.

Tento postup sám není nijak revoluční a průchod do šířky je asi tak starý jako grafy samotné. Myšlenka, se kterou Edsger Dijkstra přišel, však na svou dobu revoluční byla, neboť se jednalo o tehdy nejefektivnější způsob nalezení vzdálenosti mezi vrcholy v ohodnoceném grafu. U každého vrcholu si pamatoval váhu zatím nejkratší nalezené cesty od zdroje k němu. Uvědomil si, že **pokud vybírá vrcholy vždy od těch s minimální uloženou hodnotou**, tak v moment, **kdy projde všechny sousedy nějakého vrcholu**, pak už **je zapamatovaná hodnota v tomto vrcholu nejlepší možná**, tedy se jedná o skutečnou vzdálenost od zdroje k tomuto vrcholu a už se do něj nikdy nemusí znovu vracet.

Důvod, proč tomu tak je, není na první pohled zřejmý a objasníme ho podrobně v důkazu správnosti Dijkstrova algoritmu.

Ještě před slovním popisem a představením pseudokódu je dobré se zamyslet, o kolik více efektivní je Dijkstrův způsob oproti prostému průchodu do šířky. Pro úplnost:

- Průchod do šířky vždy putuje od vrcholu ke všem jeho sousedům, od těch zase k jejich sousedům a tak dále. Tedy se mnohokrát může (a **musí**, aby našel opravdu nejkratší cestu) vracet do vrcholů, které již prošel, v moment, kdy do nich přes nějaké vrcholy dříve navštívené vede cesta.
- Dijkstrův algoritmus nahlédne z právě vybraného vrcholu do jeho sousedů, ale pak nepokračuje nutně jimi. Naopak, právě vybraný vr-

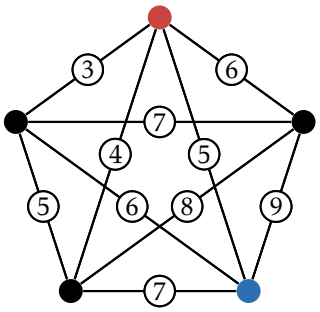
chol již „uzamkne“ a pokračuje nějakým s minimální uloženou zatím nejkratší cestou ze zdroje do něj.

Kvůli tomu, že přes sousedy každého vrcholu může zpět do něj vést nějakou oklikou cesta zpátky, projdu při běžném průchodu do šířky úplně všechny cesty v grafu $G$.

Naopak, Dijkstrův algoritmus *zakazuje* vracet se do vrcholů, jejichž všichni sousedi již byli navštíveni. Tedy, každý vrchol je navštíven v nejhorším tolikrát, kolik má sousedů. Je snadné si rozmyslet, o kolik je (řádově) v průměrném případě počet sousedů vrcholu nižší než počet cest jím procházejících.

Největší časový rozdíl mezi obvyklým průchodem do šířky a Dijkstrovým algoritmem je vidět v tzv. *úplných grafech*, tedy grafech, ve kterých je každý vrchol spojen s každým. V úplném grafu na $n$ vrcholech, který se často značí $K_n$, definuje libovolná podmnožina vrcholů cestu. Již dlouho víme, že všech podmnožin množiny o $n$ prvcích je $2^n$. Tedy, algoritmus průchodu do šířky udělá při hledání vzdálenosti mezi dvěma vrcholy v úplném grafu vždy $2^n$ kroků. Oproti tomu, Dijkstrův algoritmus udělá v grafu na $n$ vrcholech vždy nejvýše $n(n-1)$ kroků, protože každý vrchol má nejvýše $n$ sousedů. Už jen pro graf na 10 vrcholech je číslo $10 \cdot 9$ přibližně 900krát menší než číslo $2^{10}$.

Na obrázku 46 vidíte úplný graf na 5 vrcholech. Upřímně doporučuji, abyste si tvrzení předchozího odstavce vyzkoušeli v praxi.




Obrázek 46: Ohodnocený úplný graf na pěti vrcholech s vyznačeným zdrojovým a cílovým vrcholem.

Nyní přistoupíme ke slovnímu popisu Dijkstrova algoritmu. Algoritmus striktně vzato nepočítá vzdálenost mezi dvěma vrcholy, anobrž vzdálenost od *zdrojového vrcholu* ke všem ostatním.

Podobně jako Floydův-Warshallův algoritmus, i Dijkstrův algoritmus vytváří postupně zobrazení $d : V \to [0, \infty]$ takové, že po jeho skončení platí $d(v) = d_G(s, v)$ pro všechna $v \in V$ a nějaký pevně zvolený zdrojový vrchol $s \in V$. Na začátku algoritmu je pročež $d(s) = 0$ a $d(v) = \infty$ pro všechny $s \neq v \in V$. V každém kroku algoritmu navíc existuje množina $X$ „zakázaných" vrcholů, do kterých už není možné se dívat ani vracet. V moment, kdy procházím sousedy právě vybraného vrcholu, ignoruji ty, které již byly někdy zakázány.

Jeden krok algoritmu vypadá takto:

(1) Označ jako zvolený vrchol libovolný $v \in V$ **s minimální** $d(v)$.

(2) Pro každého souseda $u$ zvoleného vrcholu $v$ porovnej zatím nejkratší známou cestu z $s$ do $u$, to jest $d(u)$, s váhou zatím nejkratší cesty z $s$ do $u$ vedoucí přes $v$, to jest $d(v) + w(uv)$. Je-li druhá hodnota menší, změň $d(u)$ na $d(v) + w(uv)$.

(3) Vrchol $v$ označ jako zakázaný.

(4) Pokud ještě existuje vrchol, který není zakázaný, opakuj (1).

Jak jsme již psali – fakt, že jako další vrchol v pořadí volíme ten s minimální známou nejkratší cestou z $s$, umožňuje se nikdy do vrcholů, jejichž sousedy projdeme, nevracet, není samozřejmý. Výklad o obecném chodu Dijkstrova algoritmu zakončíme prezentací jeho pseudokódu pro **souvislý** ohodnocený graf $G = (V, E, w)$ s pevně zvoleným zdrojovým vrcholem $s \in V$. Ihned poté dokážeme jeho správnost.

> **Poznámka.** Stejně jako Floydův-Warshallův algoritmus, ani Dijkstrův algoritmus neselže pro nesouvislé grafy, ale všem vrcholům $v \in V$ nedosažitelným z $s$ zůstane hodnota $d(v) = \infty$, tedy je zcela zbytné nesouvislé grafy uvažovat.

Pro stručnost vyjádření zavedeme ještě značení

$$n(v) := \{u \in V \mid uv \in E\},$$

čili označíme výrazem $n(v)$ (od angl. ***n**eighbour*), množinu všech sousedů vrcholu $v \in V$.

**Algoritmus 3:** Dijkstrův algoritmus

**input :** souvislý ohodnocený graf $G = (V, E, w)$ s počátečním vrcholem $s \in V$
**output:** zobrazení $d : V \to [0, \infty]$ takové, že $d \equiv d_G(s, -)$

```
1  Inicializace;
2  d(s) ← 0;
3  for v ∈ V \ {s} do
4      d(v) ← ∞;
5  X ← ∅;
6  Pokračuj, dokud existují nezakázané vrcholy;
7  while X ≠ V do
8      Najdi vrchol s minimální zatím nejlepší známou vzdáleností od s;
9      v ← libovolný vrchol s minimální d(v) takový, že v ∉ X;
10     Projdi všechny jeho sousedy, kteří nejsou zakázáni;
11     for u ∈ n(v) \ X do
12         Pokud do u vede přes v kratší cesta, změň d(u) na její váhu;
13         if d(u) > d(v) + w(uv) then
14             d(u) ← d(v) + w(uv);
15     Po průchodu všech sousedů, zakaž v;
16     X ← X ∪ {v};
17 return d;
```

**Poznámka.** Pokud bychom chtěli použít Dijkstrův algoritmus na nalezení vzdálenosti mezi dvěma konkrétními vrcholy $s, t \in V$, pak stačí `while` cyklus v pseudokódu výše zastavit v moment, kdy $t \in X$. Pak už je totiž známa hodnota $d_G(s,t)$, kterou jsme chtěli spočítat.

Povšimněte si, že v tomto případě ještě k naprosté efektivitě Dijkstrově algoritmu krůček schází. Totiž, cestou od $s$ do $t$ projde velmi mnoho vrcholů blízkých $s$, přes které „víme“, že nejkratší cesta z $s$ do $t$ vést nemůže. Algoritmy používané například v GPS aplikacích staví částečně na principu Dijkstrova algoritmu, ale tento nedostatek většinově odstraňují. Jejich popis je však zároveň nad rámec tohoto textu.

**Tvrzení 4.4.1.** Dijkstrův algoritmus je korektní.

**Důkaz.** Musíme ověřit, že Dijkstrův algoritmus je konečný a počítá správně.

Konečnost je zřejmá. Vrchol $v \in V$ s minimální $d(v)$ vždy existuje (i kdyby ta hodnota měla být $\infty$), a tedy nastane chvíle, kdy v $X$ jsou všechny vrcholy grafu $G$. V ten moment $X = V$, cyklus končí a algoritmus vrací zobrazení $d$.

Korektnost ukážeme indukcí podle počtu zakázaných vrcholů, tedy podle $\#X$. Naším indukčním předpokladem bude, jako tomu obvykle v důkazech indukcí bývá, že algoritmus funguje. To přesně znamená, že v každém kroku

(a) je $d(v) = d_G(s,v)$ pro všechny vrcholy $v \in X$;

(b) je $d(u)$ váha nejkratší cesty z $s$ do $u$ vedoucí pouze přes vrcholy v $X$, pro všechna $u \in V \setminus X$.

Předpoklad (a) říká v lidské mluvě, že zakazování vrcholů si opravdu můžeme dovolit, tedy že do zakázaných vrcholů již nejde nalézt kratší cestu.

Význam předpokladu (b) je těžší dekódovat. Představme si spíše, co by se stalo, kdyby **neplatil**. Pak by existovala nějaká kratší cesta do nezakázaného vrcholu $u$ přes zakázané vrcholy. Ovšem, přesně pro to, že tyto vrcholy jsou již zakázané, nemohu během zbytku průběhu

algoritmu již nikdy tuto cestu objevit. Pokud by se pro nějaký $u \in V$ stalo, že zrovna tato cesta je tou nejkratší, pak již zároveň nemohu nalézt hodnotu $d_G(s,u)$ a algoritmus selže.

Počáteční případ $\#X = 1$ je snadný. V tomto stádiu jsme zrovna zakázali zdrojový vrchol $s$ a pro každého jeho souseda $u$ jsme položili $d(u) = w(su)$, jelikož $d(s) = 0$ a $d(u)$ bývala hodnota $\infty$. Tedy, zakázaný vrchol je jedině $s$ a $d(s) = d_G(s,s) = 0$, čímž jsme ověřili platnost (a). Jediné vrcholy, do nichž vede cesta přes zakázané vrcholy, tedy přes vrchol $s$, jsou sousedé $s$, a váha nejkratší cesty je zřejmě $w(su)$ pro každého souseda $u \in n(s)$. To dokazuje (b).

Nyní předpokládejme, že (a) i (b) platí pro $X$ a zakažme nějaký zatím nezakázaný vrchol $u \in V \setminus X$. Dokážeme nejprve platnost (a). Z indukčního předpokladu platí (a) pro všechny vrcholy $X$ kromě nově přidaného $u$, tedy stačí ukázat, že (a) platí i pro $u$.

Dokazujme tedy, že $d(u) = d_G(s,u)$. Budeme postupovat sporem. Ať tedy tato rovnost neplatí. Pak existuje nenalezená nejkratší cesta $\mathcal{P}$ z $s$ do $u$ a platí $w(\mathcal{P}) < d(u)$. Rozlišíme dva případy.

(1) $\mathcal{P}$ prochází nějakými nezakázanými vrcholy.

(2) $\mathcal{P}$ neprochází žádným nezakázaným vrcholem (tedy všechny její vrcholy jsou v $X$).

Ať nastala možnost (1). Označme písmenem $w$ **první nezakázaný** vrchol na cestě $\mathcal{P}$ (směrem od $s$). Z indukčního předpokladu (b) jsou $d(w)$ a $d(u)$ váhy nejkratší cest z $s$ do $w$, resp. z $s$ do $u$, vedoucích jen přes vrcholy z $X$. To ale znamená, že $d(u)$ je aspoň tolik, co $d(w)$ + váha nejkratší cesty z $w$ do $u$. Protože váha nejkratší cesty z $w$ do $u$ je jistě kladné číslo, plyne odtud, že $d(u) > d(w)$. Ovšem, $w$ není zakázaný a algoritmus prochází nezakázané vrcholy postupně od těch s minimální váhou zatím nejkratší známé cesty z $s$ do nich. Algoritmus navštívil $u$ dříve než $w$, pročež $d(u) \leq d(w)$. Pak ale máme

$$d(w) < d(u) \leq d(w),$$

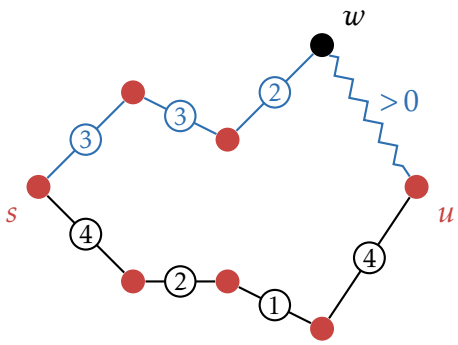
což je zřejmý spor. Vizte obrázek 47a.

Nyní ať nastala možnost (2). Ať $w$ je **předposlední** vrchol na cestě $\mathcal{P}$

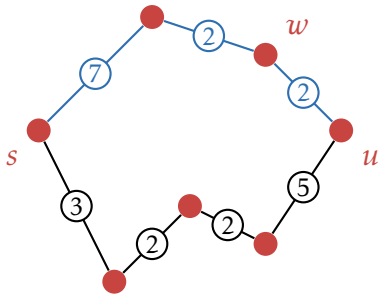
(tedy ten těsně před $u$). Protože $w$ je zakázaný, platí $d(w) = d_G(s, w)$ z indukčního předpokladu (a). Pak ale $w(\mathcal{P}) = d(w) + w(wu) < d(u)$. To je ovšem spor, protože $u$ je sousedem $w$ a $w$ byl zakázán dříve než $u$, a tedy algoritmus měl při procházení sousedů $w$ nastavit hodnotu $d(u)$ na $d(w) + w(wu)$. Vizte obrázek 47b.

Zbývá ověřit platnost (b) pro $X \cup \{u\}$. Z indukčního předpokladu platí (b) pro všechny vrcholy z $V \setminus X$ kromě sousedů $u$, protože pro ty jsme přidáním $u$ žádnou novou cestu přes zakázané vrcholy nenašli. Pro sousedy $u$ však platí (b) zřejmě také, protože pokud do nich existuje přes $u$ kratší cesta, tak u nich algoritmus váhu zatím nejkratší známé cesty z $s$ změnil na váhu této kratší cesty přes $u$.

Tím je důkaz správnosti Dijkstrova algoritmu indukcí podle $\#X$ dokončen. □




(a) Případ (1). Cesta $\mathcal{P}$ je značena modře. Množina $X$ červeně. Nemůže nastat, pokud algoritmus běží správně.




(b) Případ (2). Cesta $\mathcal{P}$ je značena modře. Množina $X$ červeně. Nemůže nastat, pokud algoritmus běží správně.

Obrázek 47: Obecný indukční krok v důkazu správnosti Dijkstrova algoritmu.

### 4.4.2 Hurá na výlet

Sekci o vzdálenosti mezi vrcholy zakončíme popisem algoritmu řešícím úlohu o rodinném výletu. V jeho jádru je nalezení nejkratší cesty mezi zvolenými vrcholy, čili Dijkstrův algoritmus zde zcela jistě najde své využití. Co ovšem s hospodami? Zadání úlohy stanovuje, že nám nestačí nalézt pouze nejkratší cestu z Třince do Orlové, nýbrž z množiny všech nejkrat-

ších cest vybrat tu, jež je tak daleko od hospod, jak to jen lze – v tom smyslu, že minimum vzdáleností od vrcholů cesty k hospodám je největší možné.

Poslední věta předchozího odstavce napovídá, že vzdálenost k hospodě máme vnímat jako *vlastnost konkrétního vrcholu*. Zcela jistě není vhodné obě části úlohy (tedy nalezení nejkratší cesty a té, která se nejvíce vyhýbá hospodám) oddělovat. Uvažme totiž případ, kdy **všechny** cesty mezi Třincem a Orlovou jsou stejně dlouhé. Potom by postup

(1) Najdi všechny nejkratší cesty.

(2) Vyber z nich tu nejdál od hospod.

vyžadoval prozkoumání všech možných cest, čemuž se z již párkrát diskutovaných důvodů chceme vyhnout.

Řádově rychlejší postup bude tedy cestu nejdál od hospod hledat současně s tou nejkratší. Abychom toto mohli udělat, musíme znát vzdálenost každého vrcholu od nejbližší hospody. To lze snadno zařídit spuštěním Dijkstrova algoritmu vícekrát s každou hospodou jako zdrojovým vrcholem.

Formálně, ať $H = \{h_1, \ldots, h_k\} \subseteq V$ značí množinu hospod. Pro každé $i \leq k$ máme z výstupu Dijkstrova algoritmu zobrazení $\theta_i : V \to [0, \infty]$ takové, že $\theta_i(v) = d_G(h_i, v)$ pro každý $v \in V$. Nyní již snadno využijeme zobrazení $\theta_i$ k definici zobrazení $\Theta : V \to [0, \infty]$, které značí pro každý vrchol jeho vzdálenost k nejbližší hospodě. Uvědomme si, že stačí definovat

$$\Theta(v) := \min_{i \leq k} \theta_i(v) \quad \forall v \in V.$$

Druhá část řešení bude též spočívat ve spuštění Dijkstrova algoritmu, avšak s mírnými úpravami. Zamysleme, jak bychom mohli během jednoho průchodu vrcholem porovnat váhu a vzdálenost od hospody v něm uložených údajů o cestě s těmi o té, po které jsme do něj právě přišli. To ve skutečnosti není vůbec složité. Přeci, primárně nám jde o to najít nejkratší cestu. Vzdálenost k hospodám řešíme až v moment, kdy by dvě různé cesty měly být stejně dlouhé. Stačí pročež vždy nejprve porovnat váhy cest do vrcholu vedoucích, a teprve když se rovnají, porovnat též jejich vzdálenosti k hospodám.

Navíc k zobrazení $d$ a množině $X$ (z popisu Dijkstrova algoritmu) si u každého vrcholu musíme pamatovat cestu, po které jsme do něj přišli. A k

těmto cestám též jejich vzdálenosti od hospod. Pro tento účel si zavedeme zobrazení $P : V \to \bigcup_{i=0}^{n} V^i$, čili zobrazení, které každému vrcholu přiřadí $m$-tici (pro nějaké $m \leq n$) vrcholů představující cestu, kterou jsem se do něj dostal. Na začátku máme $P(v) = \emptyset$ pro všechny vrcholy $v \in V$. Je-li $\mathcal{P} = v_1 \cdots v_k$, pak výrazem $\mathcal{P} \oplus u$ myslíme cestu $v_1 \cdots v_k u$; pochopitelně pouze za předpokladu, že $v_k u \in E$. Konečně, výrazem $h(\mathcal{P})$ budeme chápat vzdálenost cesty $\mathcal{P}$ k hospodám, tj. minimum přes všechny vrcholy z jejich vzdáleností k nejbližší hospodě. Formálně

$$h(\mathcal{P}) := \min_{v \in \mathcal{P}} \Theta(v).$$

Zobrazení $h$ budeme též v průběhu algoritmu vytvářet, neboť v opačném případě bychom museli při vchodu do nového vrcholu pokaždé procházet všechny vrcholy obou porovnávaných cest. Je-li totiž $\mathcal{P} = v_1 \cdots v_k$ a $v_k u \in E$, pak $h(\mathcal{P} \oplus u)$ je buď $h(\mathcal{P})$, pokud je $u$ od hospod dál než aspoň jeden vrchol v $\mathcal{P}$. V opačném případě je $h(\mathcal{P} \oplus u) = \Theta(u)$. Souhrnně, vždy máme

$$h(\mathcal{P} \oplus u) = \min(h(\mathcal{P}), \Theta(u)).$$

Přicházíme ke slovnímu popisu řešení úlohy o rodinném výletu. Počáteční vrchol představující město Třinec značíme $t$ a koncový vrchol představující město Orlová značíme $o$.

(1) Polož $d(t) := 0$ a $d(v) := \infty$ pro všechny ostatní vrcholy $v \neq t$. Dále polož $X := \emptyset$, $P(t) := t$ a $P(v) = \emptyset$ pro všechny ostatní vrcholy $v \neq t$. Konečně, polož $h(P(t)) := \Theta(t)$.

(2) Vyber libovolný vrchol $v \in V$ s minimální $d(v)$.

(3) Projdi všechny sousedy $u$ vrcholu $v$, které nejsou v $X$.

   (a) Je-li $d(u) > d(v) + w(uv)$ nebo

   (b) $d(u) = d(v) + w(uv)$ a zároveň $\min(h(P(u)), \Theta(v)) > h(P(v))$, pak polož

   - $d(u) := d(v) + w(uv)$,
   - $P(u) := P(v) \oplus u$,
   - $h(P(u)) := \min(h(P(v)), \Theta(u))$.

   (c) V ostatních případech nedělej nic.

(4) Zakaž vrchol $v$, tj. přidej ho do $X$.

(5) Pokud $o \notin X$, opakuj bod (2).

**Poznámka.** Bod (3) je jádrem popsaného algoritmu. Lidsky řečeno znamenají jeho body (a) a (b) následující.

(a) Do $u$ vede přes $v$ kratší cesta, než ta zatím známá.

(b) Do $u$ vede přes $v$ stejně dlouhá cesta, jako ta zatím známá, ale navíc je vzdálenost této nové cesty k hospodě větší než vzdálenost k hospodě té zatím známé.

Toto přesně znamená, že primárně nás zajímá váha cesty, sekundárně pak její vzdálenost k hospodě. V obou případech (a) i (b) pak vlastně jenom přepíšeme dosavadní cestu do $u$. To jest, zapamatujeme si ji samotnou ($P(v) \oplus u$), její váhu ($d(v) + w(uv)$) a, konečně, její vzdálenost k hospodě; ta zůstane přirozeně stejná, pokud je nějaký vrchol $P(v)$ blíž k hospodě než $u$, v opačném případě se změní na $\Theta(u)$, čili na vzdálenost $u$ k nejbližší hospodě.

Celou sekci završíme pseudokódem řešícím trable dědy Koláčka.

**Algoritmus 4:** Řešení úlohy o rodinném výletu

**input :** Souvislý ohodnocený graf $G = (V, E, w)$, zdrojový vrchol $t$, koncový vrchol $o$, podmnožina $H \subseteq V$ a zobrazení $\Theta : V \to [0, \infty]$ takové, že $\Theta(v) = \min_{u \in H} d_G(v, u)\ \forall v \in V$.

**output:** Nejkratší cesta $P(o)$ mezi $t$ a $o$, její váha $d(o)$ a její vzdálenost k nejbližší hospodě $h(P(o))$.

*Inicializace*;
$d(t) \leftarrow 0$;
$P(t) \leftarrow t$;
$h(P(t)) \leftarrow \Theta(t)$;
**for** $v \in V \setminus \{t\}$ **do**
    $d(v) \leftarrow \infty$;
    $P(v) \leftarrow \emptyset$;
    $h(P(v)) \leftarrow 0$;
$X \leftarrow \emptyset$;
*Pokračuj, dokud vrchol $o$ není zakázaný. Tedy do něj ještě neznáme nejkratší cestu*;
**while** $o \notin X$ **do**
    *Vezmi nezakázaný vrchol s minimální známou vzdáleností*;
    $v \leftarrow$ vrchol takový, že $d(v) = \min_{u \in V \setminus X} d(u)$ a $v \notin X$;
    *Projdi jeho nezakázané sousedy*;
    **for** $u \in n(v) \setminus X$ **do**
        *Když současná cesta zlepšuje známou vzdálenost od $t$ nebo ji zachovává a zlepšuje vzdálenost od hospod, aktualizuj ji*;
        **if** $(d(u) > d(v) + w(uv))$ **or**
        $(d(u) = d(v) + w(uv)$ **and** $\min(h(P(v)), \Theta(u)) > P(u))$ **then**
            $d(u) \leftarrow d(v) + w(uv)$;
            $P(u) \leftarrow P(v) \oplus u$;
            $h(P(u)) \leftarrow \min(h(P(v)), \Theta(u))$;
        *Zakaž vrchol $v$*;
        $X \leftarrow X \cup \{v\}$;
    **return** $P(o), d(o), h(P(o))$;

**Cvičení 4.4.1.** Vyřešte úlohu o rodinném výletu pro graf daný obrázkem 45.

# Seznam cvičení

### Úvodní pojmy

(1) Dokažte Tvrzení 2.4.3.

(2) Dokažte, že relace (ne nutně ekvivalence!) $R$ je transitivní, právě tehdy když $R \circ R \subseteq R$.

(3) Vyřešte následující úlohy rozprostřené po sekci 2.5. Konkrétně,
- dokažte, že mezi dvěma konečnými množinami různé velikosti neexistuje žádná bijekce.
- pro množinu $A$ velikosti $n$ určete počet různých bijektivních zobrazení $f : [n] \cong A$.
- určete, jakou podmínku splňují zobrazení $f : A \to B$, ke kterým existuje zobrazení inverzní.

(4) Dokažte, že každá lineární funkce $f : \mathbb{R} \to \mathbb{R}$, tedy funkce daná předpisem
$$f(x) = ax + b \quad \text{pro } a, b \in \mathbb{R}, a \neq 0$$
je bijektivní zobrazení.

(5) Nechť $A$ je konečná množina. Zformulujte důkaz, že libovolné zobrazení $f : A \to A$, které je definované pro každé $x \in A$, je **prosté, právě tehdy když je na**.

(6) Najděte příklad zobrazení $f : \mathbb{N} \to \mathbb{N}$ definovaného na celém $\mathbb{N}$, které je
   (a) prosté, ale není na.
   (b) na, ale není prosté.

(7) Udělejte cvičení rozmístěná po sekci 2.6. Konkrétně,
   (a) dokažte, že Hasseho diagram každého lineárního uspořádání má stejný tvar jako diagram na obrázku 13.
   (b) dokažte, že relace dělitelnosti $|$ je uspořádání na každé podmnožině přirozených čísel.

(8) Explicitně popište všechny relace (na libovolné množině), které jsou zároveň ekvivalencí a (částečným) uspořádáním.

(9) Řekněme, že $R$ a $S$ jsou uspořádání na množině $A$. Které z následujících relací jsou také uspořádáními na $A$?

- $R \cap S$
- $R \cup S$
- $R \setminus S$
- $R \circ S$

(10) Dokažte indukcí, že

$$\sum_{i=1}^{n} i2^i = (n-1)2^{n+1} + 2.$$

(11) Tak zvaná *Fibonacciho* posloupnost je definována tak, že další člen dostanu jako součet dvou předchozích. Formálně, $F_0 = 0, F_1 = 1$ a $F_n = F_{n-1} + F_{n-2}$, kde $n \in \mathbb{N}$. Dokažte indukcí, že

$$F_n \leq \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}$$

pro všechna $n \geq 0$.

(12) Nakresleme $n$ přímek v rovině, a to tak, že

- žádné 2 nejsou rovnoběžné a
- žádné 3 se neprotínají v jednom bodě.

Dokažte indukcí, že takhle nakreslené přímky rozdělují rovinu na $n(n+1)/2 + 1$ částí.

### Počítání

(1) Dokažte Tvrzení 3.1.2 indukcí podle velikosti množiny $A$.

(2) Určete počet všech uspořádaných dvojic $(A, B)$, kde $A \subseteq B \subseteq \{1, \ldots, n\}$.

(3) Spočtěte složení $\sigma\tau$ a $\tau\sigma$, když

(a) $\sigma = (143)(26), \tau = (146)(253)$,

(b) $\sigma = (14)(25)(36), \tau = (123456)$,

(c) $\sigma = (145)(263), \tau = (154)(236)$.

(4) Určete řád permutace $\sigma$, kde

(a) $\sigma = (1345)$,

(b) $\sigma = (1346)(28)(579)$.

(5) (těžké) Určete číslo $C_n(S_X)$, kde $\#X = 100$ a $n \leq 50$, tedy počet všech permutací na 100 číslech s aspoň jedním cyklem délky menší nebo rovné 50.

Samozřejmě jich je $100! - C_{\geq 51}(S_X)$, ale cílem úlohy je spočítat je nějak chytře, aby člověk dostal hezčí vzoreček.

(6) Ať $\sigma \in S_n$, tedy $\sigma$ je permutace množiny $\{1,\dots,n\}$. Řekneme, že $\sigma$ *invertuje* dvojici $(i,j)$, kde $i,j \in \{1,\dots,n\}$, když $i < j$, ale $\sigma(i) > \sigma(j)$.

Definujme

$$I(\sigma) := \{(i,j) \in \{1,\dots,n\}^2 \mid \sigma \text{ invertuje } (i,j)\},$$

čili $I(\sigma)$ je množina všech dvojic $(i,j)$, které $\sigma$ invertuje. Uvědomme si, že $I(\sigma)$ je podmnožinou $\{1,\dots,n\}^2$, čili relací na $\{1,\dots,n\}$.

(a) Dokažte, že $I(\sigma)$ je transitivní relace na $\{1,\dots,n\}$ pro každou permutaci $\sigma \in S_n$.

(b) (těžké) Navrhněte algoritmus, který pro danou permutaci $\sigma \in S_n$ spočte $\#I(\sigma)$.

(c) Spočtěte počet invertovaných dvojic, čili $\#I(\sigma)$, permutací $\sigma = (134)(579)(26)$.

(7) Dokažte, že

$$\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n}.$$

**Hint**: použijte lemma 3.3.6.

(8) Dokažte vzorec

$$\sum_{k=r}^{n} \binom{k}{r} = \binom{n+1}{r+1}$$

pro pevné $r \in \mathbb{N}$ indukcí podle $n \in \mathbb{N}$.

(9) (těžké) Kolik existuje podmnožin $\{1,\dots,n\}$, které neobsahují žádná dvě po sobě jdoucí čísla. Formálně, určete velikost množiny

$$\{A \subseteq \{1,\dots,n\} \mid \{i,j\} \not\subseteq A, \text{ kdykoli } |i-j| = 1\}.$$

(10) Ať $p$ je prvočíslo a $k, n$ přirozená čísla.

(a) Dokažte, že pro $k < p$ je $\binom{p}{k}$ dělitelné $p$.

(b) Dokažte, že $\binom{n}{p}$ je dělitelné $p$ právě tehdy, když $\lfloor n/p \rfloor$ je dělitelné $p$, kde $\lfloor \cdot \rfloor$ značí *dolní celou část*.

(11) Budeme vybírat $k$-tice předmětů z $n$ druhů předmětů. Budeme uvažovat různé typy výběru podle toho, jestli vybíráme $k$-tice uspořádané, nebo neuspořádané (tj. podmnožiny) a též podle toho, zda každého druhu je vždy jen jeden předmět, či nikoli. Doplňte následující tabulku:

| | Jen 1 předmět každého druhu | Libovolně mnoho předmětů každého druhu |
|---|---|---|
| Uspořádané $k$-tice | | |
| Neuspořádané $k$-tice | | |

Tabulka 2: Výběr $k$-tic předmětů z $n$ druhů předmětů.

(12) Kolika způsoby můžeme postavit 7 čarodějnic a 5 vodníků do řady tak, aby 2 vodníci nikdy nestáli vedle sebe?

(13) Jeden z oblíbených a celkem rychlých rozkladů čísla na prvočísla je tzv. *Eratosthenovo síto*.

Funguje na principu vyškrtávání násobků čísel. Konkrétně, algoritmus prochází seznam čísel až do nějaké hranice, a kdykoli narazí na ještě neodškrtnuté číslo, odškrtne z tohoto seznamu všechny jeho násobky kromě něho samotného. Sami si rozmyslete, že tímto způsobem zůstanou ve výsledném seznamu pouze prvočísla.

My si tady rozmyslíme pouze zjednodušenou verzi. Spočtěte, kolik zůstane čísel mezi 1 a 1000 potom, co vyškrtáme všechny násobky čísel $2, 3, 5$ a $7$.

(14) Určete počet přirozených čísel menších než 100, které nedělí druhá mocnina žádného přirozeného čísla (kromě 1).

(15) Kolika způsoby lze seřadit do řady 5 Čechů, 4 Maďary a 3 Rusy, aby všichni příslušníci jednoho národa nikdy nestáli hned za sebou?

### Teorie grafů

(1) Nakreslete graf $G = (V, E)$, kde
- $V = \{1, \dots, 5\}, E = \{\{1,2\},\{1,3\},\{1,5\},\{2,3\},\{3,4\},\{4,5\}\}$.
- $V = \{1, \dots, 5\}$, $E = \binom{V}{2}$.
- $V = \{1, \dots, 8\}$, $E = \{e_1, \dots, e_8\}$ a
  - $t(e_i) = s(e_{i+1}) = i + 1$ pro všechna $i \leq 7$,
  - $t(e_8) = s(e_1) = 1$.

(2) Popište všechny grafy $G = (V, E)$, kde $E$ je relace na $V$, která je antireflexivní, symetrická (to je součástí definice grafu) a navíc **transitivní**.

(3) Ať $V$ je konečná množina a $E$ je relace na $V$, která je antireflexivní a symetrická. Definujme navíc na $E$ další relaci $\sim$ předpisem

$$(v, v') \sim (w, w') \Leftrightarrow (v, v') = (w, w') \vee (v, v') = (w', w).$$

Dokažte, že pak existuje bijekce mezi $[E]_\sim$ a množinou

$$E' := \{\{v, v'\} \mid (v, v') \in E\},$$

čili mezi množinou tříd ekvivalence $E$ podle $\sim$ a množinou, kterou dostanu tak, že z uspořádaných dvojic v $E$ udělám neuspořádané dvojice, tj. dvouprvkové podmnožiny. Pro intuici vizte poznámku pod definicí 4.0.1.

(4) Spočtěte, kolik existuje grafů na $n$ vrcholech.

(5) Ať $\mathcal{P}_1 = e_1^1 e_2^1 \cdots e_n^1$ a $\mathcal{P}_2 = e_1^2 e_2^2 \cdots e_n^2$ jsou cesty. Za předpokladu, že $t(e_n^1) = s(e_1^2)$, definujeme jejich *sloučení*, které zapíšeme třeba jako $\mathcal{P}_1 \oplus \mathcal{P}_2$, přirozeně jako posloupnost hran

$$\mathcal{P}_1 \oplus \mathcal{P}_2 := e_1^1 e_2^1 \cdots e_n^1 e_1^2 e_2^2 \cdots e_n^2.$$

Určete, pro jaké cesty (v obecném grafu) $\mathcal{P}_1, \mathcal{P}_2$ platí, že
- $\mathcal{P}_1 \oplus \mathcal{P}_2 = \mathcal{P}_2 \oplus \mathcal{P}_1$;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ cesta;

- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ tah;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ sled;
- je $\mathcal{P}_1 \oplus \mathcal{P}_2$ cyklus.

(6) Dokažte, že je-li $T = (V, E)$ strom, pak $\#E = \#V - 1$.

(7) Spočtěte, kolik existuje stromů na $n$ vrcholech.

(8) (těžké) Ať $V$ je množina $n$ bodů v rovině. Pro každé dva body $x, y \in V$ definujme váhu hrany $xy$ jako vzdálenost bodů $x$ a $y$. Čili, pokud $x = (x_1, x_2)$ a $y = (y_1, y_2)$, pak

$$w(xy) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

Vzniklý graf označme obyčejně $G$.

(a) Ukažte, že v každé minimální kostře $G$ vede z každého vrcholu maximálně 6 hran.

(b) Ukažte, že existuje minimální kostra $G$, jejíž hrany se (jakožto úsečky v rovině) nekříží.

(9) *Úplným grafem* na $n$ vrcholech myslíme graf $G = (V, \binom{V}{2})$, tedy graf, mezi každým párem jehož vrcholů vede hrana. Takový graf se obvykle značí $K_n$. Najděte minimální kostru $K_n$ a spočtěte její váhu (tj, součet vah všech jejích hran), je-li $V = \{1, \ldots, n\}$ a

(a) $w(ij) = \max(i, j)$,

(b) $w(ij) = i + j$,

pro všechny páry $i, j \leq n$.

(10) Dokažte, že když $w$ (tj. ohodnocení $G$) je prosté zobrazení, pak je minimální kostra $G$ určena jednoznačně.

(11) Dokažte, že Floydův-Warshallův algoritmus selže, když připustíme i záporná ohodnocení hran.

(12) Vyřešte úlohu o rodinném výletu pro graf daný obrázkem 45.