16. Bonus: Více o Euklidově algoritmu

V této kapitole nejprve dokážeme, že Euklidův algoritmus dělá, co dělat má. Poté se podíváme na otázku jeho rychlosti a na to, zda je výsledná identita optimální. Na závěr si ukážeme alternativní velmi stručný ruční výpočet a potvrdíme si věci, kterých jsme si mohli všimnout při počítání.

Nejprve si připomeneme algoritmy, se kterými zde budeme pracovat.

Euklidův algoritmus
Iniciace: $r_0 := a$, $r_1 := b$, $k := 0$.
Dokud platí $r_{k+1} \neq 0$ opakovat krok:
$k := k + 1$, $r_{k+1} = r_{k-1} \bmod r_k$.
Pak $\gcd(a, b) = |r_k|$.

Euklidův algoritmus se zápornými zbytky
Iniciace: $r_0 := a$, $r_1 := b$, $k := 0$.
Dokud platí $r_{k+1} \neq 0$ opakovat krok:
$k := k + 1$, najdi r_{k+1} tak, aby platilo
$r_{k+1} = r_{k-1} - qr_k$ pro $q \in \mathbb{Z}$ a $-\frac{1}{2}|r_k| < r_{k+1} \leq \frac{1}{2}|r_k|$.
Pak $\gcd(a, b) = |r_k|$.

Rozšířený Euklidův algoritmus
Inicializace: $r_0 := a$, $r_1 := b$, $k := 0$,
$A_0 := 1$, $A_1 := 0$, $B_0 := 0$, $B_1 := 1$.
Dokud platí $r_{k+1} \neq 0$, opakovat kroky:
$k := k + 1$, $q_k = \left\lfloor \frac{r_{k-1}}{r_k} \right\rfloor$,
$r_{k+1} := r_{k-1} - q_k r_k$,
$A_{k+1} := A_{k-1} - q_k A_k$,
$B_{k+1} := B_{k-1} - q_k B_k$.
Pokud $r_k < 0$, změnit znaménka u r_k, A_k, B_k.
Pak $\gcd(a, b) = r_k = A_k a + B_k b$.

Rozšířený Euklidův algoritmus se zápornými zbytky
Inicializace: $r_0 := a$, $r_1 := b$, $k := 0$,
$A_0 := 1$, $A_1 := 0$, $B_0 := 0$, $B_1 := 1$.
Dokud platí $r_{k+1} \neq 0$, opakovat kroky:
$k := k + 1$, najdi $q_k \in \mathbb{Z}$ tak, aby platilo $-\frac{1}{2}|r_k| < r_{k-1} - qr_k \leq \frac{1}{2}|r_k|$.
$r_{k+1} := r_{k-1} - q_k r_k$,
$A_{k+1} := A_{k-1} - q_k A_k$,
$B_{k+1} := B_{k-1} - q_k B_k$.
Pokud $r_k < 0$, změnit znaménka u r_k, A_k, B_k.
Pak $\gcd(a, b) = r_k = A_k a + B_k b$.

V kapitole 2 jsme naznačili, jakým přístupem budeme dokazovat, že tyto algoritmy jsou správné.

Fakt 16.1.
Všechny verze Euklidova algoritmu skončí pro libovolný vstup $a, b \in \mathbb{Z}$ po konečném počtu kroků.

Jak jsme popsali v poznámce 5a.7, obecně může být otázka ukončení algoritmu velmi komplikovaná. Tady je to naštěstí vcelku snadné.

Oblíbeným nástrojem pro dokazování tvrzení o ukončení běhu algoritmu je tzv. variant. V každém kroku algoritmu vybereme či vytvoříme nějaké číslo v_k (to je ten variant), a to tak, aby pro každé k platilo $v_k > v_{k+1} > 0$. Pokud se takovýto variant podaří v algoritmu najít, pak je vyhráno a je možné udělat následující argument:

Pokud by algoritmus nikdy neskončil, tak by vygeneroval nekonečnou (ryze) klesající posloupnost přirozených čísel. Ovšem podle důsledku to není možné, proto se algoritmus musí zastavit.

V našem případě variant najdeme snadno, u klasického Euklidova algoritmu lze přímo volit $v_k = r_k$, u verze se zápornými zbytky volíme $v_k = |r_k|$. obě volby zjevně splňují požadavky kladené na variant, čímž je ukončení Euklidova algoritmu dokázáno.

Druhým úkolem je dokázat částečnou správnost, tedy že pokud se algoritmus zastaví, tak poskytne správný výsledek. To se často dělá pomocí indukce. Standardně se indukce používá s nekonečnými množinami, což by nás nutilo uměle prodlužovat všechny poslopnosti algoritmem generované na nekonenčné i v případech, kdy skončí (a my už víme, že náš algoritmus končit bude). Vyhneme se tomu tak, že teď dokážeme klíčové lemma jen o konečných úsecích čísel r_k (bez ohledu na to, zda pak algoritmus skončil nebo ne), a použijeme speciální verzi indukce pro konečné množiny, viz poznámka 5a.13.

Lemma 16.2.
Nechť $r_0, r_1, \dots, r_K, r_{K+1}$ je posloupnost čísel generovaná Euklidovým algoritmem (klasickým nebo s nezápornými zbytky), přičemž $r_0 = a \in \mathbb{Z}$ a $r_1 = b \in \mathbb{Z}$. Pak pro každé $k = 0, 1, \dots, K$ platí $\gcd(r_k, r_{k+1}) = \gcd(a, b)$.

Důkaz (poučný)**:** 1) Uvažujme nějaký běh algoritmu pro jistá vstupní data $a, b \in \mathbb{Z}$, který vytvoří čísla $r_0, r_1, \dots, r_{K+1}$.

Tvrdíme, že pro všechna $k \in \{0, 1, \dots, K\}$ platí $\gcd(r_k, r_{k+1}) = \gcd(a, b)$.
Dokážeme to matematickou indukcí.

(0) Pro $k = 0$ to platí, neboť $r_0 = a$ a $r_1 = b$. Takže $\gcd(r_0, r_1) = \gcd(a, b)$.
(1) Nechť $k \in \{0, 1, \dots, K-1\}$ a předpokládejme, že platí $\gcd(r_k, r_{k+1}) = \gcd(a, b)$. Chceme ukázat platnost vlastnosti tohoto tvrzení pro $k + 1$.

Protože $k \leq K - 1$, je $k + 1 \leq K$ a $(k + 1) + 1 \leq K + 1$, tedy čísla r_{k+1} a $r_{(k+1)+1}$ existují. Protože vznikla Euklidovým algoritmem, platí také (v případě standardních i záporných zbytků) $r_{k+2} = r_k - q_{k+1} r_{k+1}$ pro nějaké $q_{k+1} \in \mathbb{Z}$. Proto můžeme aplikovat lemma 2b.12 a máme $\gcd(r_k, r_{k+1}) = \gcd(r_{k+1}, r_{k+2})$. Podle indukčního předpokladu to znamená $\gcd(r_{k+1}, r_{(k+1)+1}) = \gcd(a, b)$.

□

Nyní jsme připraveni na důkaz, že algoritmus poskytne, co má. Uvažujme nějaká vstupní data a, b. Víme už, že algoritmus musí skončit, vznikne tedy posloupnost $r_0, r_1, \dots, r_K, r_{K+1}$, kde $r_{K+1} = 0$ a $r_K \neq 0$, přičemž dá na výstup číslo $|r_K|$. Je to správně?

Můžeme aplikovat lemma a dostáváme

$$|r_K| = \gcd(r_K, 0) = \gcd(r_K, r_{K+1}) = \gcd(a, b).$$

Algoritmus má tedy na výstupu $\gcd(a, b)$.

Nyní se podíváme na rozšířenou verzi.

Lemma 16.3.
Nechť $r_0, r_1, \dots, r_K$, $A_0, A_1, \dots, A_K$, $B_0, B_1, \dots, B_K$, jsou posloupnosti generované rozšířeným Euklidovým algoritmem (klasickým nebo s nezápornými zbytky), přičemž $r_0 = a \in \mathbb{Z}$ a $r_1 = b \in \mathbb{Z}$. Pak pro každé $k = 0, 1, \dots, K$ platí $r_k = A_k a + B_k b$.

Důkaz (poučný)**:** 1) Uvažujme nějaký běh algoritmu pro jistá vstupní data $a, b \in \mathbb{Z}$, který vytvoří čísla r_k, A_k a B_k pro $k = 0, 1, \dots, K$.

Rovnost $r_k = A_k a + B_k b$ dokážeme silnou indukcí, modifikovanou pro dva kroky zpět.

(0) Počáteční hodnoty A_0, A_1, B_0, B_1 jsou nastaveny tak, že $a = A_0 a + B_0 b$ a $b = A_1 a + B_1 b$. Protože $r_0 = a$ a $r_1 = b$, rovnost platí pro $k = 0$ a $k = 1$.
(1) Nechť $k \in \{1, 2, \dots, K_1\}$ a předpokládejme, že platí, že $r_k = A_k a + B_k b$ a $r_{k-1} = A_{k-1} a + B_{k-1} b$. Chceme ukázat platnost této rovnosti pro $k + 1$.

r_{k+1} vzniklo tak, že bylo nalezeno vhodné q_k a spočítáno $r_{k+1} = r_{k-1} - q_k r_k$, $A_{k+1} = A_{k-1} - q_k A_k$, $B_{k+1} = B_{k-1} - q_k B_k$. Spolu s indukčními předpoklady to dává

$$\begin{aligned} r_{k+1} &= r_{k-1} - q_k r_k = (A_{k-1} a + B_{k-1} b) - q_k (A_k a + B_k b) \\ &= (A_{k-1} - q_k A_k) a + (B_{k-1} - q_k B_k) n = A_{k+1} a + B_{k+1} b. \end{aligned}$$

Rovnost pro $k + 1$ je dokázána.

□

V okamžiku ukončení běhu algoritmu máme generovaná čísla $r_K \neq 0$ a $r_{K+1} = 0$. Podle prvního lemma je $r_K = \gcd(a, b)$, dle druhého pak $r_K = A_K a + B_k b$, tedy $\gcd(a, b) = A_K a + B_K b$. Rozšířený Euklidův algoritmus umí najít Bezoutovu identitu.

16a. Rychlost algoritmu a kvalita výstupu

To, jak dlouho si na výpočet počkáme, se odvíjí od řady faktorů, a jedním z dominantních je počet algebraických operací, které je nutno vykonat. V případě Euklidova algoritmu toto zjevně souvisí s tím, kolikrát musíme zopakovat základní krok. V kapitole 2 jsem ukázali, že u standardní podoby, tedy s nezápornými zbytky, generovaná čísla r_k splňují zajímavou sčítací vlastnost, kdy $r_{k+1} + r_k$ nemůže být větší než r_{k-1}. Tím je vynuceno, že r_{k+1} nemohou klesat velmi pomalu.

Tato vlastnost mohla čtenáři připomenout známou Fibonacciho posloupnost $\{F_m\}$, viz příklad 9a.c. Toto pozorování je klíčem k následujícímu výsledku. Protože se ukončení rozhoduje dle hodnoty r_k a tato je používána i pro rozšířený algoritmus, budou všechna pozorování aplikovatelná i na rozšířenou verzi. Na druhou stranu se zde omezíme pouze na standardní podobu algoritmu, protože u záporných zbytků k řadě zajímavých jevů přestává docházet. Z podobných důvodů se také omezíme jen na kladné vstupy.

Věta 16a.1. (Lamého věta)
Nechť $a > b \in \mathbb{N}$. Pak (rozšířený) Euklidův algoritmus pro $\gcd(a, b)$ vyžaduje nanejvýš tolik kroků, kolik je pětkrát počet cifer v b.

Tento výsledek (z roku 1844) je jedním z počátků zkoumání výpočetní náročnosti algoritmů. Zároveň se v něm překvapivě skrývají dva populární mystické objekty, Fibonacciho posloupnost a zlatý řez. Vše ukáže důkaz, který je spíše pro odvážnější čtenáře.

Důkaz (drsný, poučný)**:** Uvažujme teď běh algoritmu pro konkrétní $a > b > 0$ a předpokládejme, že skončil po K krocích. To znamená, že vytvořil posloupnost $r_0, \dots, r_{K+1}$ pro jisté $K \in \mathbb{N}$ s $r_K \neq 0$ a $r_{K+1} = 0$.

Protože $a > b > 0$, jsou všechna r_k generovaná algoritmem včetně prvních dvou nezáporná a tvoří klesající posloupnost. Uvažujme výpočet $r_{k+1} = r_{k-1} - q_k r_k$, kde $q_k \geq 0$. Protože je $r_{k-1} > r_{k+1}$, nemohlo nastat $q_k = 0$, tedy $q_k \geq 1$ a můžeme odhadovat $r_{k-1} = q_k \cdot r_k + r_{k+1} \geq r_k + r_{k+1}$.

Tvrdíme, že pro $k = 0, \dots, K-1$ platí $r_{K-k} \geq F_{k+2}$, kde F_m jsou Fibonacciho čísla. Dokážeme to speciální konečnou indukcí na $k \in \{0, \dots, K-1\}$ (modifikovaným principem s návratem o dva kroky).

(0) Pro $k = 0$ rovnost říká $r_K \geq F_2$, tedy $r_K \geq 1$, což je pravda, r_K je poslední nenulový zbytek.

Pro $k = 1$ tvrzení říká, že $r_{K-1} \geq F_3$, tedy $r_{K-1} \geq 2$. Je to pravda? Protože $r_{K+1} = 0$, je $r_{K-1} = q_K \cdot r_K$. Kdyby bylo $q_K = 1$, pak dostáváme $r_{K-1} = r_K$, což není možné. Je tedy $q_K \geq 2$ a máme opravdu $r_{K-1} \geq 2r_K \geq 2$.

(1) Uvažujme nějaké $k \in \{0, 1, \dots, K-2\}$ a předpokládejme, že tvrzení platí pro $0, \dots, k$. Chceme odhad pro $r_{K-(k+1)}$. Použijeme nerovnost odvozenou na začátku důkazu, indukční předpoklad a základní vlastnost Fibonacciho čísel k odvození

$$\begin{aligned} r_{K-(k+1)} = r_{K-k-1} &\geq r_{K-k-1+1} + r_{K-k-1+2} = r_{K-k} + r_{K-(k-1)} \\ &\geq F_{k+2} + F_{(k-1)+2} = F_{k+2} + F_{k+1} = F_{k+3} = F_{(k+1)+2}. \end{aligned}$$

Indukční krok (1) je potvrzen. Víme tedy, že $r_{K-k} \geq F_{k+2}$ platí pro $0 = 1, \dots, K-1$. Použijeme nerovnost pro $k = K-1$ a dostáváme $r_1 \geq F_{K+1}$ neboli $b \geq F_{K+1}$. Co z toho plyne pro K?

Z výsledků příkladu 9a.c máme následující. Když označíme $\alpha = \dfrac{1+\sqrt{5}}{2}$, pak $F_m \geq \alpha^{m-1}$. Pro nás to znamená, že $b \geq \alpha^K$, proto $K \log_{10}(\alpha) \leq \log_{10}(b)$. Kalkuliho přístroj potvrdí, že $\log_{10}(\alpha) > \frac{1}{5}$, z čehož dostaneme $K \leq 5 \log_{10}(b)$. Důkaz je hotov.

□

V kapitole 2 jsme ukázali, že při použití záporných zbytků dostaneme (pro větší b) odhad $K \leq 3.4 \log_{10}(b)$. Důkaz byl výrazně snažší.

Určili jsme počet kroků, celková výpočetní náročnost algoritmu (viz kapitola 11b) ještě musí vzít v úvahu náklady na počítání modula v každém kroku, popřípadě to, že výpočty s většími čísly mohou trvat déle. Dá se ukázat, že délka běhu programu je úměrná číslu $[\log(b)]^2$. Z hlediska computer science je jednodušší hovořit o bitech zápisu čísla, můžeme například říct, že při práci s n-bitovými čísly b vyžaduje Euklidův algoritmus zhruba dobu n^2.

Víme, že pro daná čísla $a, b \neq 0$ existuje nekonečně mnoho Bezoutových identit. To znamená, že lze najít libovolně velká čísla A, B splňující Bezoutovu identitu. Teoreticky je to jedno, ale v praxi jistě dáme přednost vyjádření $\gcd(141, 94) = 47 = 1 \cdot 141 + (-1) \cdot 94$ před konkurentem $47 = 200003 \cdot 141 + (-300004) \cdot 94$.

Mohli jste si všimnout, že při počítání Euklidova algoritmu jsou generovaná čísla A_k postupně stále větší, totéž platí pro B_k (důkaz najdete na konci kapitoly). Máme tedy motivaci skončit co nejdříve. Pokud například máme čísla $a > b > 0$, ale spustíme Euklidův algoritmus se vstupy b, a, pak (jak jsme již viděli) algoritmus nejprve v jednom kroku pozice prohodí, což prodlouží běh a způsobí, že výsledné konstanty A, B budou zbytečně velké.

V jak malá čísla můžeme doufat? Jsou to vlastně dvě otázky, jednak se ptáme, co je vůbec možné, a pak je tu otázka, zda to umíme získat. Matematici hledali vyjádření typu „určitě existuje dvojice A, B, která je v porovnání s a, b ne moc velká“. Hledáme tedy nějaké výrazy h_a, h_b závisející na a, b, které by nám omezily A, B shora. Samozřejmě se snažíme, aby ty h_a, h_b byly co nejmenší, ale aby pořád bylo zaručeno, že se nějaká A, B splnující $|A| \leq h_a$, $|B| \leq h_b$ najdou. Jaký typ omezení lze čekat?

Podívejme se na případ, kdy jsou čísla a, b hodně velká a nesoudělná. Bezoutova identita nám poskytne celá čísla $A, B \in \mathbb{Z}$ taková, že $Aa + Bb = 1$. Přepišme si to jako $Aa = 1 - Bb$. Protože $A, B \neq 0$, máme $|Aa| \geq |a|$ a $|Bb| \geq |b|$, v rovnici jsou dvě opravdu velká čísla a pak jednička, kterou lze tedy zanedbat bez vzniku velké chyby. Tím vzniká $Aa \sim -Bb$. Dostáváme tak zajímavý odhad pro Bezoutovy koeficienty $\frac{A}{B} \sim -\frac{b}{a}$. Protože se bavíme o velikostech, přepišme si to jako $\frac{|A|}{|B|} \sim \frac{|b|}{|a|}$. To znamená, že existuje nějaké číslo c takové, že $A \sim cb$ a $B \sim -ca$. Toto jsou jen hrubé odhady pro případ velkých čísel a, b, ale říkají nám něco užitečného. Velikosti A a B se odvozují od čísel b a a (takhle „do kříže“). Přesněji řečeno, má smysl hledat vyjádření typu „A je relativně malé vzhledem k b a B relativně malé vzhledem k a“.

Hledání správných výrazů h_a, h_b je v jistém smyslu otázka rovnováhy. Stává se totiž, že jsme schopni jednu konstantu udělat velice malou, ale za cenu velkého zvětšení té druhé. Z praktického pohledu je užitečné, aby obě konstanty byly rozumně malé, vycházelo se tedy z onoho křížového ovlivnění pozorovaného výše. Ukázalo se, že takové dosti optimistické omezení existuje. Navíc ještě platí, že rozšířený Bezoutův algoritmus (ve standardní podobě a s uspořádanými kladnými vstupy) právě takto příjemné dvojice nachází.

Věta 16a.2.
Uvažujme čísla $a > b > 0$. Pak rozšířený Euklidův algoritmus se vstupními hodnotami $a > b > 0$ má na výstupu čísla A_K, B_K splňující $|A_K| \leq \frac{b}{2\gcd(a,b)}$ a $|B_K| \leq \frac{a}{2\gcd(a,b)}$.

Než se dáme do důkazu, stojí za zmínku, že ony dva horní odhady ve větě již nelze dále vylepšit. Máme totiž $\gcd(3,2) = 1 = 1 \cdot 3 + (-1) \cdot 2$. V tomto případě je $|A| = 1 = \frac{2}{2} = \frac{b}{2\gcd(3,2)}$, tedy koeficient A je přesně roven onomu hornímu odhadu a je jasné, že menší celé číslo (v absolutní hodnotě) než 1 už v této Bezoutově identitě nenajdeme. Jinak řečeno, pokud by někdo chtěl napsat tvrzení, ve kterém by jako horní odhad pro koeficienty použil něco menšího, tak se $\gcd(3,2)$ stane protipříkladem.

V kapitole o diofantických rovnicích jsme také viděli, že v párech Bezoutových koeficientů (A, B) se hodnoty A mohou lišit o násobky $\frac{b}{\gcd(a,b)}$ a hodnoty B o násobky $\frac{a}{\gcd(a,b)}$. To znamená, že kromě minimálních koeficientů poskytnutých rozšířeným Euklidovým algoritmem už neexistuje žádný další pár splňující dokazovaný horní odhad.

Tento výsledek je už dost hluboký, čemuž také odpovídá dobrodružný důkaz pro odvážné čtenáře.

Důkaz (dobrý, z povinnosti)**:** Důkaz povedeme indukcí na počet kroků programu nutný k dosažení výsledku, tedy ke stavu $r_K > 0$ a $r_{K+1} = 0$. Dokážeme následující tvrzení:

V(k): Pokud rozšířený Euklidův algoritmus aplikovaný na nějaká celá čísla $\alpha > \beta > 0$ skončí po K krocích ve stavu $r_K > 0$ a $r_{K+1} = 0$, pak $|A_K| \leq \frac{\beta}{2\gcd(\alpha,\beta)}$ a $|B_K| \leq \frac{\alpha}{2\gcd(\alpha,\beta)}$.

(0) $K = 1$. Pokud běh skončí hned v prvním kroku, máme situaci $r_0 = \alpha$, $r_1 = \beta$ a $r_2 = \alpha \bmod \beta = 0$. Zajímají nás registry $A_1 = 0$ a $B_1 = 1$. Evidentně platí $|A_1| = 0 \leq \frac{\beta}{2\gcd(\alpha,\beta)}$.

Protože $r_2 = \alpha \bmod \beta = 0$, tak β dělí α, tedy $\gcd(\alpha, \beta) = \beta$. Protože $\alpha > \beta$, musí platit $\frac{\alpha}{\gcd(\alpha,\beta)} = \frac{\alpha}{\beta} \geq 2$, tedy opravdu $\frac{\alpha}{2\gcd(\alpha,\beta)} = \frac{\alpha}{2\beta} \geq 1 = |B_1|$.

(1) Nechť $K \in \mathbb{N}$ je libovolné. Předpokládejme, že libovolný běh rozšířeného Euklidova algoritmu, který pro nějaké vstupy $\alpha > \beta > 0$ končí po K krocích, má na výstupu čísla splňující žádanou nerovnost.

Potřebujeme ukázat, že i běhy končící po $K + 1$ krocích dávají správné výstupy.

Uvažujme proto nějaká čísla $\alpha > \beta > 0$ taková, že když Euklidův algoritmus skončil, tak měl na výstupu čísla A_{K+1}, B_{K+1}. Nechť $\alpha = q\beta + r$. Protože program nekončí prvním krokem, je $r > 0$. Registry byly po prvním kroku algoritmu ve stavu $r_0 = \alpha$, $r_1 = \beta$, $r_2 = r$, $q_1 = q = \lfloor \frac{\alpha}{\beta} \rfloor$, dále $A_0 = 1$, $A_1 = 0$, $A_2 = 1$ a $B_0 = 0$, $B_1 = 1$, $B_2 = -q$.

Uvažujme běh rozšířeného Euklidova algoritmu aplikovaného na vstup $\alpha' = \beta$ a $\beta' = r$, který vede na čísla r'_k, A'_k a B'_k. Stručně řečeno, prostě jsme začali s druhým a třetím řádkem původní Euklidovské tabulky a počítali znovu (čtenáři pomůže, pokud si teď jeden takový konkrétní případ spočítá, aby mohl porovnávat s důkazem). Nejprve se podíváme jen na levý sloupec, tedy klasický Euklidův algoritmus.

Iniciační hodnoty nového běhu jsou $r'_0 = b$ a $r'_1 = r$. V prvním kroku počítáme $q'_1 = \lfloor \frac{b}{r} \rfloor$ a $r'_2 = b \bmod r$. To je ovšem stejné, jako když jsme v původním běhu počítali druhý krok, $q_2 = \lfloor \frac{b}{r} \rfloor$ a $r_3 = b \bmod r$. Oba algoritmy dál počítají stejné výpočty se stejnými čísly, budou se tedy lišit jen číslem kroku, tedy indexem. Tvrdíme proto,

že pro $k = 1, \dots, K+1$ platí $r_k = r'_{k-1}$. Toto pomocné tvrzení dokážeme indukcí na k (indukce v indukci). Protože při výpočtu r_{k+1} potřebujeme znát dva předchozí stavy, použijeme silnou indukci modifikovanouna dva zpětné kroky.

(p0) Iniciační hodnoty ukazují, že vzorec platí pro $k = 1$ a $k = 2$.

(p1) Nechť $k \in \{2, 3, \dots, K\}$. Předpokládejme, že $r_k = r'_{k-1}$ a $r_{k-1} = r'_{k-2}$. Pak

$$r_{k+1} = r_k \bmod r_{k-1} = r'_{k-1} \bmod r'_{k-2} = r'_k = r'_{(k+1)-1}.$$

Tím je vnořená indukce hotova, vzorec $r_k = r'_{k-1}$ je potvrzen. Pak už snadno odvodíme, že

$$q_k = \lfloor \frac{r_{k-1}}{r_k} \rfloor = \lfloor \frac{r'_{k-2}}{r'_{k-1}} \rfloor = q'_{k-1}.$$

Když se ale podíváme na první krok nového běhu u registrů A_k a B_k, najdeme tam změnu. Sice nové hodnoty počítáme vzorcem $A'_2 = A'_0 - q'_1 A'_1$, který je stejný jako u druhého kroku původního běhu, tam $A_3 = A_1 - q_2 A_2$ a víme už, že $q_2 = q'_1$, jenže se neshodují vstupní hodnoty. Zatímco první krok nového běhu má $A_0 = 1$, $B_0 = 0$, $A_1 = 0$ a $B_1 = 1$ (spouštíme algoritmus od začátku), původní běh pracoval s hodnotami $A_1 = 0$, $B_1 = 1$, $A_2 = 1$, $B_2 = -q$. Pokud jste si spočítali nějaký konkrétní příklad, tak jste viděli, že sloupce pro A a B měly v novém běhu jiné hodnoty.

Naštěstí lze mezi nimi najít převod. Tvrdíme, že pro $k = 1, \dots, K+1$ platí $A_k = B'_{k-1}$ a $B_k = A'_{k-1} - qB'_{k-1}$. Důkaz je opět silnou indukcí na k s návratem o dva kroky.

(0) Pro $k = 1$ máme $A_1 = 0 = B'_0$, $B_1 = 1 = A'_0 - qB'_0$.

Pro $k = 2$ máme $A_2 = 1 = B'_1$, $B_2 = -q = A'_1 - qB'_1$.

(1) Vezměme nějaké $k \in \{2, 3, \dots, K\}$ a předpokládejme, že vzorce platí pro k a $k-1$. Registry pro $k+1$ jsou počítány stejným vzorcem pomocí stejné konstanty q_k, proto máme (s využitím $q_k = q'_{k-1}$)

$$\begin{aligned} A_{k+1} &= A_{k-1} - q_k A_k = B'_{k-2} - q'_{k-1} B'_{k-1} = B'_k, \\ B_{k+1} &= B_{k-1} - q_k B_k = (A'_{k-2} - qB'_{k-2}) - q'_{k-1}(A'_{k-1} - qB'_{k-1}) \\ &= (A'_{k-2} - q'_{k-1} A'_{k-1}) - q(B'_{k-2}) - q'_{k-1} B'_{k-1}) = A'_k - qB'_k. \end{aligned}$$

Tím je tato vnořená indukce hotova.

Použijeme její závěr pro $k = K+1$ a dostáváme $A_{K+1} = B'_K$ a $B_{K+1} = A'_K - qB'_K$. O číslech ze zkráceného běhu víme, že dávají nejmenší společný dělitel, $\gcd(\beta, r) = \gcd(\alpha, \beta) = A'_K \alpha + B'_k \beta$.

Zároveň díky indukčnímu předpokladu (teď se vracíme k vnější indukci na K) aplikovanému na zkrácený běh víme, že $|A'_K| \le \frac{r}{2\gcd(\beta,r)} = \frac{r}{2\gcd(\alpha,\beta)}$ a $|B'_K| \le \frac{\beta}{2\gcd(\beta,r)} = \frac{\beta}{2\gcd(\alpha,\beta)}$.

Máme tedy $|A_{K+1}| = |B'_K| \le \frac{\beta}{2\gcd(\alpha,\beta)}$, přesně jak jsme chtěli. Dále odhadujeme

$$|B_{K+1}| \le |A'_K| + q|B'_K| \le \frac{r}{2\gcd(\alpha,\beta)} + q\frac{\beta}{2\gcd(\alpha,\beta)} = \frac{r + q\beta}{2\gcd(\alpha,\beta)} = \frac{\alpha}{2\gcd(\alpha,\beta)}.$$

Tím jsme potvrdili, že běhy o $K+1$ krocích také dávají na výstupu řešení dle tvrzení a indukce je hotova.

□

16b. Zajímavosti o Euklidově algoritmu

16b.1 Poznámka (alternativa k tabulkovému výpočtu):

Existuje algoritmus pro ruční výpočet, který je úspornější, vychází ze zápisu Euklidova algoritmu do řádku. Probíhá ve dvou fázích, začneme tak, že prostě provedeme Euklidův algoritmus s tím, že do prvního řádku píšeme hodnoty a, b a pod to odpovídající hodnoty q.

a, b	408	108	84	24	12	0
q		3	1	3	2	

Teď vytvoříme třetí řádek, ten ale děláme zprava doleva. Nejprve pod poslední hodnoty q napíšeme (zprava) 0 a 1.

a, b	408	108	84	24	12	0
q		3	1	3	2	
				1	0	

Jsme připraveni k druhé fázi algoritmu, která funguje následovně: Vezmeme poslední číslo v třetím řádku (teď 0) a odečteme od něj předposlední číslo (teď 1) vynásobené koeficientem q nad předposledním číslem (teď 3), dostaneme $0 - 1 \cdot 3 = -3$ a napíšeme to doleva od předposledního čísla, tedy pod jedničku. Proces opakujeme, jen se vzorec přesune v tabulce o pole doleva. Vezmeme druhé číslo zprava (jedničku), odečteme číslo nalevo (tedy -3) vynásobené číslem nad ním (jedničkou), výsledek $1 - (-3) \cdot 1 = 4$ zapíšeme ještě o jedno doleva. V posledním kroku počítáme $-3 - 4 \cdot 3 = -15$ a zapíšeme zcela doleva.

a,b	408	108	84	24	12	0
q		3	1	3	2	
	-15	4	-3	1	0	

Co teď s tím? Začněme zprava, v horním a dolním řádku vidíme napravo dvojice $12 \leftrightarrow 0$ a $24 \leftrightarrow 1$. Když hodnoty z dolního řádku prohodíme, dostaneme $12 \leftrightarrow 1$ a $24 \leftrightarrow 0$ a $12 \cdot 1 + 24 \cdot 0 = 12 = \gcd(408, 108)$. Posuňme se o jedno doleva. Porovnáním horního a dolního řádku máme dvojice $24 \leftrightarrow 1$ a $84 \leftrightarrow -3$, prohodíme na $24 \leftrightarrow -3$ a $84 \leftrightarrow 1$ a dostaneme $24 \cdot (-3) + 84 \cdot 1 = 12$. Další posun si zkusíte sami. Happy end: V levých dvou sloupcích máme $408 \leftrightarrow -15$ a $108 \leftrightarrow 4$, prohodíme na $408 \leftrightarrow 4$ a $108 \leftrightarrow -15$ a dostaneme $408 \cdot 4 + 108 \cdot (-15) = 12$, což je přesně hledaná Bezoutova identita. Náhoda? Nikoliv, takto to funguje vždy.

Tento způsob je bezesporu nejúspornější, ale je už docela neprůhledný, je tedy více náchylný na chybu. Tabulková metoda, kterou zde preferujeme, je nejen průhlednější, ale také nám umí pomoci s řešením diofantických rovnic.

Protože je vypadá trochu jako černá magie, asi nás zajímá důkaz, že tento algoritmus opravdu funguje. Nejprve se vrátíme ke značení z důkazu správnosti Euklidova algoritmu, kde jsme zvolili $r_0 = a$, $r_1 = b$ a poté definovali rekurzí $q_k = \left\lfloor \frac{r_{k-1}}{r_k} \right\rfloor$, $r_{k+1} = r_{k-1} - q_k r_k$. Toto se opakuje, dokud nenastane $r_{K+1} = 0$, pak $r_K = \gcd(a, b)$.

Nyní definujeme $s_0 = 0$, $s_1 = 1$ a $s_{k+1} = s_{k-1} - q_{n-k} s_k$ pro $k = 1, \dots, K-1$. Tvrdíme, že pro každé $k = 1, \dots, K$ platí $r_{n-k+1} s_k + r_{n-k} s_{k-1} = r_n$, dokážeme to indukcí.

(0) Pro $k = 1$ začneme levou stranou vzorce: $r_{n-1+1} s_1 + r_{n-1} s_0 = r_n \cdot 1 + r_{n-1} \cdot 0 = r_n$, souhlasí.

(1) Nechť $k \in \{1, \dots, K-1\}$, předpokládejme, že $r_{n-k+1} s_k + r_{n-k} s_{k-1} = r_n$. Dosadíme za r_{n-k+1} z rekurentní definice a po úpravě dostaneme vzorec, který potřebujeme pro $k+1$:

$$\begin{aligned} r_n &= r_{n-k+1} s_k + r_{n-k} s_{k-1} = (r_{n-k-1} - q_{n-k} r_{n-k}) s_k + r_{n-k} s_{k-1} = r_{n-k-1} s_k - q_{n-k} r_{n-k} s_k + r_{n-k} s_{k-1} \\ &= r_{n-k-1} s_k + r_{n-k}(s_{k-1} - q_{n-k} s_k) = r_{n-k-1} s_k + r_{n-k} s_{k+1} = r_{n-k} s_{k+1} + r_{n-k-1} s_k \\ &= r_{n-(k+1)+1} s_{k+1} + r_{n-(k+1)} s_{(k+1)-1}. \end{aligned}$$

Tím je naše tvrzení dokázáno. Teď jej použijeme pro $k = K$:

$$\gcd(a, b) = r_K = r_1 s_K + r_0 s_{K-1} = b \cdot s_K + a \cdot s_{K-1}.$$

Opravdu tedy Bezoutovu identitu získáme pomocí posledních dvou čísel zpětného chodu, když je prohodíme. △

Vrátíme se k doporučenému tabulkovému způsobu výpočtu. Už jsme těch tabulek několik spočítali a možná jste si všimli některých zajímavostí. Jako inspiraci se podívejme na následující tabulku.

a,b	A	B	
33	1	0	
13	0	1	2
7	1	-2	1
6	-1	3	1
1•	2•	-5•	6
0	-13	33	

Vidíme například, že v každém řádku jsou A_k, B_k nesoudělné. To se dá relativně snadno dokázat.

Věta 16b.2.
Nechť A_k, B_k pro $k = 0, 1, \dots, K+1$ jsou čísla generovaná jedním během rozšířeného Euklidova algoritmu. Pak pro všechna tato k platí, že A_k, B_k jsou nesoudělná čísla.

Důkaz (poučný, dobrý)**:** Tento důkaz je trochu trikový, nejprve totiž dokážeme indukcí, že pro $k \in \{0, 1, \dots, K\}$ platí zdánlivě nesouvisející vzorec $A_k B_{k+1} - A_{k+1} B_k = (-1)^k$.

(0) $k = 0$: Algoritmus má nastaveno $A_0 = 1$, $B_0 = 0$, $A_1 = 0$, $B_1 = 1$, proto $A_0 B_1 - A_1 B_0 = 1 = (-1)^0$.

(1) Nechť $k \in \{0, 1, \dots, K-1\}$, předpokládáme, že $A_k B_{k+1} - A_{k+1} B_k = (-1)^k$. Chceme dokázat vzorec pro $k+1$. Začneme levou stranou, kde si dosadíme za čísla z kroku $k+2$ odpovídající vzorce z algoritmu, a po úpravě použijeme indukční předpoklad.

$$\begin{aligned} A_{k+1} B_{k+2} - A_{k+2} B_{k+1} &= A_{k+1}(B_k - q_{k+1} B_{k+1}) - (A_k - q_{k+1} A_{k+1}) B_{k+1} \\ &= A_{k+1} B_k - A_{k+1} q_{k+1} B_{k+1} - A_k B_{k+1} + q_{k+1} A_{k+1} B_{k+1} \\ &= -(A_k B_{k+1} - A_{k+1} B_k) = -(-1)^k = (-1)^{k+1}. \end{aligned}$$

Důkaz (1) je hotov.

Teď tedy vezměme nějaké $k \in \{0, 1, \dots, K\}$. Pak platí $A_k B_{k+1} - A_{k+1} B_k = (-1)^k$. Úpravou dostáváme $[(-1)^k B_{k+1}] A_k - [(-1)^k A_{k+1}] B_k = (-1)^{2k} = 1$. To znamená, že existuje celočíselná lineární kombinace čísel A_k, B_k, která dává jedničku. Podle Důsledku 2b.20 proto $\gcd(A_k, B_k) = 1$.

Všimli jste si, že náš důkaz jednu hodnotu vynechal? Pokud zvolíme $k = K$ a podíváme se na rovnost $[(-1)^K A_K]B_{K+1} - [(-1)^K B_k]A_{K+1} = 1$, tak vidíme, že také $\gcd(A_{K+1}, B_{K+1}) = 1$. Teď je to všechno.

□

Další zajímavá věc v našem inspiračním příkladě je, že se v posledním řádku (s nulou vlevo) objevila znovu původní čísla (až na znaménko). Pokud jste si spočítali víc takovýchto běhů pro nesoudělná čísla, tak se to vždycky opakovalo. Náš oblíbený příklad 2b.b ovšem ukazuje, že v případě soudělných čísel už to nefunguje. Existuje vzorec, který potvrdí naše pozorování pro nesoudělná čísla a rozšíří jej na obecný případ.

Věta 16b.3.
Nechť A_k, B_k pro $k = 0, 1, \ldots, K+1$ jsou čísla generovaná jedním během rozšířeného Euklidova algoritmu ukončeného stavem $r_{K+1} = 0$. Pak

$$A_{K+1} = (-1)^{K+1}\frac{b}{\gcd(a,b)}, \qquad B_{K+1} = (-1)^K\frac{a}{\gcd(a,b)}.$$

Důkaz (poučný, dobrý)**:** Tento výsledek dokážeme indukcí na K, stejně jako jsme to dělali v důkazu Věty 16a.2. Nejmenší možné K pro $b \neq 0$ je $K = 1$, kdy algoritmus skončí hned prvním krokem.

(0) $K = 1$. To znamená, že $r_2 = a \bmod b = 0$, tedy b dělí a. Pak $\gcd(a, b) = b$ a $q_1 = \frac{a}{b}$. Dostáváme $A_2 = 1$ a $B_2 = -q$ neboli $A_2 = 1 \cdot \frac{b}{b} = (-1)^2\frac{b}{\gcd(a,b)}$ a $B_2 = (-1) \cdot \frac{a}{b} = (-1)^1\frac{a}{\gcd(a,b)}$.

(1) Nechť $K \in \mathbb{N}$ je libovolné. Předpokládejme, že pro všechny běhy algoritmu končící po K krocích již platí dokazované vzorce. Uvažujme teď nějaký běh algoritmu s počátečními hodnotami $a > b > 0$, který skončil až po $K + 1$ krocích, tedy $r_{K+2} = 0$. Chceme dokázat, že vzorce platí i pro něj.

Tento běh programu měl vstupní hodnoty $r_0 = a$, $A_0 = 1$, $B_0 = 0$ a $r_1 = b$, $A_1 = 0$, $B_1 = 1$. Po prvním kroku přibyly hodnoty $r_2 = r = a \bmod b$, $A_2 = 1$, $B_2 = -q$, kde q splňuje $a = qb + r$. Protože nastal ještě další krok, je $r > 0$.

Uvažujme teď alternativní běh programu s počátečními hodnotami b, r, který vygeneruje hodnoty r'_k, A'_k, B'_k. Z důkazu Věty 16a.2 již víme, že skončil po K krocích a platí $r_k = r'_{k-1}$, $A_k = B'_{k-1}$ a $B_k = A'_{k-1} - qB'_{k-1}$. Podle indukčního předpokladu také $A'_{K+1} = (-1)^{K+1}\frac{r}{\gcd(b,r)}$ a $B'_{K+1} = (-1)^K\frac{b}{\gcd(b,r)}$. Pak

$$A_{K+2} = B'_{K+1} = (-1)^K\frac{b}{\gcd(b,r)} = (-1)^{K+2}\frac{b}{\gcd(a,b)} = (-1)^{(K+1)+1}\frac{b}{\gcd(a,b)},$$

$$\begin{aligned} B_{K+2} &= A'_{K+1} - qB'_{K+1} = (-1)^{K+1}\frac{r}{\gcd(b,r)} - q(-1)^K\frac{b}{\gcd(b,r)} = (-1)^{K+1}\frac{r}{\gcd(a,b)} + q(-1)^{K+1}\frac{b}{\gcd(a,b)} \\ &= (-1)^{K+1}\frac{r+qb}{\gcd(a,b)} = (-1)^{K+1}\frac{a}{\gcd(a,b)}. \end{aligned}$$

Důkaz je hotov.

□

Mimochodem se tím potvrdila další věc, které jste si už asi všimli, že pro kladná a, b mají Bezoutovy koeficienty vždy opačná znaménka.

To lze docela snadno dokázat přímo pro všechny řádky v tabulce (kromě prvních dvou).

Fakt 16b.4.
Nechť A_k, B_k pro $k = 0, 1, \ldots, K+1$ jsou čísla generovaná jedním během rozšířeného Euklidova algoritmu se vstupem $a > b > 0$. Pak
(i) A_k je kladné a B_k záporné pro $k \geq 2$ sudá, naopak pro $k \geq 2$ lichá,
(iii) $|A_{k+1}| \geq |A_k| + |A_{k-1}|$ a $|B_{k+1}| \geq |B_k| + |B_{k-1}|$ pro $k \geq 1$.

Důkaz: Jako obvykle to budeme dokazovat indukcí.

V prvním kroku odečteme b od a alespoň jednou, nechť je to q_1 krát. Pak $A_2 = 1 - q_1 \cdot 0 = 1$ (kladné) a $B_2 = 0 - q_1 \cdot 1 = -q_1$ (záporné).

Máme také $|A_2| = 1 = 0 + 1 = |A_1| + |A_0|$ a $|B_2| = q_1 \geq= 1 = 1 + 0 = |B_1| + |B_0|$

Pokud algoritmus udělal i druhý krok, tak se $r = a - q_1 b$ alespoň jednou odečetlo od b, nechť je to q_2 krát. Pak $A_3 = 0 - q_2 \cdot 1 = -q_2$ (záporné) a $B_3 = 1 - q_2 \cdot (-q_1) = 1 + q_1 q_2$ (kladné).

Máme také $|A_3| = q_2 \geq 1 = 1 + 0 = |A_2| + |A_1|$ a $|B_3| = 1 + q_1 q_2 \geq 1 + q_1 = |B_1| + |B_2|$.

Pokud algoritmus udělal více než tři kroky, postupujeme indukcí:

Pro $k \geq 3$ dokážeme, že znaménka A_k, B_k jsou dle tvrzení a $|A_k| \geq |A_{k-1}| + |A_{k-2}|$, $|B_k| \geq |B_{k-1}| + |B_{k-2}|$.
(0) Pro $k = 3$ jsme to právě potvrdili.
(1) Uvažujme $k \geq 4$, předpokládejme platnost tvrzení pro toto k.
Konstanty s indexem $k+1$ vznikly pomocí koeficientu q_k takto: $A_{k+1} = A_{k-1} - q_k A_k$, $B_{k+1} = B_{k-1} - q_k B_k$.
Případ k sudé: Pak $A_k > 0$ a $A_{k-1} < 0$, proto $-q_k A_k < 0$ a tedy $A_{k+1} < 0$. Také proto platí

$$|A_{k+1}| = -A_{k-1} + q_k A_k \geq -A_{k-1} + A_k = |A_{k-1}| + |A_k|.$$

Dále máme $B_k < 0$ a $B_{k-1} > 0$, proto $-q_k B_k > 0$ a tedy $B_{k+1} > 0$. Také proto platí

$$|B_{k+1}| = B_{k-1} - q_k B_k \geq B_{k-1} - B_k = |B_{k-1}| + |B_k|.$$

Případ k liché se udělá obdobně.

□

Z části (ii) mimo jiné plyne, že posloupnost $\{|A_k|\}$ je neklesající pro $k \geq 2$ a (ryze) rostroucí pro $k \geq 3$, zatímco $\{|B_k|\}$ je (ryze) rostoucí už pro $k \geq 2$. Ve skutečnosti je ale situace ještě horší. Dá se snadno odvodit, že tyto dvě posloupnosti musí růst alespoň tak rychle, jako Fibonacciho posloupnost, což je bohužel exponenciální rychlost růstu. Jsme tedy opravdu rádi, že počet iterací je relativně malý a nenechá koeficienty příliš narůst.

Tím končí naše exkurze po krajině Euklidova algoritmu.