

Úkol 1 – Nelineární zobrazení [2b]

Hned nám napovídá slovo *nelineární*. Příklad funkce, které není lineární je kvadratická funkce. Jak by fungovalo kvadratické zobrazení? Uveďme nejjednodušší možnou kvadratickou funkci:

$$f(\mathbf{x}) = \mathbf{x}^2$$

Tato funkce vrátí vektor, který je transformovaný podle své vlastní velikosti, tedy místo rovnice $f(a\mathbf{x}) = af(\mathbf{x})$ by platila rovnice $f(a\mathbf{x}) = a^2f(\mathbf{x})$.

Úkol 2 – Fibonacciho čísla [5b]

Uznávám, že řešení tohoto problému jsem už viděl a to v knížce **Průvodce albyrintem algoritmů**, ale protože jsem při čtení zadání problém hned spojil s tím co jsem četl, pokládám to za něco, co už je součástí mých znalostí.

Začneme tedy hledáním matice \mathbf{Q} , které při vynásobení s maticí $\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix}$ získáme matici $\begin{pmatrix} F_{n+1} \\ F_n + F_{n+1} \end{pmatrix}$. Odpověď je hned jasná, dokud si pamatujeme pravidla násobení matic:

$$\mathbf{Q} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Ted' zbývá za úkol jen provést násobení v lépe než lineárním čase. K tomu využijeme vlastnost asociativity u matic a chytrý rekursivní algoritmus, který dokáže umocnit čísla v čase $\mathcal{O}(\log n)$. Nejprve si ukážeme, že matici $\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix}$ můžeme napsat jako $\mathbf{Q}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$. Můžeme si všimnout, že pro $n = 1$ už víme, že $\mathbf{Q}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$ nám dá $\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$. Dále také víme, že abychom získali F_3 , musíme tuto matici vynásobit zleva maticí \mathbf{Q} , tedy: $\mathbf{Q}\mathbf{Q} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} F_2 \\ F_3 \end{pmatrix}$. A kvůli vlastnosti asociativity, toto můžeme zapsat jako: $\mathbf{Q}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} F_2 \\ F_3 \end{pmatrix}$. Tím jsme dokázali, že $\mathbf{Q}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix}$.

Ted' stačí jen vysvětlit trik rychlé umocňování.

Nejdříve uvažme: $x \in \mathbb{R}$, kterou umocníme číslem $n \in \mathbb{N}$. Víme, že $x^{2k} = (x^2)^k$ a $x^{k+1} = x \cdot (x^k)$, takže přepíšeme sudé n jako $2k$ a liché jako $2k + 1$. Takhle můžeme neustále redukovat exponent, dokud nezískáme 0 a vrátíme výsledek umocnění x^0 , tedy 1. Takhle vypadá algoritmus v pseudokódu:

Algorithm 1 FastExponent

Vstup: $x \in \mathbb{R}, n \in \mathbb{N}$

```
if  $n = 0$  then
    return 1
end if
 $a \leftarrow \text{FastExponent}(x, \lfloor n/2 \rfloor)$ 
if  $n$  is even then
    return  $a \cdot a$ 
else
    return  $a \cdot a \cdot x$ 
end if
```

Výstup: x^n

Takhle provedeme jen nějakých $\mathcal{O}(\log n)$ operací místo $\mathcal{O}(N)$.

Úkol 3 – Dosažitelnost [5b]

K tomuto úkolu můžeme využít algoritmus z předchozího cvičení. Umocnění matici cest provedeme přes algoritmus *FastExponent*. Vzhledem k tomu, že zadání říká, abychom se vyhnuly počítání obrovských čísel, nemůžeme jen tak upravit výslednou matici. Tudíž musíme jinak.

Protože nás zajímá jen jestli existuje nějaká cesta a ne kolik jich je, můžeme přidat jednu *for* cyklus, kde každý index matice, který je větší než jedna, změníme na jedničku. Takhle potom bude největší číslo v matici m , tedy počet vrcholů v graphu. Samozřejmě toto není perfektní řešení, protože pro každé násobení projedeme celou matici. Tedy provedeme $\mathcal{O}(m^2 \cdot \log n)$. Toto ze začátku vypadá dost nevýhodně, ale to je také protože jsme zanedbali další důležitou změnu v algoritmu a to je, že násobíme matice, ne čísla. Násobení matice je m^3 operací, tudíž můžeme zanedbat m^2 jako člen který roste pomaleji. Tudíž bychom skončili s $\mathcal{O}(m^3 \cdot \log n)$, což je stejné jako normální umocnění matice a skončili bychom s maticí, kde bude 1, jestli existuje cesta z vrcholu za n kroků.

Úkol 4 – Obecná inverze [3b]

Je zřejmé, že zadání naznačuje, abychom nejdříve představili zobrazení matici \mathbf{A} a z toho vyvodit její inverzi. Ale vzhledem k tomu, že matice \mathbf{A} má velikost 2×2 , je mnohem rychlejší vypočítat inverzní matici \mathbf{A}^{-1} přes determinant a pochopit, proč to tak je.

$$\mathbf{A} = \begin{pmatrix} c & -c \\ c & c \end{pmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{\mathbf{det}|\mathbf{A}|} \begin{pmatrix} c & c \\ -c & c \end{pmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{2c^2} \begin{pmatrix} c & c \\ -c & c \end{pmatrix}$$

$$\mathbf{A}^{-1} = \begin{pmatrix} \frac{1}{2c} & \frac{1}{2c} \\ \frac{-1}{2c} & \frac{1}{2c} \end{pmatrix}$$

Ted' když vidíme původní matici \mathbf{A} a její inverze \mathbf{A}^{-1} , je už zřejmé, jak bychom mohli najít inverzi bez typického výpočtu. Můžeme nahlédnout, že původní matice \mathbf{A} je dost podobná k otočení, akorát že místo nul má další c , tudíž pro její inverzi je potřeba se těchto c zbavit. To zvládneme právě pomocí dělení extra c . Proto je inverzní matice plná $\frac{1}{2c}$. Změna pozice mínusu je také zřejmé.