NONOGRAM MAKER

Maturitní práce z předmětu Informatika, Tomáš Vrána, 2018/2019

Vedoucím práce je Emil Miler

Obsah

Zadání práce

Vytvořte program, který z fotografie vygeneruje nonogram v černobílé barvě. Součástí aplikace bude nastavení, kde si uživatel nastaví velikost (a tedy i detail) nonogramu.

Prohlašuji, že jsem jediným autorem této maturitní práce a všechny citace, použitá literatura a další zdroje jsou v práci uvedené.

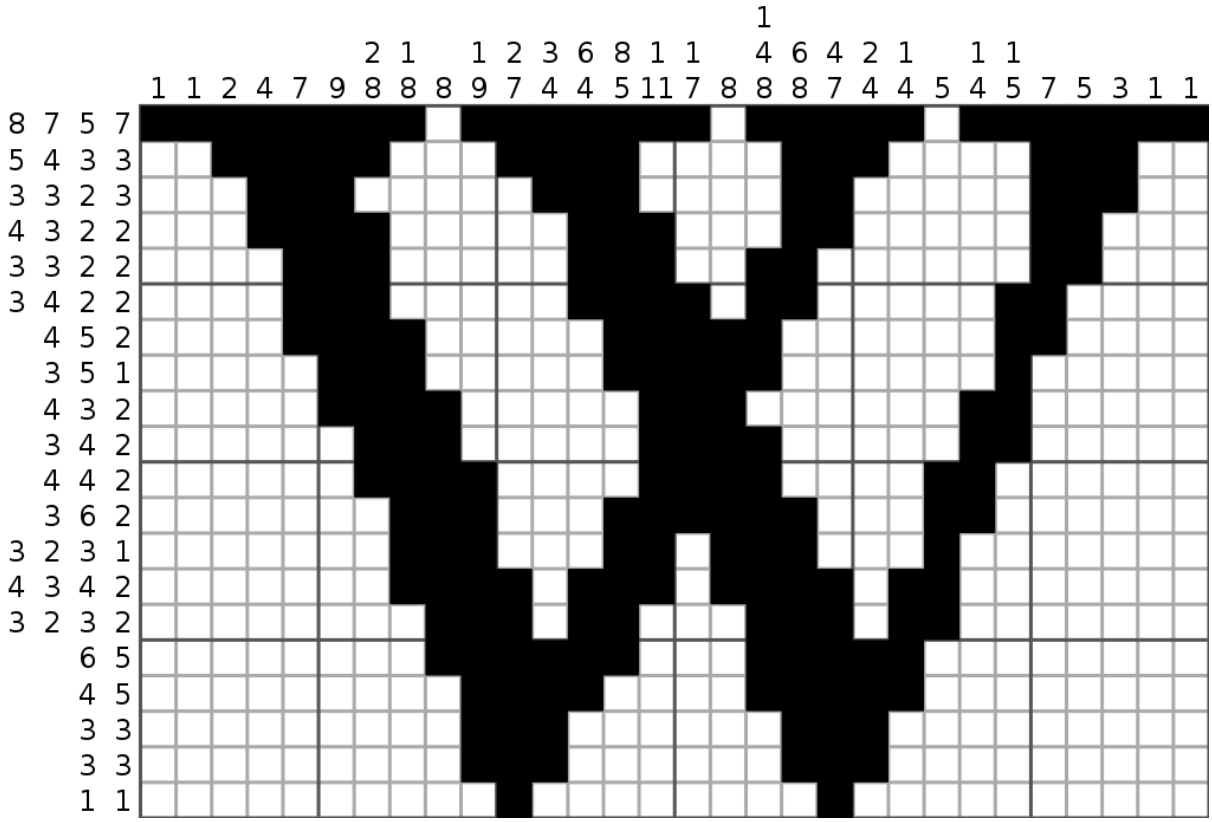
Tímto dle zákona 121/2000 Sb.ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium Jana Keplera, Praha 6, Parléřova 2 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Tomáš Vrána

V této Maturitní práci jsem si dal za cíl vytvořit program, který dokáže z vloženého obrazového materiálu vytvořit černobílý rébus, mající původ v Japonsku, zvaný nonogram.

A co to ten nonogram vlastně je?

> *"Malovaná křížovka (nonogram, někdy také Zakódovaný obrázek) je logický hlavolam, při kterém je okolo mřížky umístěná legenda s čísly a pomocí nich lze získat obrázek. Každé číslo v legendě určuje počet za sebou následujících čtverečků stejné barvy."*[1]



Jednoduše řečeno, cílem řešitele je pomocí horizontálních a vertikálních nápověd vyluštit pozici černých pixelů, které tvoří nějaký obrazec. Každé číslo reprezentuje blok po sobě jdoucích černých pixelů (jejich délku udává dané číslo), které se bezprostředně po sobě nacházejí v dané řádce/ daném sloupci. Mezi více bloky na řádku/ sloupci musí být minimálně jeden bílý pixel.

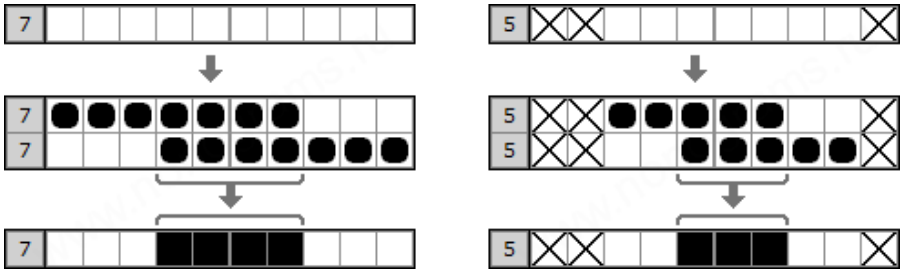
[1] Zdroj úryvku v kurzívě a obrázku:
(https://cs.wikipedia.org/wiki/Malovan%C3%A1_k%C5%99%C3%AD%C5%BEovka)

Metody pro řešení nonogramu[2]

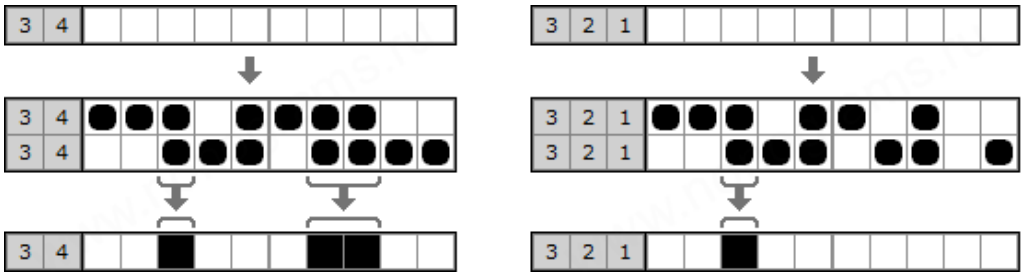
K řešení nonogramu je jako u většiny podobných hlavolamů zapotřebí kromě nějakých postupů využít i selský rozum. Jednotlivé metody je však dobré znát a mít při řešení na paměti.

1. Superpozice extrémních pozic

    - Pokud je v daném řádku / sloupci pouze jedna nápověda, jejíž hodnota je větší než příslušný rozměr výsledného obrázku, tak i když blok umístíme zcela doleva, nebo zcela doprava; vždy zbydou uprostřed pixely, které budou vždy černé; můžeme je tedy začernit.

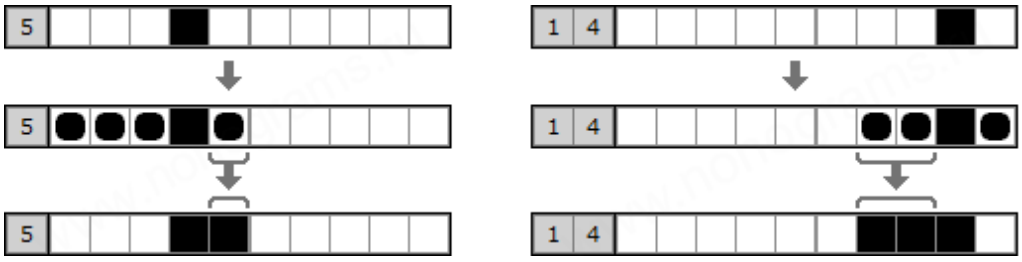


    - Pokud je v řádku/ sloupci více nápověd, můžeme zkusit podobnou metodu; bloky s jednou mezerou dáme zcela doprava a zcela doleva. Ne vždy, ale může nastat situace, že některé pixely budou začerněny pořád; můžeme je tedy začernit



2. Vyplnění z kraje

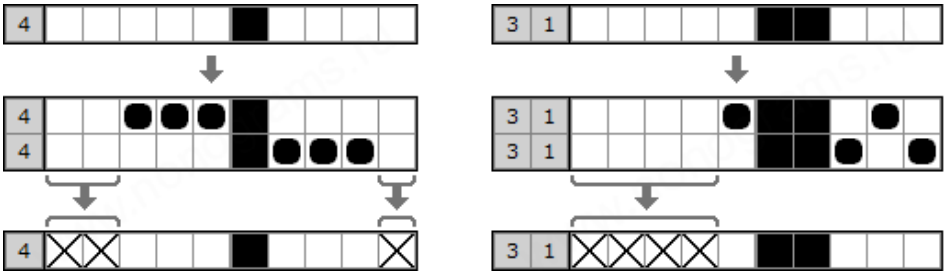
    - Pokud je v řádce/ sloupci již vybarvený pixel, můžeme blok dát co nejvíc doprava a doleva tak, aby již existující pixel byl součást obrázku. Ne vždy k této situaci dojde, ale podobně jako v 1.) může dojít k překryvu; tyto pixely můžeme vybarvit



[2] Jelikož neexistuje nějaké jednotné značení, jednotlivé metody jsem roztřídil a rozlišil podle stránek: https://www.nonograms.org/methods Ze stejné stránky jsem použil také obrázky k této kapitole.

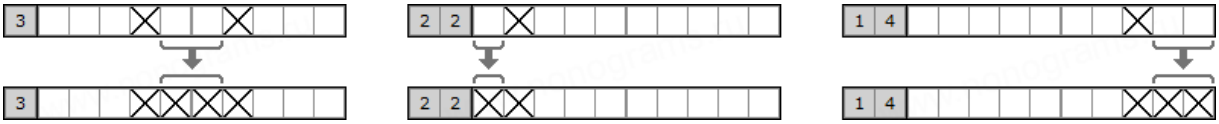
3. Nepřístupnost

   - Tato metoda je vlastně komplementární k obou předchozím. Pokud bloky budeme posouvat do jejich krajních pozic, může nastat situace, že některé pixely nebudou zabarveny nikdy. S těmito pixely se při řešení nonogramu také pracuje; daný čtvereček se proškrtne.



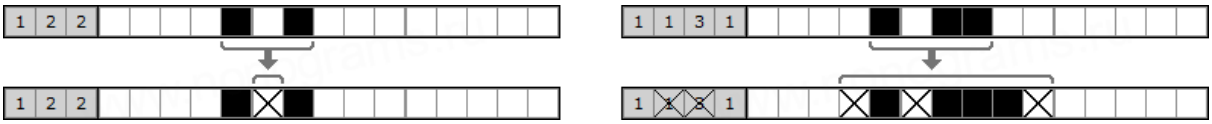
4. „Does not fit“

   - Může nastat situace, že nám dva „křížky“ ohraničují počet pixelů, do kterého se žádná nápověda již nevejde. Poté můžeme zaškrtat i tyto pixely.



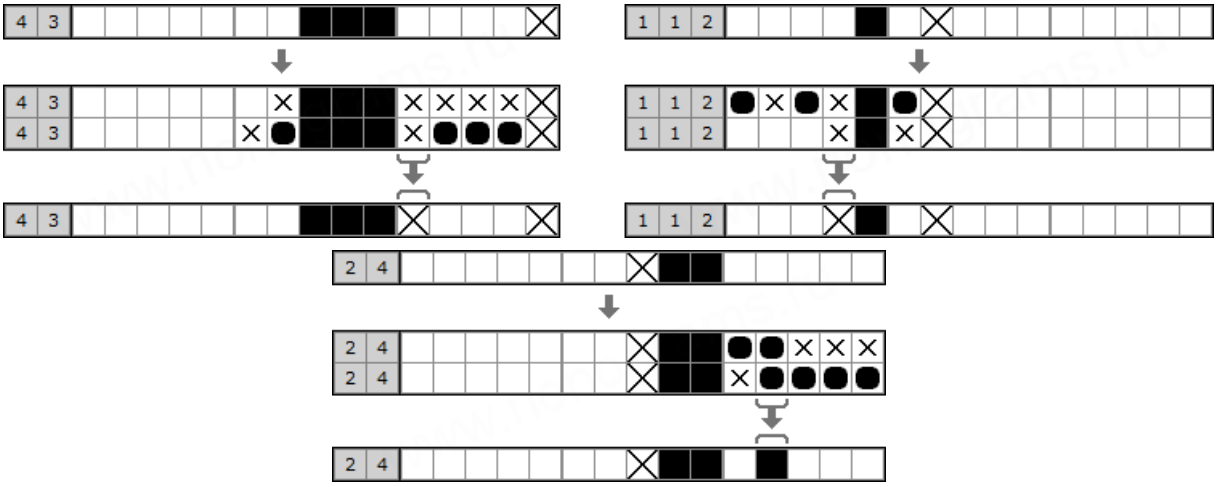
5. Oddělení

   - Pokud jsou v řádku/ sloupci již některé pixely vybarvené (s jednou mezerou mezi sebou), je potřeba se podívat, jestli jejich propojení odpovídá nějaké nápovědě. Jestli ne, tak mezeru můžeme proškrtnout.



6. Dvojité umístění

   - Pokud nastane situace, kdy budou existovat jen dvě možné varianty umístění bloků; je dobré obě dvě vyzkoušet a zaznačit si jejich případný překryv, příp. zajisté nezabarvené pixely.



Použité programy, knihovny

K programování jsem použil programovací jazyk Python (ve verzi 3.7) a k samotnému psaní kódu PyCharm, který mě zaujal svou grafickou přívětivostí a také přímým napojením na webovou službu GitHub, kterou jsem používal k verzování projektu.

Ze standartních knihoven pythonu jsem při tvorbě využil:

- OS (konkrétně OS.path)
    - K práci se jmény souborů a adresářů.

Kromě standartních knihoven pythonu jsem využil:

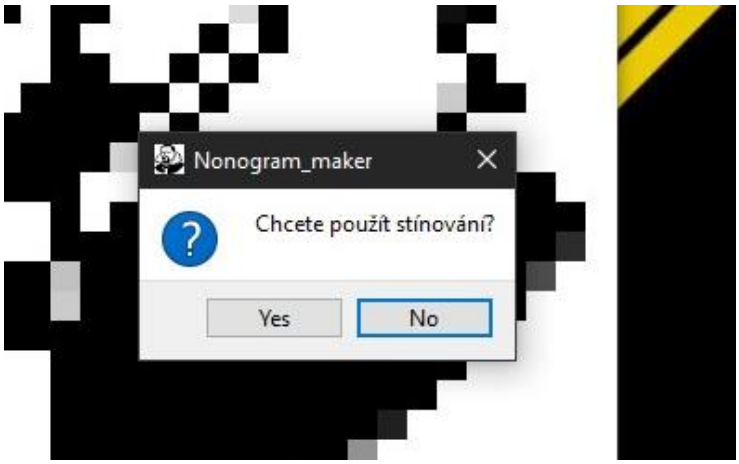
- Pillow
    - K práci s obrazovým materiálem.
- PyQt5
    - Pro vytvoření grafického rozhraní
- Numpy
    - Kvůli lepší práci se seznamy.

K následné konverzi kódu do samostatně spustitelné formy (.exe) jsem použil nástroj pyinstaller.

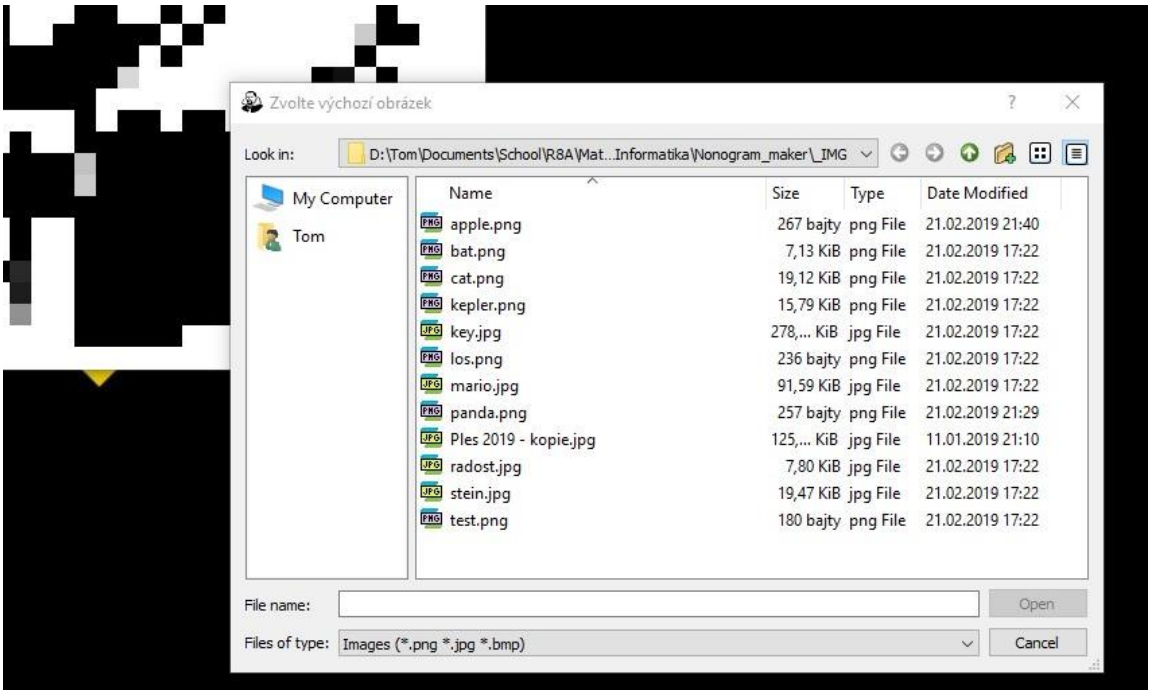
Běh programu

Program pracuje poměrně jednoduše; nejdříve se program uživatele zeptá, jestli „chce využít stínování“.

Jedná se o použití argumentu „Dither“ v knihovně Pillow, který určuje to, jak je obrázek přetransformovaný do dvou barev – černé a bílé.



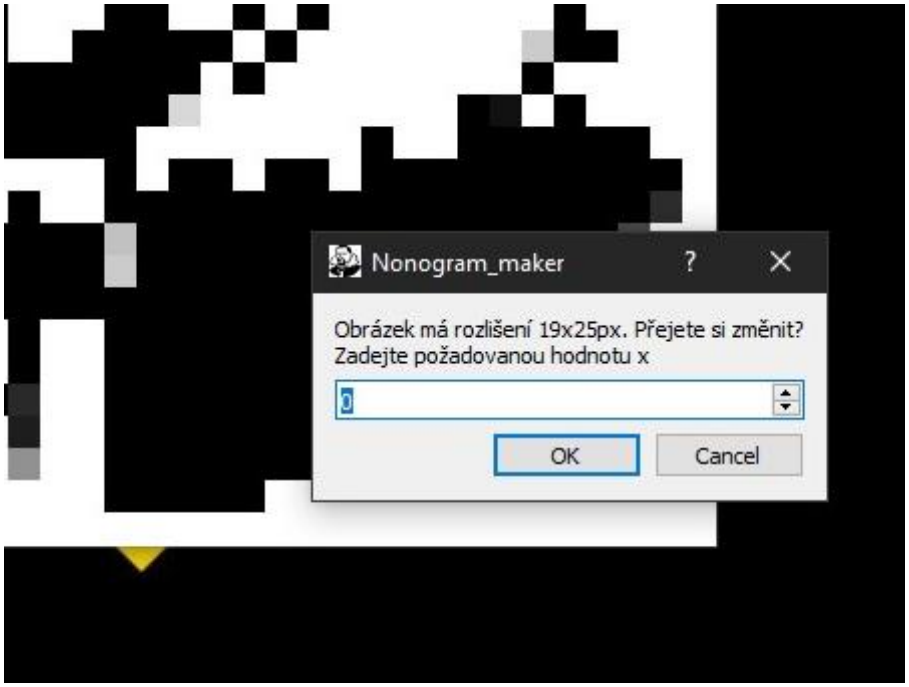
Poté uživatele vyzve k vybrání obrazového podkladu (ve formátech .png .jpg a .bmp). Program si při svém spuštění vytvoří složku _IMG, která se nachází ve stejné složce jako program samotný, avšak obrázek lze vybrat zcela kdekoli na disku počítače.



Poté program zjistí rozměry obrázku a zeptá se uživatele, jestli je s rozměry spokojený, nebo je chce změnit.

Pokud uživatel chce rozměry změnit, do dialogového okna zadá požadovanou hodnotu horizontálního rozměru obrázku a program obrázek upraví ve správném poměru stran.

Pokud je uživatel s velikostí obrázku spokojený, může buď v poli ponechat nulu a potvrdit stiskem „OK“, nebo operaci změny velikosti zrušit tlačítkem „Cancel“

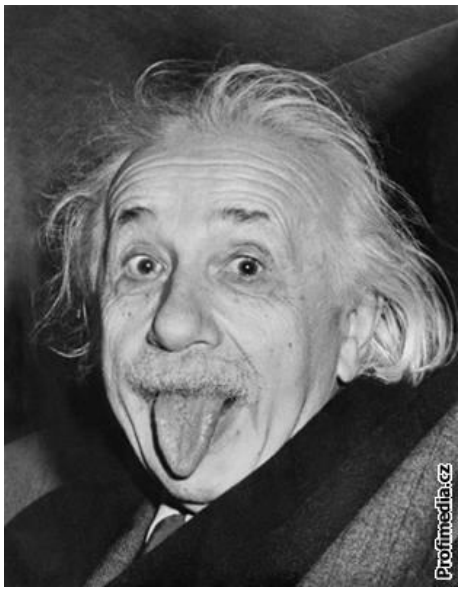


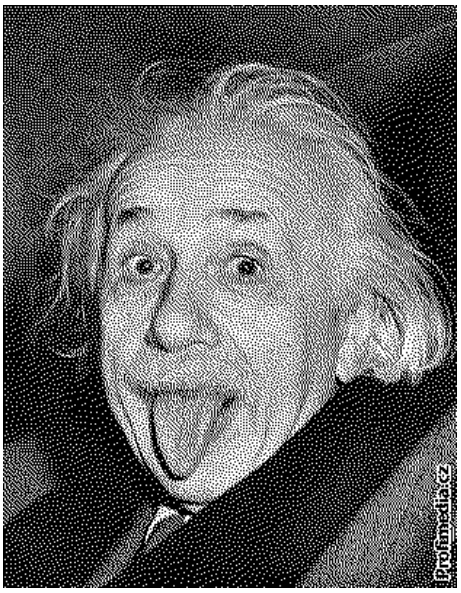
Následně se obrázek převede do černé a bílé barvy. Vstupní obrázek se vždy převede do odstínů šedé, avšak další postup se odvíjí od toho, jestli uživatel povolí „stínování“ (viz. výše), nebo ne.

- Pokud uživatel zvolí, že chce „stínování“, při konvertování obrázku se využije Floyd–Steinbergova metoda, *která slouží ke změně barevné charakteristiky obrazu při zachování původní vizuální informace v co největší míře*[3].
- Pokud uživatel zvolí, že nechce „stínování“, program vyhodnotí u každého bodu, jestli hodnota jedné z barevných složek je menší než 128 → přemění se na černý pixel, a naopak. Tuto „metodu“ jsem pracovně nazval „Metoda jednoduché separace“

Na obrázcích níže můžete vidět využití obou metod. Je poměrně očividné, že obě metody mají svá uplatnění. Floyd-Steinbergova metoda je obecně vhodnější pro všechny složitější obrázky; metoda jednoduché separace však výsledným obrázkům dodává více „nonogramový vzhled“.

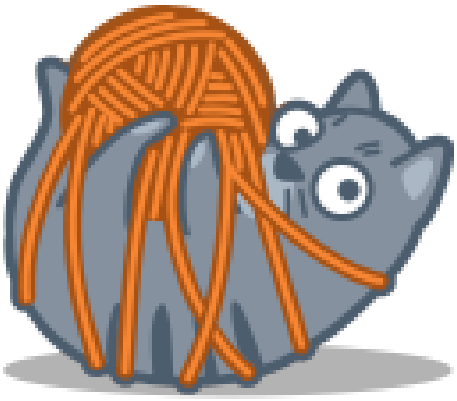
[3] Použito z: https://cs.wikipedia.org/wiki/Floydova%E2%80%93Steinbergova_metoda

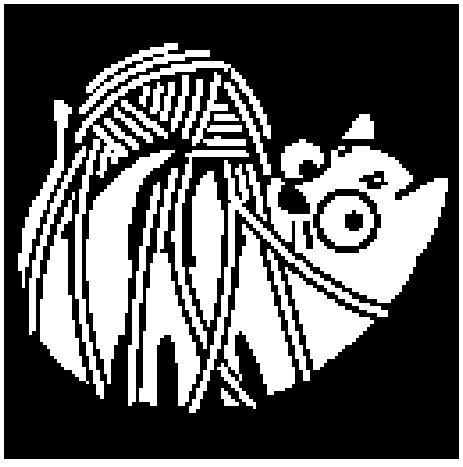




Floyd–Steinbergova metoda

Metoda jednoduché separace

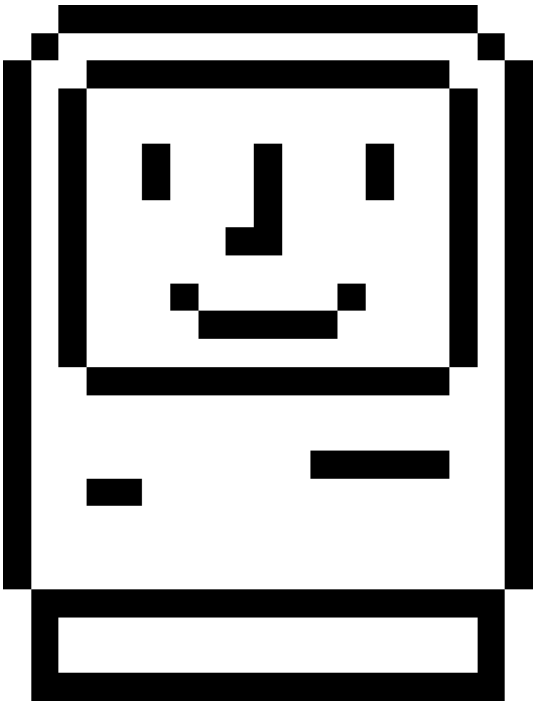


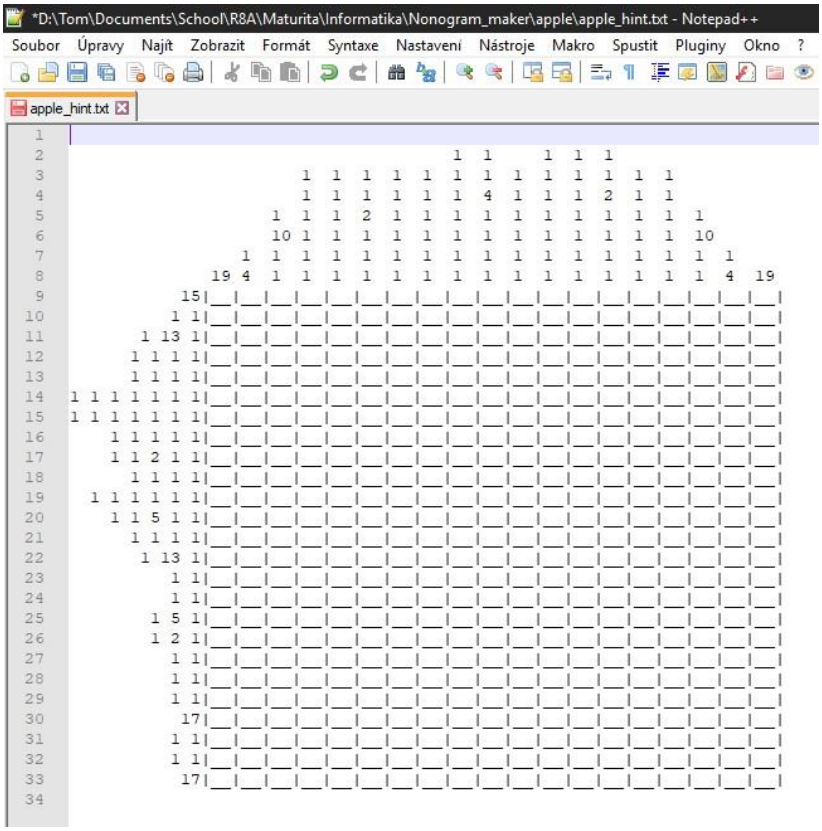


Metoda jednoduché separace

Takto vzniklá obrazová data jsou přetransformována do nested listů a následně jsou z těchto listů vytvořeny horizontální a vertikální "nápovědy" nonogramu.

Tyto nápovědy jsou po určitém formátování spolu s černobílým obrázkem uloženy do nově vytvořené složky (ve stejném umístění jako spuštěný program se vytvoří složka _NONOGRAM, ve které se vždy vytvoří složka s názvem obrázku).





Instalace programu

- Pro běh .exe souboru (běží pouze na OS Windows) vám stačí si stáhnout tento soubor na GitHubu, ze kterého lze po extrahování rovnou spustit „Nonogram_maker.exe“

- Pro případný běh kódu v pythonu je potřeba mít stažený:
    - Python (odzkoušeno na pythonu3.7)
    - Knihovnu Pillow
    - Knihovnu PyQt5
    - Knihovnu Numpy
    - A samozřejmě i samotný kód

  Všechny zmíněné knihovny se dají nainstalovat pomocí nástroje pip

  Po instalaci všeho potřebného a extrahování souboru z GitHubu již lze pomocí pythonu spustit soubor „Nonogram_maker.pyw“

V obou případech je na počítači potřeba mít nainstalovaný libovolný textový editor (mě se osvědčil Notepad++) a libovolný prohlížeč fotografií, či obrázkový editor. Samotný program lze bez těchto aplikací spustit, avšak jsou potřeba pro zobrazení vytvořených souborů nonogramu.

Závěr

Jelikož se jedná o můj první počin v tomto směru vůbec, tak se tvorba neobešla bez mnohého vyhledávání informací a četného testování, za kterém ve velké části případů stála nějaká poměrně malá chyba.

Myslím si, že i přesto program splňuje zadaná kritéria, avšak jak na samotném kódu, tak na GUI celého programu by se dalo ještě zapracovat…

Samotná tvorba programu mi však přinesla mnoho zajímavých poznatků a zkušeností, které se mi zajisté budou hodit k mému budoucímu studiu na fakultě informatiky.