



# KOMPRES

Adam Klepáč

14. listopadu 2024

# OBSAH

Bezztrátová komprese

Bezztrátové formáty

Ztrátová komprese

# PROČ KOMPRIMOVAT?

- Menší velikost souboru.



# PROČ KOMPRIMOVAT?

- Menší velikost souboru.
- Často je možné jinou reprezentací souboru v paměti zmenšit jeho velikost i bez ztráty dat – tzv. **bezztrátová komprese**.

# PROČ KOMPRIMOVAT?

- Menší velikost souboru.
- Často je možné jinou reprezentací souboru v paměti zmenšit jeho velikost i bez ztráty dat – tzv. **bezztrátová komprese**.
- Druhá možnost je **ztrátová komprese**, kdy jsou některé části souboru trochu změněny pro usnadnění komprese za cenu ztráty přesnosti.

# PROČ KOMPRIMOVAT?

- Menší velikost souboru.
- Často je možné jinou reprezentací souboru v paměti zmenšit jeho velikost i bez ztráty dat – tzv. **bezztrátová komprese**.
- Druhá možnost je **ztrátová komprese**, kdy jsou některé části souboru trochu změněny pro usnadnění komprese za cenu ztráty přesnosti.
- Můžeme uložit mnohem více souborů, když jsou komprimované.

# PROČ KOMPRIMOVAT?

- Menší velikost souboru.
- Často je možné jinou reprezentací souboru v paměti zmenšit jeho velikost i bez ztráty dat – tzv. **bezztrátová komprese**.
- Druhá možnost je **ztrátová komprese**, kdy jsou některé části souboru trochu změněny pro usnadnění komprese za cenu ztráty přesnosti.
- Můžeme uložit mnohem více souborů, když jsou komprimované.
- Komprimované soubory se též rychleji přenášejí mezi počítači.

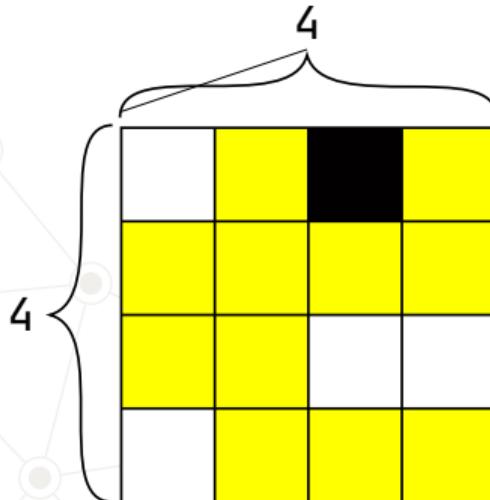
# PROČ KOMPRIMOVAT?

- Menší velikost souboru.
- Často je možné jinou reprezentací souboru v paměti zmenšit jeho velikost i bez ztráty dat – tzv. **bezztrátová komprese**.
- Druhá možnost je **ztrátová komprese**, kdy jsou některé části souboru trochu změněny pro usnadnění komprese za cenu ztráty přesnosti.
- Můžeme uložit mnohem více souborů, když jsou komprimované.
- Komprimované soubory se též rychleji přenášejí mezi počítači.
- Primárně chceme komprimovat zvuk, obrázky a video.

# BEZZTRÁTOVÁ KOMPRESE

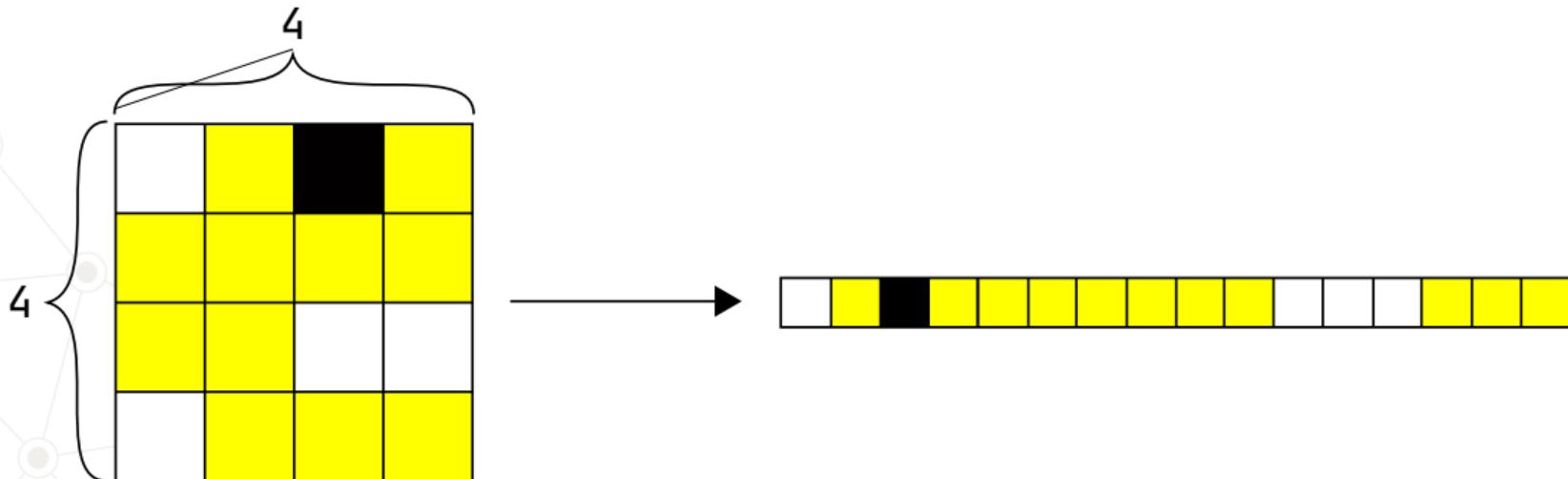
## PŘÍKLAD – OBRÁZEK

Obrázky jsou uloženy na počítači (kromě metadat) jako posloupnosti pixelů – trojic (R,G,B).



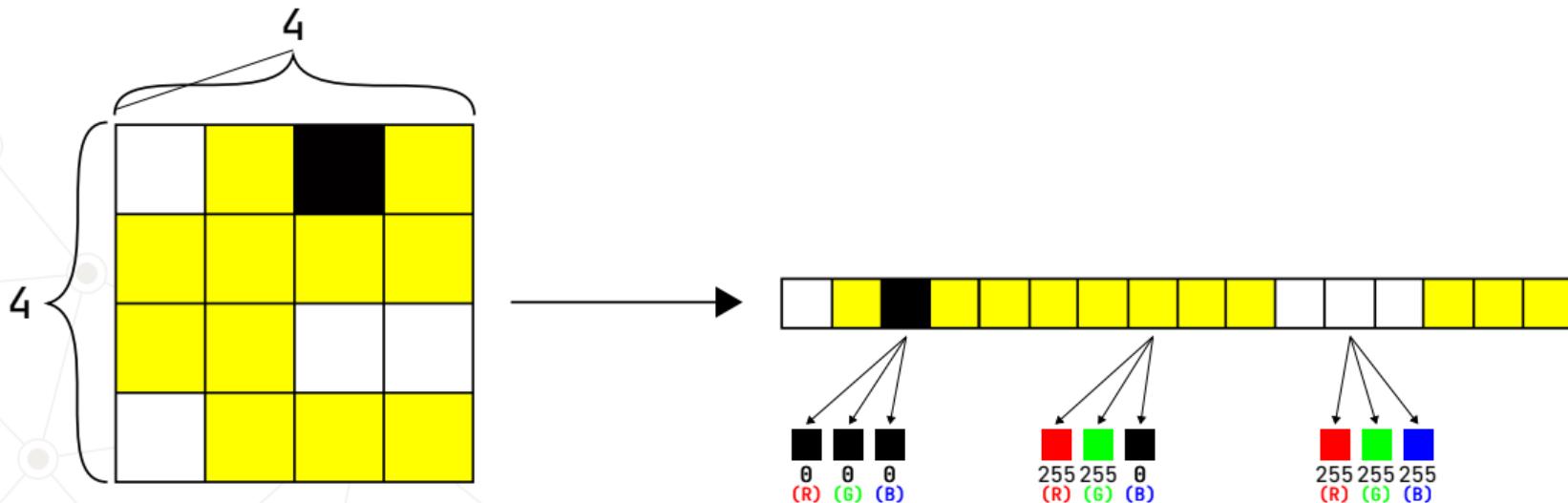
## PŘÍKLAD – OBRÁZEK

Obrázky jsou uloženy na počítači (kromě metadat) jako posloupnosti pixelů – trojic (R,G,B).



## PŘÍKLAD – OBRÁZEK

Obrázky jsou uloženy na počítači (kromě metadat) jako posloupnosti pixelů – trojic (R,G,B).



## RUN-LENGTH KOMPRESSE

Tato posloupnost zabírá  $16 \cdot 3 = 48\text{ B}$ , protože každý pixel zabírá 3 B, jeden byte pro každou barevnou složku.



## RUN-LENGTH KOMPRESE

Tato posloupnost zabírá  $16 \cdot 3 = 48\text{ B}$ , protože každý pixel zabírá 3 B, jeden byte pro každou barevnou složku.



Jeden způsob komprese (tzv. **run-length** komprese) využívá toho, že se některé pixely po sobě opakují.

## RUN-LENGTH KOMPRESE

Tato posloupnost zabírá  $16 \cdot 3 = 48\text{ B}$ , protože každý pixel zabírá 3 B, jeden byte pro každou barevnou složku.



Jeden způsob komprese (tzv. **run-length** komprese) využívá toho, že se některé pixely po sobě opakují. Není proto třeba si pamatovat každý pixel zvlášť, ale jenom jeden + záznam o tom, kolik takových jde za sebou.

# RUN-LENGTH KOMPRESE

Na obrázku



si stačí například místo těch 7 žlutých pixelů v řadě pamatovat jenom číslo 7 a jeden žlutý pixel – třeba takhle:



# RUN-LENGTH KOMPRESSE

Na obrázku



si stačí například místo těch 7 žlutých pixelů v řadě pamatovat jenom číslo 7 a jeden žlutý pixel – třeba takhle:



Tak jednoduché to ale není. Jak má počítač poznat, kdy to, co čte je pixel a kdy číslo?

# RUN-LENGTH KOMPRESE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje.

## RUN-LENGTH KOMPRESE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje. Z obrázku



se tímto postupem nakonec stane

## RUN-LENGTH KOMPRESSE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje. Z obrázku



se tímto postupem nakonec stane



## RUN-LENGTH KOMPRESE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje. Z obrázku



se tímto postupem nakonec stane

1	1	1	7	3	3
---	---	---	---	---	---

Poslední otázka: Kolik bytů rezervujeme pro ukládání počtu následujících pixelů?

## RUN-LENGTH KOMPRESE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje. Z obrázku



se tímto postupem nakonec stane

1	1	1	1	7	3	3	
---	---	---	---	---	---	---	--

Poslední otázka: Kolik bytů rezervujeme pro ukládání počtu následujících pixelů?

Nejpřímější možnost je volit tolik bytů, aby mohlo uložené číslo být větší než celkový počet pixelů v obrázku, tj. šířka x výška – v tomto případě bohatě stačí 1 B.

## RUN-LENGTH KOMPRESE

Jediným rozumným řešením je před každou posloupnost stejných pixelů napsat číslo, které říká, kolik jich následuje. Z obrázku



se tímto postupem nakonec stane

1	1	1	1	7	3	3	
---	---	---	---	---	---	---	--

Poslední otázka: Kolik bytů rezervujeme pro ukládání počtu následujících pixelů?

Nejpřímější možnost je volit tolik bytů, aby mohlo uložené číslo být větší než celkový počet pixelů v obrázku, tj. šířka x výška – v tomto případě bohatě stačí 1 B.

Použitím **run-length** komprese můžeme snížit velikost tohoto obrázku na 24 B – polovinu původní velikosti – **bez ztráty dat**.

## HUFFMANŮV STROM (STROM „PRIORIT“)

Dalším oblíbeným způsobem komprese je tzv. **Huffmanův strom**.



# HUFFMANŮV STROM (STROM „PRIORIT“)

Dalším oblíbeným způsobem komprese je tzv. **Huffmanův strom**.  
Jeho princip spočívá v

- Rozdělení obrázku na bloky.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Dalším oblíbeným způsobem komprese je tzv. **Huffmanův strom**.  
Jeho princip spočívá v

- Rozdělení obrázku na bloky.
- Spočítání četností bloků v obrázku.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Dalším oblíbeným způsobem komprese je tzv. **Huffmanův strom**.

Jeho princip spočívá v

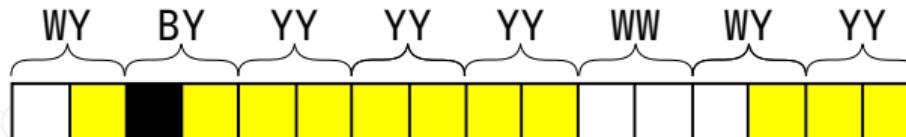
- Rozdělení obrázku na bloky.
- Spočítání četností bloků v obrázku.
- Přiřazení číselných kódů blokům, tak aby **nejčetnější bloky dostaly nejkratší kódy**.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Zvolíme-li (pro jednoduchost) rozdělení obrázku

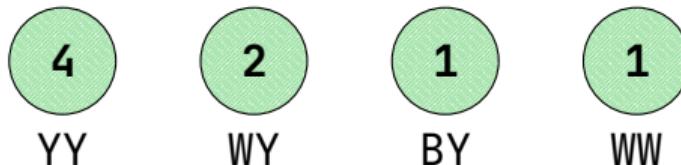


na bloky po  $2B$ , dostaneme následující 4 typy bloků:



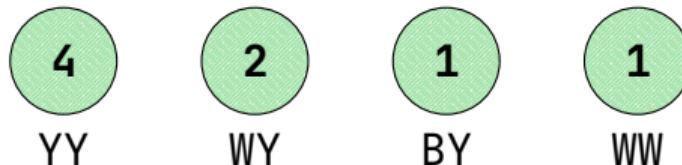
# HUFFMANŮV STROM (STROM „PRIORIT“)

Spočítáme četnosti každého bloku.

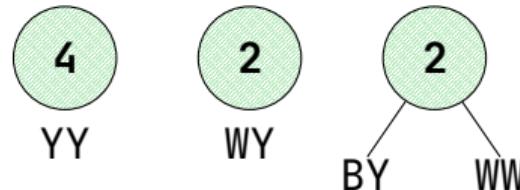


# HUFFMANŮV STROM (STROM „PRIORIT“)

Spočítáme četnosti každého bloku.

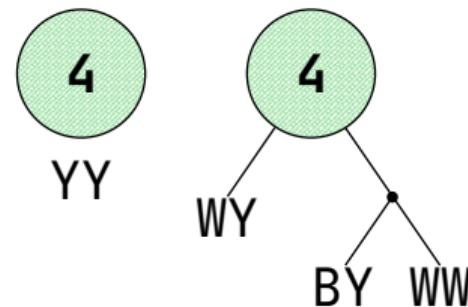


Z dvou bloků s nejmenší četností vytvoříme binární strom, u kterého si zapamatujeme součet obou četností.



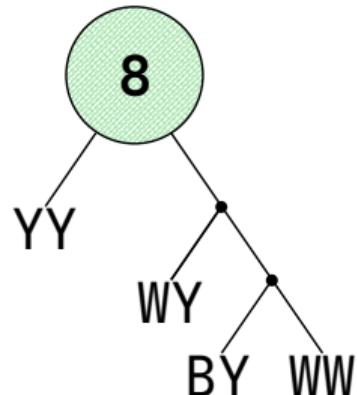
# HUFFMANŮV STROM (STROM „PRIORIT“)

Pokračujeme dále se spojováním stromů (či bloků) s nejmenšími četnostmi, dokud nedostaneme jediný strom.



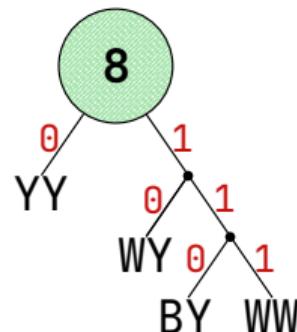
# HUFFMANŮV STROM (STROM „PRIORIT“)

Pokračujeme dále se spojováním stromů (či bloků) s nejmenšími četnostmi, dokud nedostaneme jediný strom.



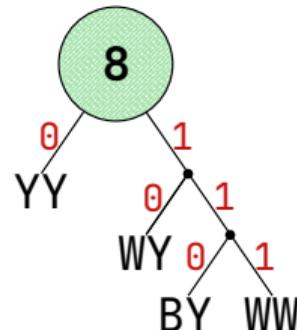
## HUFFMANŮV STROM (STROM „PRIORIT“)

Nakonec každému úseku přiřadíme 0 nebo 1 podle toho, jestli je vpravo nebo vlevo od posledního.



# HUFFMANŮV STROM (STROM „PRIORIT“)

Nakonec každému úseku přiřadíme 0 nebo 1 podle toho, jestli je vpravo nebo vlevo od posledního.



Tím dostaneme následující kódování

$$\begin{array}{ll} YY \rightarrow 0 & WY \rightarrow 10 \\ BY \rightarrow 110 & WW \rightarrow 111 \end{array}$$

# HUFFMANŮV STROM (STROM „PRIORIT“)

Huffmanův strom má tři důležité vlastnosti:

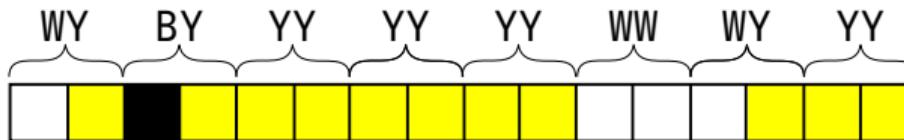
- Každý blok má jiný kód.
- Čím častější je blok, tím kratší má kód.
- Každý kód má na začátku počet jedniček odpovídající tomu, jak hluboko je ve stromě.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Huffmanův strom má tři důležité vlastnosti:

- Každý blok má jiný kód.
- Čím častější je blok, tím kratší má kód.
- Každý kód má na začátku počet jedniček odpovídající tomu, jak hluboko je ve stromě.

Posloupnost pixelů

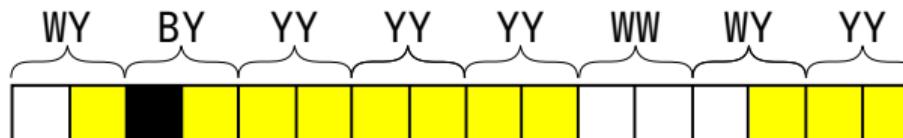


# HUFFMANŮV STROM (STROM „PRIORIT“)

Huffmanův strom má tři důležité vlastnosti:

- Každý blok má jiný kód.
- Čím častější je blok, tím kratší má kód.
- Každý kód má na začátku počet jedniček odpovídající tomu, jak hluboko je ve stromě.

Posloupnost pixelů



bude zakódována takhle:

10	110	0	0	0	111	10	0
----	-----	---	---	---	-----	----	---

# HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!



## HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!

Původní uložení obrázku mělo 48 bytů. To je asi 27krát méně.



## HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!

Původní uložení obrázku mělo 48 byte. To je asi 27krát méně.

Kde je háček? Právě se nám podařilo bezztrátově 27krát zmenšit velikost obrázku.



## HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!

Původní uložení obrázku mělo 48 byte. To je asi 27krát méně.

Kde je háček? Právě se nám podařilo bezztrátově 27krát zmenšit velikost obrázku.

Přece, aby počítač uměl správně přečíst vyrobený kód, musí vědět, jaká data jsou uložena pod každým číslem.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!

Původní uložení obrázku mělo 48 byte. To je asi 27krát méně.

Kde je háček? Právě se nám podařilo bezztrátově 27krát zmenšit velikost obrázku.

Přece, aby počítač uměl správně přečíst vyrobený kód, musí vědět, jaká data jsou uložena pod každým číslem.

V našem případě to znamená, že musíme na začátku obrázku vypsat **celý Huffmanův strom**, třeba v podobě „kód,blok,kód,blok,...“.

# HUFFMANŮV STROM (STROM „PRIORIT“)

Posloupnost čísel 10110000111100 má pouze **14 bitů**!

Původní uložení obrázku mělo 48 byte. To je asi 27krát méně.

Kde je háček? Právě se nám podařilo bezztrátově 27krát zmenšit velikost obrázku.

Přece, aby počítač uměl správně přečíst vyrobený kód, musí vědět, jaká data jsou uložena pod každým číslem.

V našem případě to znamená, že musíme na začátku obrázku vypsat **celý Huffmanův strom**, třeba v podobě „kód,blok,kód,blok,...“.

Celý obrázek bude proto uložen takto a zabírat  $4 \cdot 6 + 4 + 2 = 30 B$ .

0				10			110					1011 0000	1111 00
---	--	--	--	----	--	--	-----	--	--	--	--	--------------	------------

1

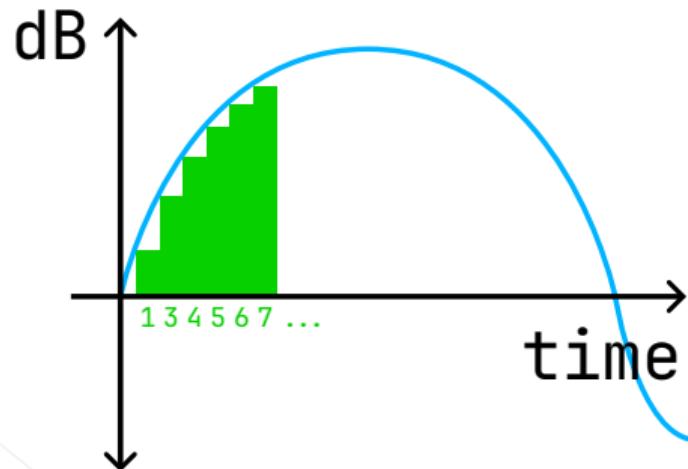
## BEZZRÁTOVÉ FORMÁTY

# AUDIO

Zvuk je na počítači uložen v podobě posloupnosti čísel reprezentujících **amplitudu** (víceméně „hlasitost“, v dB) **v okamžik měření**.

# AUDIO

Zvuk je na počítači uložen v podobě posloupnosti čísel reprezentujících **amplitudu** (víceméně „hlasitost“, v dB) **v okamžik měření**.



# AUDIO

Hlavička každého audio souboru obsahuje přinejmenším dva údaje:

# AUDIO

Hlavička každého audio souboru obsahuje přinejmenším dva údaje:

- **sample rate** – frekvence, se kterou počítač měří amplitudu. Vlastně kolik čísel (těch zelených obdélníků) má uložených pro každou vteřinu záznamu.

# AUDIO

Hlavička každého audio souboru obsahuje přinejmenším dva údaje:

- **sample rate** – frekvence, se kterou počítač měří amplitudu. Vlastně kolik čísel (těch zelených obdélníků) má uložených pro každou vteřinu záznamu.
  - Za standardní sample rate dvoukanálového audia se považuje většinou 44100 Hz nebo 48000 Hz. To je dvakrát (kvůli dvěma kanálům) vyšší frekvence zvuku, než je průměrný člověk schopen slyšet. Počítač tedy ukládá tento počet čísel pro každou vteřinu zvuku.

# AUDIO

Hlavička každého audio souboru obsahuje přinejmenším dva údaje:

- **sample rate** – frekvence, se kterou počítač měří amplitudu. Vlastně kolik čísel (těch zelených obdélníků) má uložených pro každou vteřinu záznamu.
  - Za standardní sample rate dvoukanálového audia se považuje většinou 44100 Hz nebo 48000 Hz. To je dvakrát (kvůli dvěma kanálům) vyšší frekvence zvuku, než je průměrný člověk schopen slyšet. Počítač tedy ukládá tento počet čísel pro každou vteřinu zvuku.
- **bit depth** – počet bitů, který je určen pro uložení výšky amplitudy. Vlastně kolik mezistupňů existuje mezi nejvyšší a nejnižší měřitelnou amplitudou.

# AUDIO

Hlavička každého audio souboru obsahuje přinejmenším dva údaje:

- **sample rate** – frekvence, se kterou počítač měří amplitudu. Vlastně kolik čísel (těch zelených obdélníků) má uložených pro každou vteřinu záznamu.
  - Za standardní sample rate dvoukanálového audia se považuje většinou 44100 Hz nebo 48000 Hz. To je dvakrát (kvůli dvěma kanálům) vyšší frekvence zvuku, než je průměrný člověk schopen slyšet. Počítač tedy ukládá tento počet čísel pro každou vteřinu zvuku.
- **bit depth** – počet bitů, který je určen pro uložení výšky amplitudy. Vlastně kolik mezistupňů existuje mezi nejvyšší a nejnižší měřitelnou amplitudou.
  - Za standardní bit depth se považuje obvykle 16, čili velikost amplitudy je rozdělena na  $2^{16} = 65536$  stupňů. Pro vysokou kvalitu (až na výjimky téměř neslyšitelnou) nahrávky se občas používá 24 bitů.

# AUDIO

Bezztrátová komprese audia funguje stejně jako bezztrátová komprese obrázku.



# AUDIO

Bezztrátová komprese audia funguje stejně jako bezztrátová komprese obrázku.  
Buď se mnoho po sobě jdoucích stejných velikostí amplitudy vyjádří ve dvou bytech jako „počet a velikost“ (run-length),

# AUDIO

Bezztrátová komprese audia funguje stejně jako bezztrátová komprese obrázku. Buď se mnoho po sobě jdoucích stejných velikostí amplitudy vyjádří ve dvou bytech jako „počet a velikost“ (run-length), nebo se velikosti amplitud zakódují tak, aby nejčastější velikosti měly nejkratší kódy (Huffman).

## AUDIO – FORMÁTY

- Nejběžnější formáty, ve kterých se ukládají audio soubory **bez jakékoli komprese**, jsou **WAV** (.wav, Wave Audio File Format) nebo **AIFF** (.aiff, Audio Interchange File Format).
- Bezztrátově komprimované audio se ukládá v mnoha různých formátech. Nejčastěji asi jako
  - **FLAC** (.flac, Free Lossless Audio Codec), který používá metodu komprese podobnou Huffmanově stromu;

## AUDIO – FORMÁTY

- Nejběžnější formáty, ve kterých se ukládají audio soubory **bez jakékoli komprese**, jsou **WAV** (.wav, Wave Audio File Format) nebo **AIFF** (.aiff, Audio Interchange File Format).
- Bezztrátově komprimované audio se ukládá v mnoha různých formátech. Nejčastěji asi jako
  - **FLAC** (.flac, Free Lossless Audio Codec), který používá metodu komprese podobnou Huffmanově stromu;
  - **Monkey's Audio** (.ape) je nejnovější bezztrátová formát audia a využívá složitý algoritmus pro kódování. Tím dosahuje nejmenší velikosti souboru ze všech bezztrátových formátů za cenu náročnosti dekódování;

## AUDIO – FORMÁTY

- Nejběžnější formáty, ve kterých se ukládají audio soubory **bez jakékoli komprese**, jsou **WAV** (.wav, Wave Audio File Format) nebo **AIFF** (.aiff, Audio Interchange File Format).
- Bezztrátově komprimované audio se ukládá v mnoha různých formátech. Nejčastěji asi jako
  - **FLAC** (.flac, Free Lossless Audio Codec), který používá metodu komprese podobnou Huffmanově stromu;
  - **Monkey's Audio** (.ape) je nejnovější bezztrátová formát audia a využívá složitý algoritmus pro kódování. Tím dosahuje nejmenší velikosti souboru ze všech bezztrátových formátů za cenu náročnosti dekódování;
  - **ALAC** (.alac, Apple Lossless Audio Codec) – vlastně Apple verze FLACu. Dosahuje lehce nižších velikostí souboru, ale dekódování je průměrně čtyřikrát náročnější. Funguje na principu lineární predikce – vlastně vylepšená verze run-length.

# OBRÁZKY

- Obrázky jsou na počítači často uloženy jako posloupnosti trojic (R,G,B) – tzv. **rastrový formát**.



# OBRÁZKY

- Obrázky jsou na počítači často uloženy jako posloupnosti trojic (R,G,B) – tzv. **rastrový formát**.
- Existují též **vektorové formáty** obrázků, kde je obrázek reprezentován jako čáry a křivky, které má počítat vykreslit. Ty zatím přeskočíme.

# OBRÁZKY

- Obrázky jsou na počítači často uloženy jako posloupnosti trojic (R,G,B) – tzv. **rastrový formát**.
- Existují též **vektorové formáty** obrázků, kde je obrázek reprezentován jako čáry a křivky, které má počítat vykreslit. Ty zatím přeskočíme.
- V hlavičce každého souboru s obrázkem jsou přinejmenším
  - šířka a výška,

# OBRÁZKY

- Obrázky jsou na počítači často uloženy jako posloupnosti trojic (R,G,B) – tzv. **rastrový formát**.
- Existují též **vektorové formáty** obrázků, kde je obrázek reprezentován jako čáry a křivky, které má počítat vykreslit. Ty zatím přeskočíme.
- V hlavičce každého souboru s obrázkem jsou přinejmenším
  - šířka a výška,
  - **color depth** (hloubka barev) – počet bitů užitých k uložení každé trojice R,G,B. Nejčastěji to je 24 bitů (tedy 8 bitů – číslo mezi 0 a 255 pro každou složku). Některé aplikace pro práci s grafikou používají i 48 bitů.

# OBRÁZKY

- Obrázky jsou na počítači často uloženy jako posloupnosti trojic (R,G,B) – tzv. **rastrový formát**.
- Existují též **vektorové formáty** obrázků, kde je obrázek reprezentován jako čáry a křivky, které má počítat vykreslit. Ty zatím přeskočíme.
- V hlavičce každého souboru s obrázkem jsou přinejmenším
  - šířka a výška,
  - **color depth** (hloubka barev) – počet bitů užitých k uložení každé trojice R,G,B. Nejčastěji to je 24 bitů (tedy 8 bitů – číslo mezi 0 a 255 pro každou složku). Některé aplikace pro práci s grafikou používají i 48 bitů.
- Bezztrátová komprese obrázku funguje obvykle opět na principu run-length nebo nějakého stromu četností.

## OBRÁZKY – FORMÁTY

Bezztrátově komprimované obrázky jsou ukládány často ve formátech

- **GIF** (.gif, Graphics Interchange Format), který využívá metodu komprese typu run-length zvanou LZW (Lempel-Ziv-Welch). Formát GIF je v základu limitován na 256 barev (1 B pro celou trojici R,G,B), ale moderní verze podporují i 8bitovou hloubku. Formát GIF poskytuje nejlepší komprimaci v případech, kdy obsahem obrázku je jednoduchá grafika s jednobarevnými plochami.

## OBRÁZKY – FORMÁTY

Bezztrátově komprimované obrázky jsou ukládány často ve formátech

- **GIF** (.gif, Graphics Interchange Format), který využívá metodu komprese typu run-length zvanou LZW (Lempel-Ziv-Welch). Formát GIF je v základu limitován na 256 barev (1 B pro celou trojici R,G,B), ale moderní verze podporují i 8bitovou hloubku. Formát GIF poskytuje nejlepší komprimaci v případech, kdy obsahem obrázku je jednoduchá grafika s jednobarevnými plochami.
- **PNG** (.png, Portable Network Graphics) – bohatě nejoblíbenější bezztrátový formát podporující navíc **průhlednost**. Byl vytvořen jako open-source alternativa k GIF, jehož tvůrci si nechali algoritmus LZW patentovat. PNG používá ke komprimaci formát DEFLATE, postavený na principu Huffmanova stromu. Z toho důvodu exceluje v komprimaci obrázků s velkými jednobarevnými plochami.

# OBRÁZKY – FORMÁTY

- **WebP** (.webp, Web Picture) – formát vytvořený Googlem v roce 2010 podporující jak ztrátovou tak bezztrátovou kompresi. Jeho primárním účelem je nahrazení formátů PNG a JPEG jako hlavních formátů obrázků na webu. WebP používá ke kompresi formát **VP8**, což je komprimační formát na blokové bázi, ale výrazně odlišný od Huffmanova.

# VIDEO

Video je v čisté (raw) podobě v počítači uloženo jako posloupnost obrázků – každý pro jeden snímek.



# VIDEO

Video je v čisté (raw) podobě v počítači uloženo jako posloupnost obrázků – každý pro jeden snímek.

Na rozdíl od obrázků a audia, je komprese video pro jeho uložení v zásadě nezbytná a většina kamer proto umí video komprimovat už během jeho natáčení.

# VIDEO

Video je v čisté (raw) podobě v počítači uloženo jako posloupnost obrázků – každý pro jeden snímek.

Na rozdíl od obrázků a audia, je komprese video pro jeho uložení v zásadě nezbytná a většina kamer proto umí video komprimovat už během jeho natáčení.

Pro představu: pětiminutové 4K 60 FPS video bez žádné komprese by zabralo přibližně

$$3840 \cdot 2160 \cdot 3 \cdot 60 \cdot 300 = 447897600000$$

bytů, neboli asi 448 GB.

# VIDEO

Komprimaci videa lze rozdělit na dvě obecné metody:

- **Prostorová** (intraframe) komprese – komprese na úrovni jednotlivých snímků.  
Protože jednotlivé snímky jsou obrázky, neliší se tato metoda nijak od komprese obrázků.

# VIDEO

Komprimaci videa lze rozdělit na dvě obecné metody:

- **Prostorová** (intraframe) komprese – komprese na úrovni jednotlivých snímků. Protože jednotlivé snímky jsou obrázky, neliší se tato metoda nijak od komprese obrázků.
- **Časová** (interframe) komprese – komprese na úrovni souvislosti mezi bezprostředními snímky. Mnoho videí je točeno způsobem, který upřednostňuje prostředek obrazovky a obsah krajů zůstává po mnoha snímků nezměněn.

# VIDEO

Komprimaci videa lze rozdělit na dvě obecné metody:

- **Prostorová** (intraframe) komprese – komprese na úrovni jednotlivých snímků. Protože jednotlivé snímky jsou obrázky, neliší se tato metoda nijak od komprese obrázků.
- **Časová** (interframe) komprese – komprese na úrovni souvislosti mezi bezprostředními snímky. Mnoho videí je točeno způsobem, který upřednostňuje prostředek obrazovky a obsah krajů zůstává po mnoha snímků nezměněn. Pixely v těchto částech proto není třeba ukládat zvlášť v každém snímku – stačí si pamatovat, které pixely se nemění a po kolik snímků.

# VIDEO

Kdyby se například blok  $64 \times 64$  pixelů někde v našem 4K 60 FPS videu neměnil, mohli bychom ho bezztrátově zmenšit z původních  $64 \cdot 64 \cdot 3 \cdot 60 \cdot 300 = 221184000$  bytů na  $64 \cdot 64 \cdot 3 \cdot 2 = 24576$  bytů, čili z přibližně 221 MB na 24.5 KB.

# VIDEO

Kdyby se například blok  $64 \times 64$  pixelů někde v našem 4K 60 FPS videu neměnil, mohli bychom ho bezztrátově zmenšit z původních  $64 \cdot 64 \cdot 3 \cdot 60 \cdot 300 = 221184000$  bytů na  $64 \cdot 64 \cdot 3 \cdot 2 = 24576$  bytů, čili z přibližně 221 MB na 24.5 KB.

Pro **časovou** bezztrátovou komprimaci videa se používají téměř výhradně run-length metody podobného typu.

## VIDEO – FORMÁTY

Kvůli paměťové náročnosti se formáty bezztrátové komprese videa téměř nepoužívají.



## VIDEO – FORMÁTY

Kvůli paměťové náročnosti se formáty bezztrátové komprese videa téměř nepoužívají. Za zmínu stojí například formáty používané ve filmovém průmyslu určené k archivaci, jako

- **DPX** (.dpx, Digital Moving Picture Exchange Bitmap) je video formát užívající pouze run-length časovou kompresi. Jednotlivé snímky jsou uložené v čistém (též raw či bitmap) formátu.

## VIDEO – FORMÁTY

Kvůli paměťové náročnosti se formáty bezztrátové komprese videa téměř nepoužívají. Za zmínu stojí například formáty používané ve filmovém průmyslu určené k archivaci, jako

- **DPX** (.dpx, Digital Moving Picture Exchange Bitmap) je video formát užívající pouze run-length časovou kompresi. Jednotlivé snímky jsou uložené v čistém (též raw či bitmap) formátu.
- **OpenEXR** (.exr) – open-source bezztrátový komprimační formát využívající rovněž run-length časovou komprimaci a (obvykle) DEFLATE formát komprese pro prostorovou kompresi (jako PNG).

# ZTRÁTOVÁ KOMPRESE

## ZTRÁTOVÁ KOMPRESE OBECNĚ

Ztrátová komprese funguje na obecném principu ztotožnění částí dat, která jsou bezprostředně u sebe a jsou dostatečně podobná na to, aby tento proces nevedl ke snadno pozorovatelné ztrátě kvality.

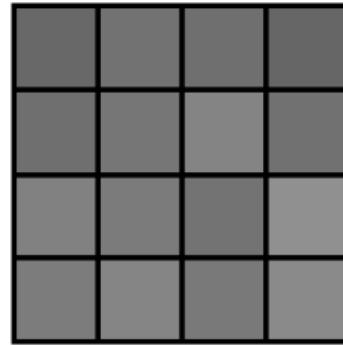
## ZTRÁTOVÁ KOMPRESE OBECNĚ

Ztrátová komprese funguje na obecném principu ztotožnění částí dat, která jsou bezprostředně u sebe a jsou dostatečně podobná na to, aby tento proces nevedl ke snadno pozorovatelné ztrátě kvality.

Takto upravená data lze pak mnohem účinněji komprimovat jedním z **bezztrátových** komprimacích postupů.

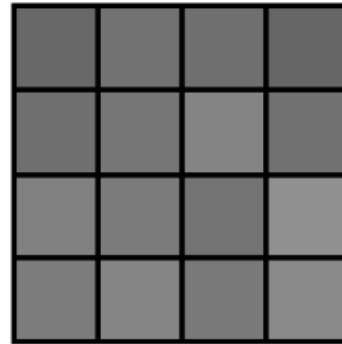
## PŘÍKLAD – OBRÁZEK

Vezměme třeba následující obrázek:



## PŘÍKLAD – OBRÁZEK

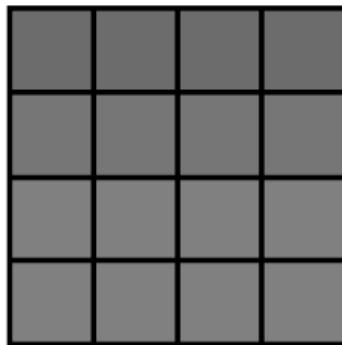
Vezměme třeba následující obrázek:



Každý jeho pixel má ve skutečnosti jinou barvu, takže bezztrátová komprese nemůže zmenšit jeho velikost vůbec.

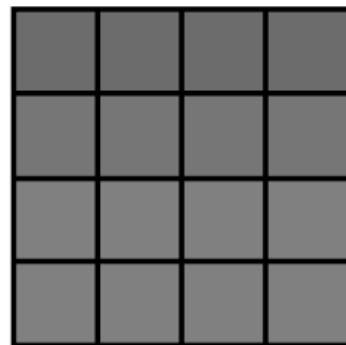
## PŘÍKLAD – OBRÁZEK

Když vynalezneme algoritmus, který umí poznat, že jsou pixely „velmi podobné“, můžeme například vzít průměrnou barvu v každém řádku a přetvořit obrázek na



## PŘÍKLAD – OBRÁZEK

Když vynalezneme algoritmus, který umí poznat, že jsou pixely „velmi podobné“, můžeme například vzít průměrnou barvu v každém řádku a přetvořit obrázek na

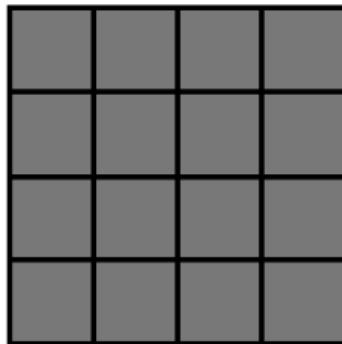


Tento obrázek se již dá run-length kompresí zapsat jako



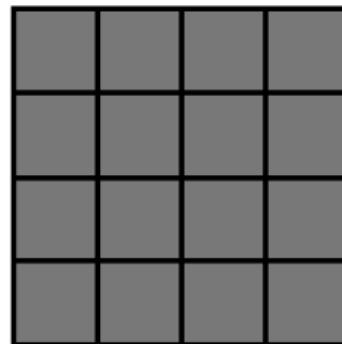
## PŘÍKLAD – OBRÁZEK

Jednotlivé řádky obrázku se ale od sebe taky příliš neliší, takže když všech 16 pixelů nahradíme jejich průměrnou barvou, dostaneme



## PŘÍKLAD – OBRÁZEK

Jednotlivé řádky obrázku se ale od sebe taky příliš neliší, takže když všech 16 pixelů nahradíme jejich průměrnou barvou, dostaneme



Z tohoto obrázku udělá run-length komprese zkrátka



# AUDIO – FORMÁTY

Některé oblíbené **ztrátové** audio formáty jsou

- **MP3** (.mp3, MPEG-1 Audio Layer III) – jeden z nejstarších a nejpoužívanějších formátů.  
Pro kompresi využívá primárně dvou věcí:
  - neschopnosti lidského ucha slyšet tóny vysokých frekvencí přes tóny frekvencí nižších;
  - aproximačního algoritmu známého jako **MDCT** (Modified Discrete Cosine Transform).

# AUDIO – FORMÁTY

Některé oblíbené **ztrátové** audio formáty jsou

- **MP3** (.mp3, MPEG-1 Audio Layer III) – jeden z nejstarších a nejpoužívanějších formátů. Pro kompresi využívá primárně dvou věcí:
  - neschopnosti lidského ucha slyšet tóny vysokých frekvencí přes tóny frekvencí nižších;
  - aproximačního algoritmu známého jako **MDCT** (Modified Discrete Cosine Transform).
- **AAC** (.aac, Advanced Audio Coding) – míňen jako nástupce MP3. Využívá lepších komprimačních algoritmů než MP3 a dosahuje menších velikostí za stejných ztrát kvality. Díky tomu se ujal jako hlavní formát zvukových stop v oblíbených ztrátových video formátech MPEG-4 nebo Matroska.

# OBRÁZEK – FORMÁTY

Pár ztrátových formátů obrázků:

- **JPEG** (.jpg, Joint Photographic Experts Group) je zcela jistě nejpoužívanější ztrátový formát obrázku. Využívá faktu, že lidské oko není schopné vnímat ostré posuny v intenzitě barvy, ale místo toho vlastně do sebe barvy prolne. Pokud je toto prolnutí již součástí obrázku, tak člověk v ideálním případě nic nepozná a velikost obrázku se tím dá výrazně zmenšit. Tento algoritmus „prolínání“ se nazývá DCT (Discrete Cosine Transform).

## OBRÁZEK – FORMÁTY

Pár ztrátových formátů obrázků:

- **JPEG** (.jpg, Joint Photographic Experts Group) je zcela jistě nejpoužívanější ztrátový formát obrázku. Využívá faktu, že lidské oko není schopné vnímat ostré posuny v intenzitě barvy, ale místo toho vlastně do sebe barvy prolne. Pokud je toto prolnutí již součástí obrázku, tak člověk v ideálním případě nic nepozná a velikost obrázku se tím dá výrazně zmenšit. Tento algoritmus „prolínání“ se nazývá DCT (Discrete Cosine Transform).
- **WebP** – již zmíněný formát vynalezený Googlem, který má za účel nahradit JPEG a PNG na webu. Umí ztrátovou i bezztrátovou kompresi.

## VIDEO – FORMÁTY

U video souborů je výrazný rozdíl mezi **kodekem** a **datovým kontejnerem**.

# VIDEO – FORMÁTY

U video souborů je výrazný rozdíl mezi **kodekem** a **datovým kontejnerem**.

- **Kodek** (Video Coding Format) je komprimační formát videa.

# VIDEO – FORMÁTY

U video souborů je výrazný rozdíl mezi **kodekem** a **datovým kontejnerem**.

- **Kodek** (Video Coding Format) je komprimační formát videa.
- **Datový kontejner** (Video Container Format) je formát, který v sobě uchovává několik video, audio i titulkových stop, z nichž všechny mohou být obvykle komprimovány v různých formátech.

# VIDEO – FORMÁTY

U video souborů je výrazný rozdíl mezi **kodekem** a **datovým kontejnerem**.

- **Kodek** (Video Coding Format) je komprimační formát videa.
- **Datový kontejner** (Video Container Format) je formát, který v sobě uchovává několik video, audio i titulkových stop, z nichž všechny mohou být obvykle komprimovány v různých formátech.

O kvalitě videa nejvíce rozhoduje parametr **bit rate**, který popisuje, kolik bitů video souboru je přeneseno za vteřinu. Účelem ztrátových kodeků je s klesajícím bit ratem co nejméně snižovat pozorovatelnou kvalitu.

## VIDEO – ZTRÁTOVÉ KODEKY

Aktivně se používají vlastně jen dva ztrátové kodeky.

- **MPEG-4** (Moving Picture Experts Group) je ztrátový formát, který využívá
  - algoritmickou metodu zvanou „kompenzace pohybu“. Další snímek se často liší od předchozího pouze pohybem objektu na kameře nebo samotné kamery. To umožňuje předpovědět pozici mnoha pixelů na dalším snímku bez jejich explicitního ukládání v paměti.

# VIDEO – ZTRÁTOVÉ KODEKY

Aktivně se používají vlastně jen dva ztrátové kodeky.

- **MPEG-4** (Moving Picture Experts Group) je ztrátový formát, který využívá
  - algoritmickou metodu zvanou „kompenzace pohybu“. Další snímek se často liší od předchozího pouze pohybem objektu na kameře nebo samotné kamery. To umožňuje předpovědět pozici mnoha pixelů na dalším snímku bez jejich explicitního ukládání v paměti.
  - „prolnutí“ bezprostředních snímků pomocí DCT (Discrete Cosine Transform).
- **AVC, též H.264**, (Advanced Video Coding) a jeho nástupce **HEVC, též H.265**, (High Efficiency Video Coding) je kodek, který vznikl za účelem reprodukce videa ve stejné kvalitě jako MPEG-4 s méně než polovičním bit ratem. Ujal se jako hlavní formát videa na Blu-ray discích a streamingových platformách. Používá kombinaci mnoha sofistikovaných kompresních algoritmů.

## VIDEO – DATOVÉ KONTEJNERY

Formátů datových kontejnerů je *velmi mnoho*. Uvedeme si pár rozšířených:

- **Matroska** (.mkv) je open-source projekt s účelem vytvoření kontejneru, který je schopen uchovat teoreticky libovolné množství video, audio i titulkových stop v jednom souboru. Neklade žádná omezení na formát videa ani audia.

# VIDEO – DATOVÉ KONTEJNERY

Formátů datových kontejnerů je *velmi mnoho*. Uvedeme si pár rozšířených:

- **Matroska** (.mkv) je open-source projekt s účelem vytvoření kontejneru, který je schopen uchovat teoreticky libovolné množství video, audio i titulkových stop v jednom souboru. Neklade žádná omezení na formát videa ani audia.
- **AVI** (.avi, Audio Video Interleave) je kontejner vytvořený Microsoftem pro použití ve Windowsu. Pro uchování video i audio stop používá kontejner **RIFF** (Resource Interchange File Format), který funguje na principu rozdělení souborů na stejně velké kusy, každý s vlastní hlavičkou obsahující informace o následujících datech. AVI podporuje libovolný formát audia, videa i titulků. Díky své implementaci je též schopno sloučit bezprostřední kusy videa různých kodeků, ačkoli této vlastnosti se zřídka užívá.

## VIDEO – DATOVÉ KONTEJNERY

- **QTFF** (.mov, QuickTime File Format) je kontejner vytvořený Applem. Nepodporuje mnoho video a audio formátů, ale na rozdíl od ostatních kontejnerů si stopy ukládá v typu „složkové“ struktury. To jej dělá zvlášť užitečným pro editaci videa, neboť přesouvání částí je možné pouhým přepsáním údajů v tabulce, bez nutnosti kopírování dat. Z toho důvodu je využíván jako primární kontejner například v Adobe produktech.

## VIDEO – DATOVÉ KONTEJNERY

- **QTFF** (.mov, QuickTime File Format) je kontejner vytvořený Applem. Nepodporuje mnoho video a audio formátů, ale na rozdíl od ostatních kontejnerů si stopy ukládá v typu „složkové“ struktury. To jej dělá zvlášť užitečným pro editaci videa, neboť přesouvání částí je možné pouhým přepsáním údajů v tabulce, bez nutnosti kopírování dat. Z toho důvodu je využíván jako primární kontejner například v Adobe produktech.
- **MP4** (.mp4, MPEG-4 Part 14) též nepodporuje mnoho video ani audio formátů a obecně oproti ostatním uvedeným kontejnerům nepřináší nic nového. Hlavní důvody jeho rozšíření jsou zpětná kompatibilita s předchozími verzemi, snadné zašifrování obsažených stop a možnost uložení teoreticky neomezeného množství metadat ve formátu XML.