

# Obliczenia Naukowe

Wykład 6.

O kompresji i zawartości informacyjnej

15. IV 2021

Bartek Wilczyński

# Plan na dziś

- Po co nam kompresja, korzyści/wady
- Kompresja bezstratna lub stratna
- Rodzaje danych do kompresji
  - Pliki tekstowe/binarne
  - Pliki dźwiękowe
  - Obrazy
  - Filmy
  - Sekwencje biologiczne
- Kody Huffmana
- Zawartość informacyjna i entropia

# Po co nam kompresja?

- Przechowywanie danych kosztuje
- Skompresowane dane zajmują mniej pamięci dyskowej
- Często łatwiej dane przechowywać niż kasować ale,...
- Odczytanie danych skompresowanych zajmuje więcej czasu procesora
- Jeśli kompresja jest stratna, to nie można odzyskać całości oryginalnej zawartości pliku ze skompresowanej wersji

# Co to jest kompresja

- Dla danego pliku, możemy przyjąć, że algorytm kompresji musi składać się z 2 funkcji:
  - Kompresującej  $K(p) \rightarrow pk$ , gdzie rozmiar  $pk$  jest zwykle mniejszy od rozmiaru  $p$
  - Dekompresującej  $D(pk) \rightarrow p'$ , gdzie  $p'=p$  w przypadku kompresji bezstratnej, a  $p' \sim p$  w przypadku kompresji stratnej
- W praktyce istotny jest czas potrzebny do wykonania funkcji  $K$  i  $D$ , choć w niektórych zastosowaniach istotniejszy jest czas wykonania  $D$  niż  $K$
- Oczywiście istotny jest też współczynnik kompresji  $|pk|/|p|$ , zwykle obliczany dla “typowych” danych.
- Zwykle zarówno koszt, jak i współczynnik kompresji zależy od danych, dlatego stosujemy różne funkcje  $K$  i  $D$  dla różnych danych

# Kompresja stratna sygnałów

- W przypadku sygnałów, często naturalne jest stosowanie kompresji stratnej
- Najprostsze co można zrobić, to manipulować częstotliwością próbkowania i liczbą bitów na próbkę
- Uzyskujemy w ten sposób “kompresję” stratną
- Repróbkowanie sygnałów zwykle wymaga interpolacji, choćby w postaci uśredniania sygnału
- Zmniejszenie liczby bitów na próbkę może czasami być kompresją bezstratną, np. Pliki tekstowe rzadko wykorzystują wszystkie 256 znaków ASCII

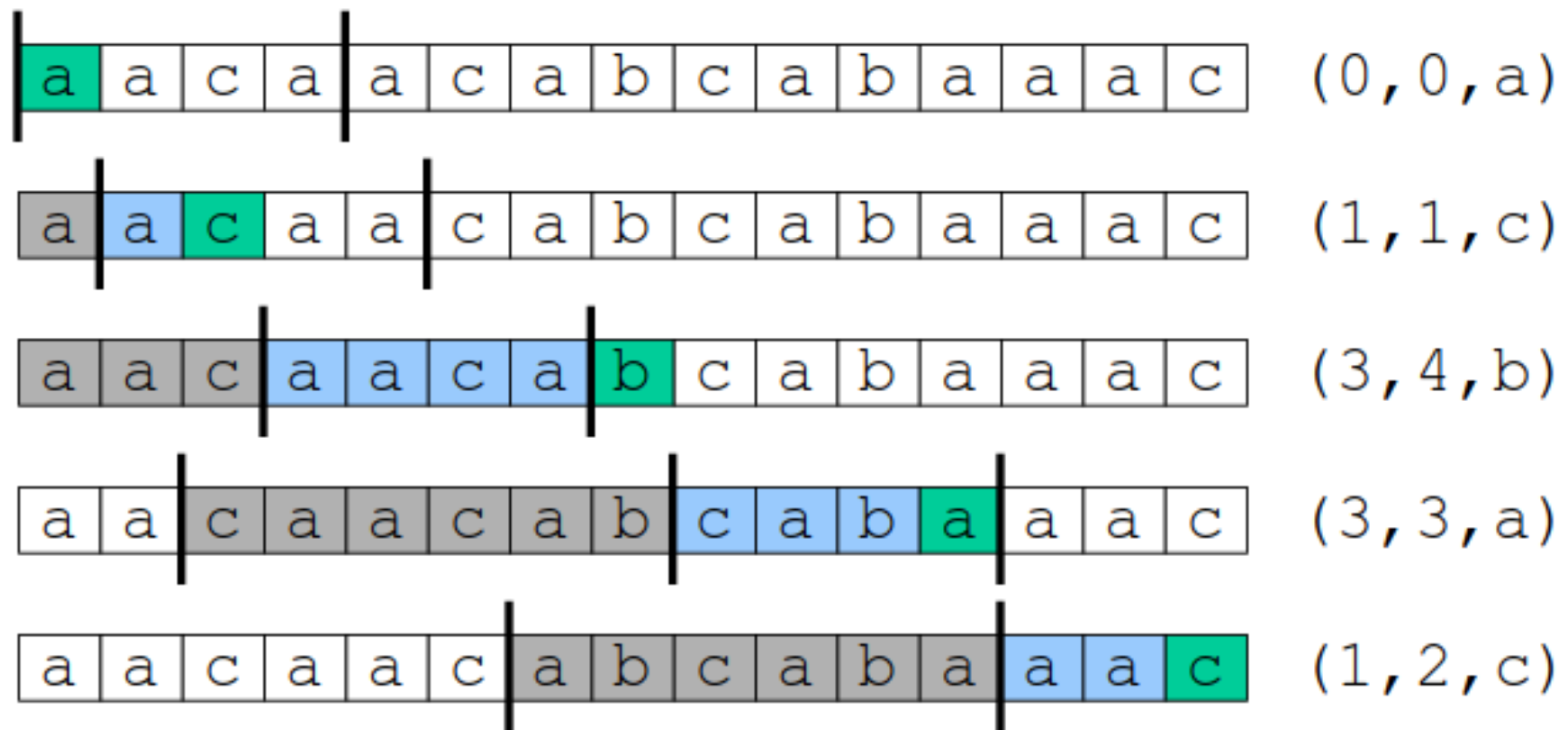
# Kompresja bezstratna


- Najprostszymi metodami kompresji bezstratnej dla strumienia danych (np. Pliku tekstowego) są metody słownikowe
- Możemy np. wykryć powtarzające się symbole i je zastąpić np:
- $K(\text{aaaacbgbbbbbbaaaddddd}) = \#4\text{acbg}\#4\text{b}\#3\text{a}\#4\text{d}$
- Zwykle potrzeba dodatkowego symbolu (np. “#”)
- W skrajnych przypadkach współczynnik kompresji może być  $>1$

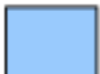
# Metoda prefiksowa Lempel, Ziv '77


- Sprytnym rozszerzeniem metody słownikowej jest metoda LZ77, której warianty są wykorzystywane m.in. w formatach gif, zip, png
- Zasada działania jest następująca:
  - Wczytaj kolejno okna danych o ustalonym rozmiarze
  - Dla każdego okna: zapisz kolejne prefiksy części niezakodowanej, jako trójki: pozycja w oknie, długość, kolejny znak, aż do wyczerpania okna

# LZ77: Example



 Dictionary (size = 6)

 Longest match

 Next character

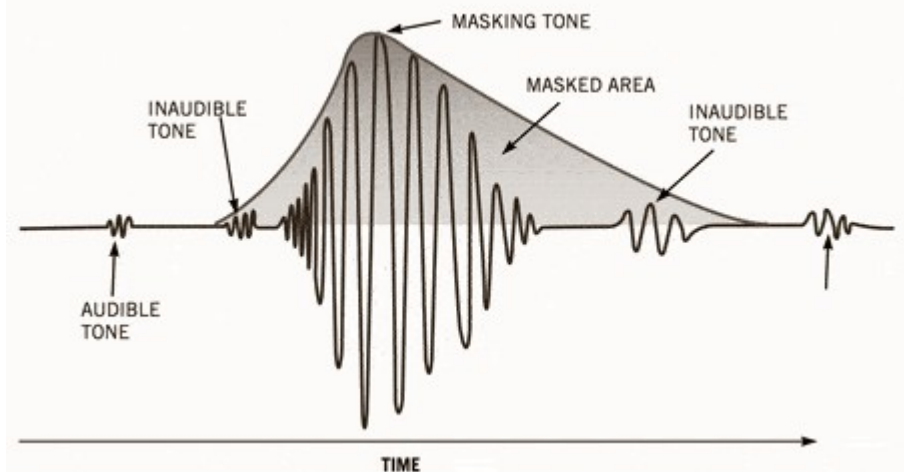


# Kompresja plików dźwiękowych

- Typowo, pliki w jakości “CD” mają 2 ścieżki stereo, próbkowane 44100Hz po 16 bitów na próbkę, czy 176KBps
- Jeśli nie potrzebujemy takiej jakości, możemy np. Po prostu zmniejszyć parametry sygnału, co często robią np. dyktafony i telefony
- Dla muzyki często stosuje się złożone algorytmy kompresji bezstratnej (np. FLAC) lub stratnej (MP3), często uzyskując dobrą jakość dźwięku przy 128kbps

# Maskowanie audio

- Jeśli słyszymy głośny dźwięk, to nie słyszymy cichszych dźwięków tuż przed nim i po nim
- Podobnie z harmonicznymi dźwiękami o różnej częstotliwości
- Badania empiryczne w zakresie niedoskonałości słuchu ludzkiego były prowadzone już w XIXw.
- Algorytmy takie jak MP3 wykorzystują te własności do zwiększonego zapisu plików audio

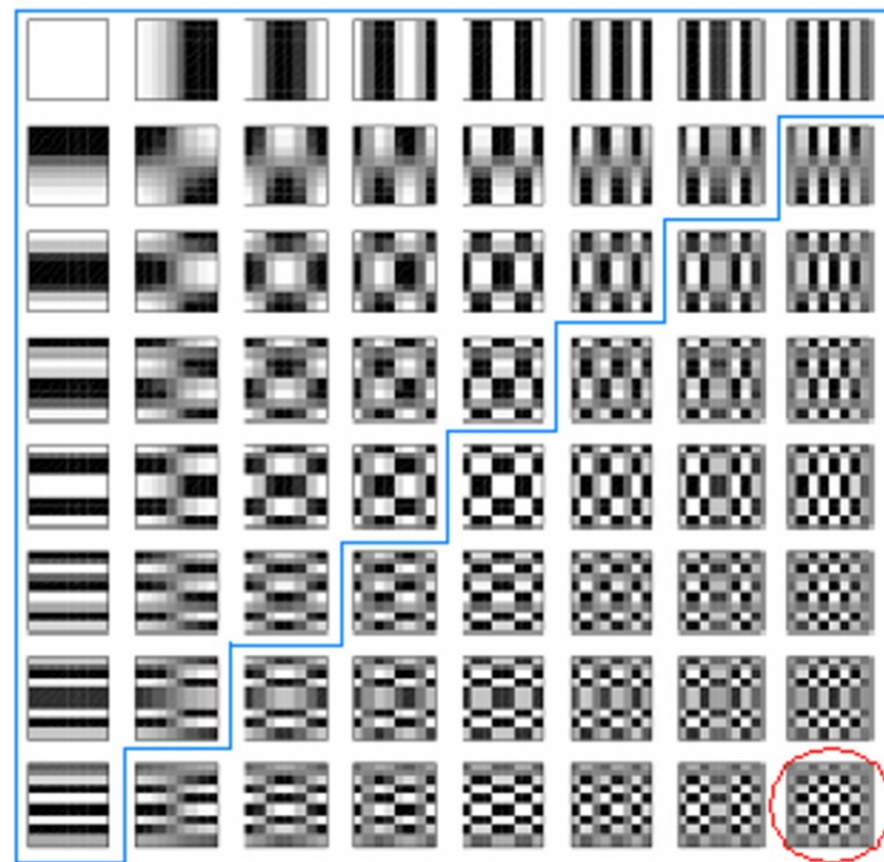


# Kompresja obrazów

- Ponieważ nasz wzrok jest niedoskonały, tak jak słuch, często stosujemy kompresję stratną dla obrazów
- Znowu - można manipulować głębią kolorów i rozdzielczością (tak jak w plikach GIF)
- W formacie JPG stosuje się dyskretną transformatę kosinusową (DCT), która jest podobna do transformaty Fouriera, ale używa jedynie kosinusów i wartości rzeczywistych

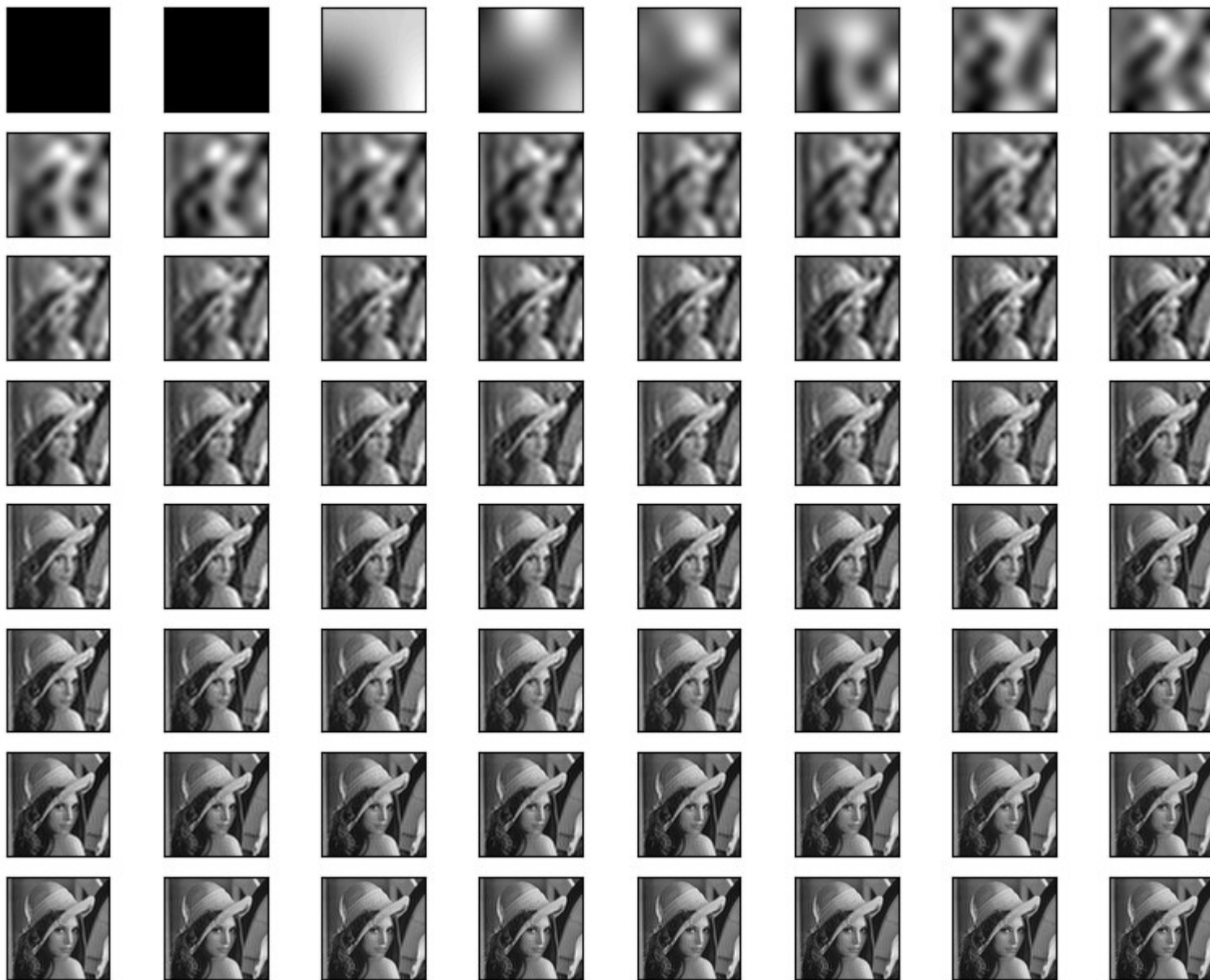
# Dyskretna transformata kosinusowa

- Zaproponowana w 1972
- Podobny pomysł jak w przypadku transformaty Fouriera,
- zapisujemy sygnał jako współczynniki funkcji bazowych
- Korzystamy z dyskretnych funkcji okresowych jako bazy
- Zerując współczynniki składowych o wysokich częstotliwościach uzyskujemy kompresję stratną
- Wykorzystywana w bardzo wielu podstawowych formatach graficznych/wideo (jpg, mpg, mp4...)



- Funkcje bazowe 2D DCT

# Kolejne przybliżenia obrazu w DCT

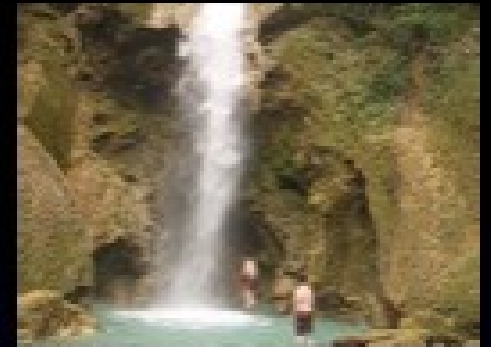


# Filmy, czyli sekwencje obrazów

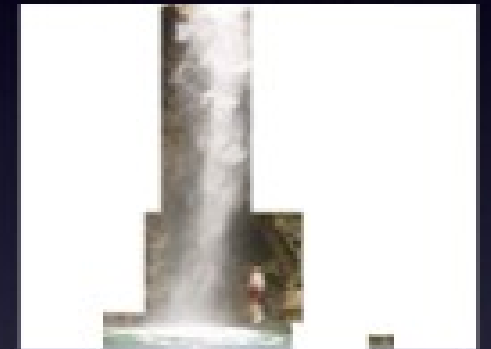
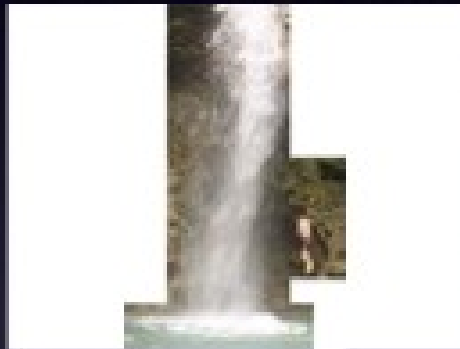
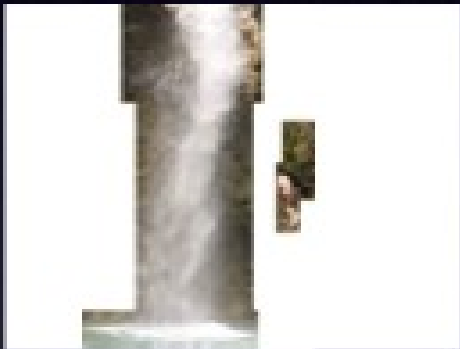
- Typowy film zawiera ok. 24 klatek/s, w stosunkowo niewielkiej rozdzielczości
- W większości filmów, większość par sąsiednich klatek jest niemal identyczna
- Jest to wykorzystywane w metodach kompresji cyfrowego obrazu, takich jak MPEG2/ DivX itp.
- Oprócz standardowej kompresji przez DCT, dodatkowo mamy różnicowe kodowanie kolejnych klatek, oprócz tzw. Klatek kluczowych

# Przykład kompresji wideo

## Full I-Frames



## Compression using P-Frames



# Sekwencje biologiczne

- Sekwencje DNA i białkowe są często przechowywane w plikach tekstowych dla czytelności, ale oczywiście jest to bardzo nieefektywne – jedna litera DNA może zajmować 2 bity, a znak ASCII zajmuje 8
- Najczęściej stosuje się standardowe metody kompresji takie jak gzip, pozwalające na względnie szybki dostęp do danych
- Osiąga się w ten sposób sporą kompresję, niemal bez straty czasu na dekompresję



# Rozmiar danych DNA pacjentów przyrasta szybciej niż przestrzeń dyskowa

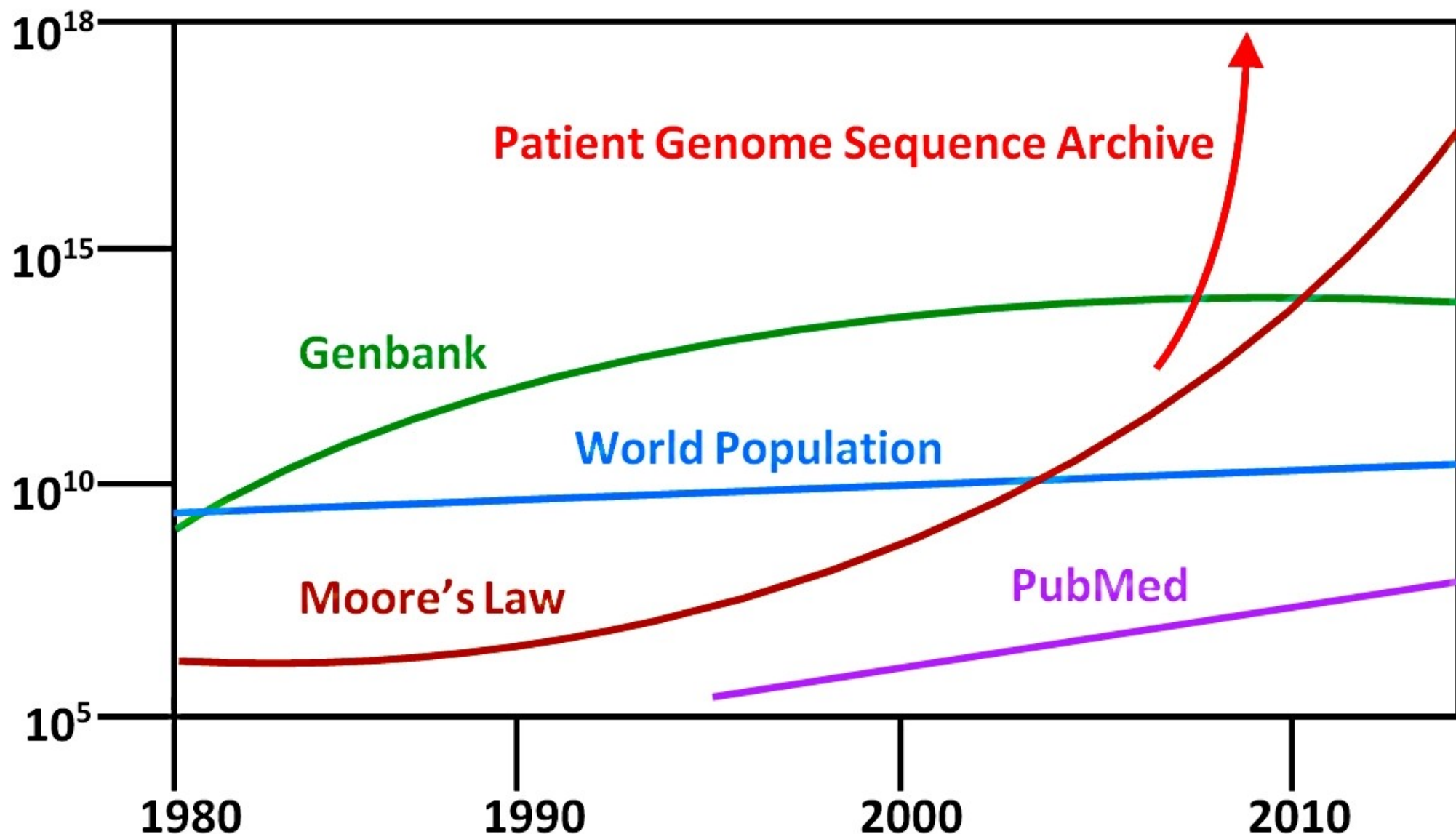


Figure 1: Approximate Growth of Different Data Populations

# Kompresja danych DNA i kompresja danych do DNA

- W związku z burzliwym rozwojem technologii sekwencjonowania DNA, pojawiają się problemy z przechowywaniem sekwencji DNA na dyskach
- Powstają nowe algorytmy kompresji specjalizowane do DNA, osiągające lepsze współczynniki kompresji
- Pojawiają się też pomysły odwrotne – skoro sekwencjonowanie DNA rozwija się tak szybko, to może można je wykorzystać do przechowywania danych?

# Przechowywanie danych w DNA

1 of 4

Towards practical, high-capacity, low-maintenance information storage in synthesized DNA

goldman2013.pdf

104,59%

Outline

Title

Authors

Abstract

References

Figure 1 Digital inf...

Figure 2 Scaling pr...

LETTER

doi:10.1038/nature11875

## Towards practical, high-capacity, low-maintenance information storage in synthesized DNA

Nick Goldman<sup>1</sup>, Paul Bertone<sup>1</sup>, Siyuan Chen<sup>2</sup>, Christophe Dessimoz<sup>1</sup>, Emily M. LeProust<sup>2</sup>, Botond Sipos<sup>1</sup> & Ewan Birney<sup>1</sup>

Digital production, transmission and storage have revolutionized how we access and use information but have also made archiving an increasingly complex task that requires active, continuing maintenance of digital media. This challenge has focused some interest on DNA as an attractive target for information storage<sup>1</sup> because of its capacity for high-density information encoding, longevity under easily achieved conditions<sup>2–4</sup> and proven track record as an information bearer. Previous DNA-based information storage approaches have encoded only trivial amounts of information<sup>5–7</sup> or were not amenable to scaling-up<sup>8</sup>, and used no robust error-correction and lacked examination of their cost-efficiency for large-scale information archival<sup>9</sup>. Here we describe a scalable method that can reliably store more information than has been handled before. We encoded computer files totalling 739 kilobytes of hard-disk storage and with an estimated Shannon information<sup>10</sup> of  $5.2 \times 10^6$  bits into a DNA code, synthesized this DNA, sequenced it and reconstructed the original files with 100% accuracy. Theoretical analysis indicates that our DNA-based storage scheme could be scaled far beyond current global information volumes and offers a realistic technology for large-scale, long-term and infrequently accessed digital archiving. In fact, current trends in technological advances are reducing DNA synthesis costs at a pace that should make our scheme cost-effective for sub-50-year archiving within a decade.

digits (ASCII text), giving a total of 757,051 bytes or a Shannon information<sup>10</sup> of  $5.2 \times 10^6$  bits (see Supplementary Information and Supplementary Table 1 for full details).

The bytes comprising each file were represented as single DNA sequences with no homopolymers (runs of  $\geq 2$  identical bases, which are associated with higher error rates in existing high-throughput sequencing technologies<sup>19</sup> and led to errors in a recent DNA-storage experiment<sup>9</sup>). Each DNA sequence was split into overlapping segments, generating fourfold redundancy, and alternate segments were converted to their reverse complement (see Fig. 1 and Supplementary Information). These measures reduce the probability of systematic failure for any particular string, which could lead to uncorrectable errors and data loss. Each segment was then augmented with indexing information that permitted determination of the file from which it originated and its location within that file, and simple parity-check error-detection<sup>10</sup>. In all, the five files were represented by a total of 153,335 strings of DNA, each comprising 117 nucleotides (nt). The perfectly uniform fragment lengths and absence of homopolymers make it obvious that the synthesized DNA does not have a natural (biological) origin, and so imply the presence of deliberate design and encoded information<sup>2</sup>.

We synthesized oligonucleotides (oligos) corresponding to our designed DNA strings using an updated version of Agilent Tech-

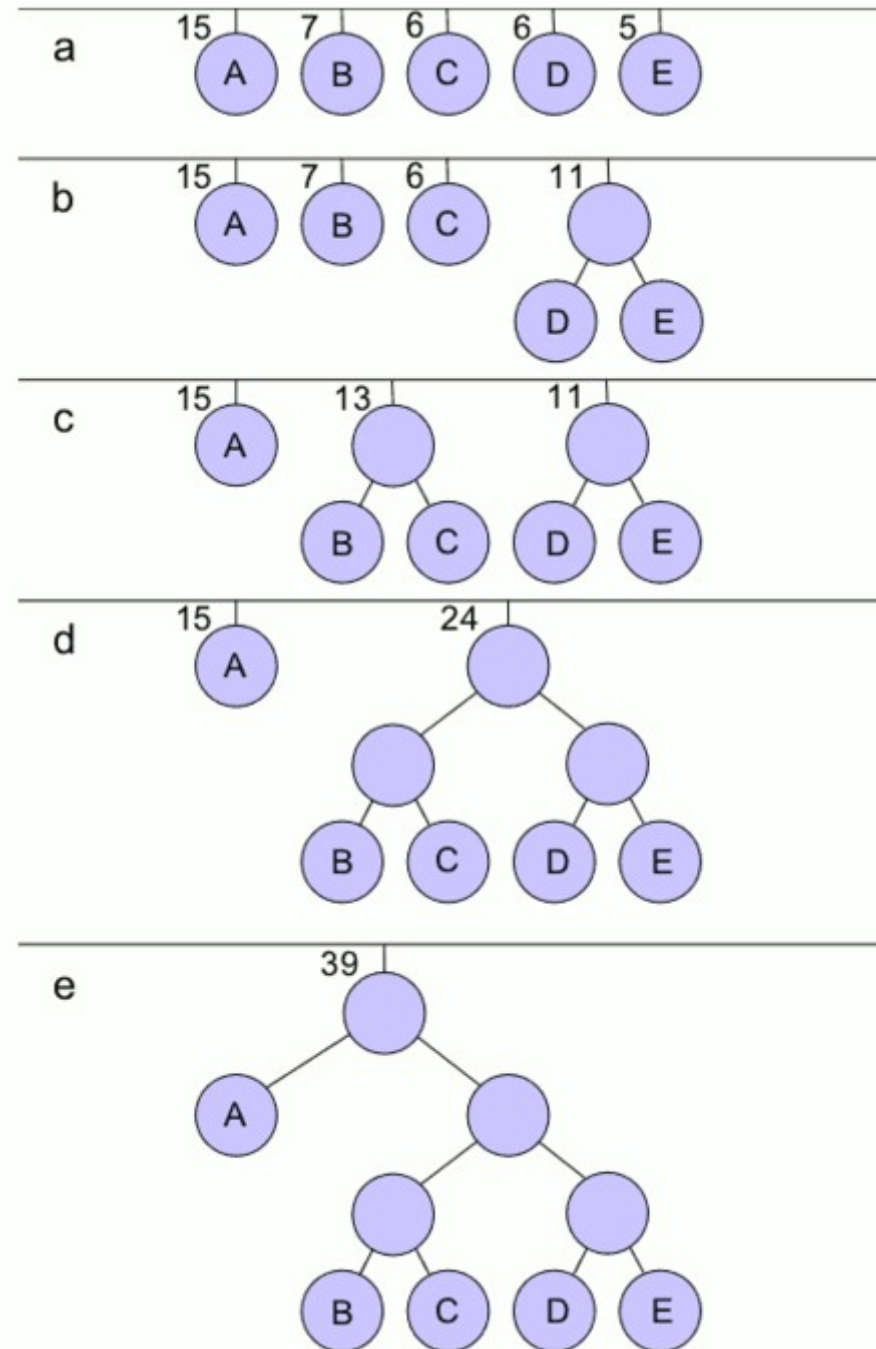
# Czy zawsze można lepiej skompresować?

- Czemu jedne dane “pakuja” się gorzej niż inne?
- Czy jeśli skompresujemy plik 2 razy ( $K(K(p))$ ), to uzyskamy lepszą kompresję?
- Czy istnieje “idealny” algorytm kompresji?
- Czy możemy określić jakieś ogólne własności danych, które ułatwiają/utrudniają kompresję?

# Kody Huffmana

- Załóżmy, że mamy skończony zestaw znaków pojawiających się z różnymi częstościami w strumieniu danych
- Czy możemy przypisać każdemu z nich symbol binarny (niekoniecznie równej długości), tak aby uzyskać optymalną “kompresję”?
- Okazuje się, że tak, jest to tzw. Kodowanie Huffmana
- Powstaje ono z konstrukcji drzewa binarnego, “od dołu”

- A – 15 wystąpień – 0
- B – 7 wystąpień - 100
- C – 6 wystąpień – 101
- D – 6 wystąpień - 110
- E – 5 wystąpień - 111



# Entropia rozkładu

- Claude Shannon interesował się naukowo kodowaniami i zbudował podwaliny współczesnej teorii informacji
- Zauważył, że entropia rozkładu prawdopodobieństwa

$$H(X) = \sum_{i=1}^n P(x_i) I(x_i) = - \sum_{i=1}^n P(x_i) \log_b P(x_i),$$

to dobra miara (jej jednostką jest bit) zawartości informacyjnej strumienia danych o zadanych prawdopodobieństwach  $n$  symboli.

- Jakie własności ma entropia zmiennej dwuwartościowej zależnie od prawdopodobieństwa jedynki?