sortowanie **na bazie** – wykorzystanie sortowania **ze zliczaniem**

sortowanie na bazie – polega na sortowaniu fragmentów danych, ***cyfr***, od cyfry najmniej znaczącej do najbardziej znaczącej.Można np. sortować liczby całkowite 64-pozycyjne jako czterocyfrowe wartości o bazie ”2 do 16” (tzn. tyle jest wartości jakie te cyfry mogą przyjąć) lub trzydziestodwu-cyfrowe wartości o bazie ”dwa do drugiej = 4”. To jakie wartości zostaną wybrane jako baza, zależy od właściwości samych danych (od ich oceny przez sortującego)

sortowanie **na bazie**

opis argumentów

int rxsort(int *data, int size, int p, int k)

data - *wskazuje na dane*

size - *ile elementów do posortowania*

p - *liczba cyfr w liczbach*

k – *baza (np. jeśli mamy liczby zapisane w układzie dziesiętnym i sortujemy po jednej cyfrze, to k=10)*

sortowanie **na bazie** – rxsort()

```
#include <limits.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
int rxsort(int *data, int size, int p, int k);

int rxsort(int *data, int size, int p, int k) {

int             *counts,
               *temp;
```

sortowanie **na bazie** – rxsort()

```
int             index,
                pval,
                i,
                j,
                n;
/**********************************
Alokacja pamięci na liczniki.
**********************************/
if ((counts = (int *)malloc(k * sizeof(int))) == NULL)
   return -1;
```

sortowanie **na bazie** – rxsort()

```
/*****************
Alokacja pamięci na elementy posortowane
*****************/

if ((temp = (int *)malloc(size * sizeof(int))) == NULL)
   return -1;
```

sortowanie **na bazie** – rxsort()

```
/* sortowanie od pozycji najmniej znaczącej do najbardziej
znaczącej */

  for (n=0; n<p; n++) {

/* poniżej inicjalizacja liczników  */

  for (i=0; i<k; i++)

  counts[i] = 0;
```

sortowanie **na bazie** – rxsort()

```
/* Wyliczenie wartości pozycji.                                    */
pval = (int)pow((double)k, (double)n);

/* Zliczanie wystąpień poszczególnych cyfr  */

for (j = 0; j < size; j++) {
   index = (int)(data[j] / pval) % k;
   counts[index] = counts[index] + 1;
}
```

sortowanie **na bazie** – rxsort()

```
/* Korekta liczników w celu uwzględnienia liczników je
poprzedzających.   */
   for (i = 1; i < k; i++)
      counts[i] = counts[i] + counts[i - 1];
/* Użycie liczników do umieszczenia poszczególnych elementów
we właściwych miejscach /*
   for (j = size - 1; j >= 0; j--) {
      index = (int)(data[j] / pval) % k;
      temp[counts[index] - 1] = data[j];
      counts[index] = counts[index] - 1;
   }
```

sortowanie **na bazie** – rxsort()

```
/*  Przygotowanie do zwrócenia posortowanych dotąd danych */
   memcpy(data, temp, size * sizeof(int));


}
/* Zwolnienie pamięci zaalokowanej na sortowanie  */
        free(counts);
        free(temp);
        return 0;
}
```

sortowanie **na bazie** – rxsort()

opis argumentów

int rxsort(int *data, int size, int p, int k)

data - *wskazuje na dane*

size - *ile elementów do posortowania*

p - *liczba cyfr w liczbach*

k – *baza*

przykład sortowania **na bazie** – wykorzystanie sortowania **ze zliczaniem**

302	6**1**1	9**0**1	1**0**2
253	9**0**1	3**0**2	2**5**3
611	3**0**2	1**0**2	3**0**2
901	1**0**2	6**1**1	5**2**9
529	2**5**3	5**2**9	6**1**1
102	5**2**9	2**5**3	9**0**1
n=0	n=1	n=2	n=3

(baza = {0,1,2,4,5,6,7,8,9})

sortowanie zbioru liczb jako liczb zapisanych w systemie dziesiętnym)

Kompresja danych

Kompresja danych polega na zmniejszeniu liczby bitów potrzebnych do zapisania danych. Podstawy teoretyczne – **teoria informacji (dział matematyki).** Kompresja i dekompresja danych.

Dlaczego dane w ogóle można skompresować?

Kompresja stratna, kompresja bezstratna.

....zajmujemy się teraz kompresją bezstratną...

Każde dane można scharakteryzować jeśli chodzi o ich zawartość informacyjną używając pojęcia zapożyczonego z termodynamiki – **entropia. Definiujemy ją dla informacji na poziomie ”sygnału”.**

Kompresja danych

Co to jest informacja?

- informacja na poziomie sygnału
- informacja na poziomie semantycznym (zdania...)
- rozumowanie logiczne

Na którym poziomie odbywać się może kompresja?

Claude Shannon

Warren Weaver, 1948 rok

The Bell System Technical Journal, vol. 27, pages 379-423, 623-656, July & October, 1948, „A mathematical theory of communication”

Kompresja danych

- Przykład informacji na poziomie semantycznym (zdania...)

„linie proste a,b,c należą do tej samej płaszczyzny π”

Kompresja danych

•Przykład rozumowania logicznego

„Czy liczbę sqrt(2) da się przedstawić jako iloraz dwóch liczb całkowitych?”

(nie da się wyliczyć ilości informacji (?), można przedstawić dowód, że się nie da)

Kompresja danych

Fundamentalnym problemem komunikacji jest reprodukowanie możliwie wiernie (bezstratnie lub małostratnie) wiadomości. Wiadomość ta jest przesyłana z punktu do punktu (czasoprzestrzeni). Na poziomie sygnałów nie rozpatruje się znaczenia (sensowności) tych wiadomości. Istotne jest to, że przesłana wiadomość jest jedną wybraną ze zbioru możliwych wiadomości. System komunikacyjny musi być opracowany tak, by każdą z tych możliwych wiadomości potrafił przesłać; każdą gdyż przed przesłaniem informacji nie wiadomo która to z możliwych wiadomości będzie przesyłana.

To z rozważań o komunikacji pojawiła się „jednostka informacji w sensie sygnału” – jeden bit.

Kompresja danych

System komunikacji składa się z:

-źródła informacji

-transmittera (zamienia wiadomość na sygnały odpowiednie dla kanału)

-kanału (to medium transportujące – kabel, fala radiowa, światłowód)

-odbiornik (działa odwrotnie do transmittera)

-odbiornik informacji (osoba, dysk optyczny, robot przemysłowy...)

Oczywiście można informację przesyłać w sposób nadmiarowy, np. tą samą wiadomość przesłać dwoma listami czy dwukrotnie przeprowadzić rozmowy telefoniczne czy dwukrotnie przesłać datagram używając jako kanału Internetu; ile trzeba przesłać minimalnie by wiadomość była przesłana bezstratnie ?

Kompresja danych

Rozważmy: Niech system komunikacyjny może przyjąć jeden z K podstanów.

Teraz a) przesyła N sygnałow ; b) przesyła 2N sygnałow .

O ile więcej przesyła informacji w przypadku b) w porównaniu do przypadku a) ?

(...i jeszcze chcemy by informacja miała własność addytywności...)

Kompresja danych

Gdyby liczyć to w ilości kombinacji możliwych wiadomości, to w przypadku **a)** tych możliwości jest K do potęgi N, a w przypadku **b)** jest ich K do potęgi 2N. Czy to byłaby właściwa miara ilości informacji przekazywanej ?

Jakie cechy winna mieć miara informacji (będziemy ją nazywać entropią) ?

Powinna **liniowo** zależeć od ilości sygnałów, oraz powinna rosnąć (tylko jak ?) z ilością możliwych podstanów w jednym sygnale. Okazuje się, że istnieje tylko jedna możliwość zdefiniowania, by ten warunek spełnić:

entropia E – oto jej definicja

$$\mathbf{E = const \quad N \log_2 K}$$

jeśli każdy z podstanów równie prawdopodobny, to $p = p_i = 1./K$,

$$\mathbf{E = const \quad (-N) \log_2 p}$$

jeśli p_i są różne to

$$\mathbf{E = const \ \Sigma_i (-N_i) \log_2 (p_i)}$$

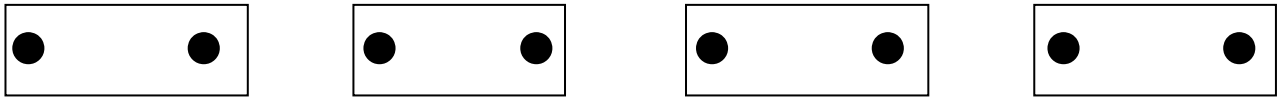
(jeśli const=1, to jest to ilość informacji w bitach)

Kompresja danych

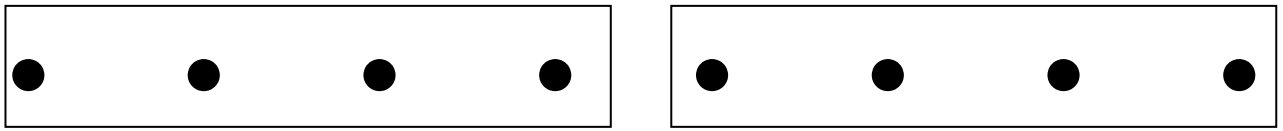
Ilość informacji nie może zależeć od tego, co zdefiniujemy za **sygnał**, a co za możliwy **podstan sygnału** (przyjęta definicja entropii czyli ilości informacji to zapewnia ! – sprawdzimy to na przykładzie przedstawionym na następnej folii)

Kompresja danych

Niech np. na każdym przerzutniku, symbolizowanym przez znak ● może pojawić się *stan wysoki* lub *stan niski*, z równym prawdopodobieństwem ½ ; przesłana wiadomość niech ma 4 sygnały zbudowane jak poniżej



lub równoważnie 2 sygnały (patrz poniżej)



(w obu powyższych przypadkach ilość informacji wynosi 8 bitów, czyli tyle samo!)

Kompresja danych

inna równoważna definicja entropii; jeśli mamy źródło które produkuje sygnały, każdy sygnał ma K możliwych podstanów, prawdopodobieństwo danego podstanu i wynosi **p(i)**, to entropia E (czyli ilość informacji) wynosi

$$E = - \text{const} \sum_i p(i) \log_2(p(i))$$

(w tym sumowaniu i przebiega od 1 do K; przyjmuje się const=1)

Kompresja danych

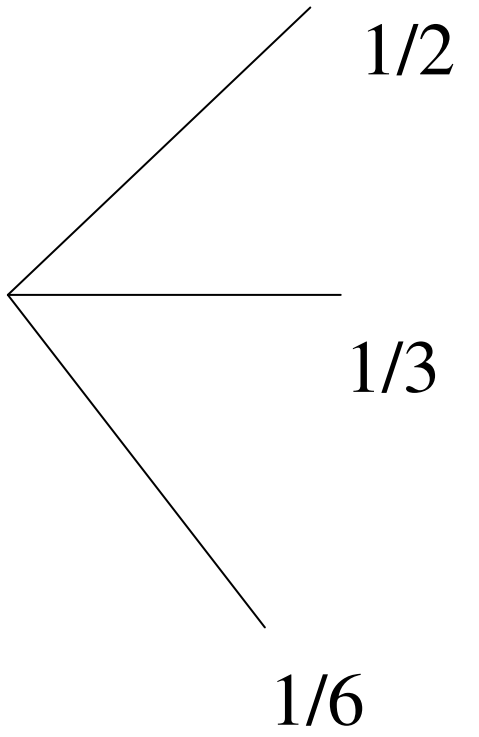
np. entropia w przypadku gdy jest jeden sygnał o dwóch możliwościach o prawdopodobieństwach p oraz (1-p)

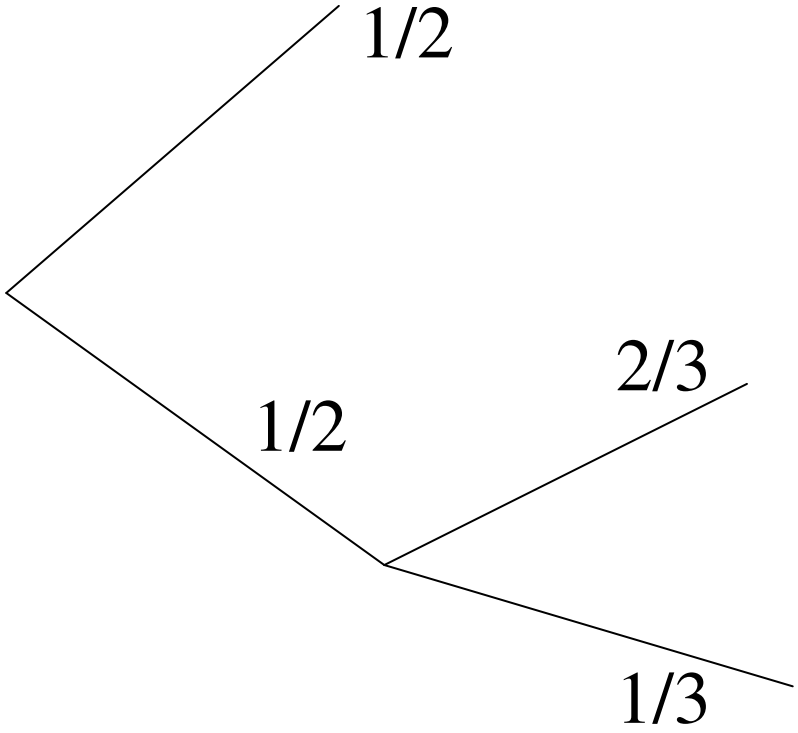
$$E = -(p \log p + (1-p) \log(1-p))$$

Jeśli p dąży do zera lub do jeden, to ilość informacji dąży do zera (E dąży do zera), jeśli o wielkości p nic nie wiadomo, to p=1/2, wtedy E=1 (jeden bit informacji).

Kompresja danych

np. ilość informacji powinna być taka sama dla poniższych dwóch przypadków rozkładów prawdopodobieństwa.





Kompresja danych

oznaczajac przez E ilość informacji, dla poprzedniego slajdu powinno być

$$E(1/2; 1/3; 1/6) = E(1/2; 1/2) + 1/2 * E(2/3; 1/3)$$

(wzór na E jest trzy folie wcześniej)

(więcej informacji pod adresem

http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf, np. na początku 6 rozdziału)

Kompresja danych

przykładowa entropia

drzewo Huffmana

kodowanie Huffmana jako przykład algorytmu kompresji

(przedstawiony przykład udowadnia, że kodowanie Huffmana wydajnie pakuje w sposób bezstratny dane)

poniżej adres strony: jak zbudować drzewo Huffmana

http://www.siggraph.org/education/materials/HyperGraph/video/mpeg/mpegfaq/huffman_tutorial.html

Kompresja danych

kolejny przykład: niech zródło informacji produkuje sekwencję liter A,B,C,D z prawdopodobieństwami 0.5, 0.25, 0.125, 0.125. Można policzyć, że na każdy przesyłany sygnał (tutaj – znak) ilość informacji wynosi 7/4 bitów. Tymczasem standartowo rozrzutnie używa się 8 bitów. Można zakodować(„drzewo Huffmana”)

A = 0 B = 10 C = 110 D = 111

W 1000 przesłanych znaków jest 500 A, 250 B, 125 C, 125 D. Źródło przeznaczyło na nie 1000 * 8 bitów. Liczona entropia daje 1000*7/4 = 1750 bitów. Po skompresowaniu zużywamy 500*1 + 250*2 + 125*3 + 125*3 = 1750 bitów. Czyli osiągnęliśmy nawet **granice kompresji ”jako sygnału”.**

Kompresja danych

kolejny przykład: niech zródło informacji produkuje sekwencję liter A,B,C,D z prawdopodobieństwami 0.5, 0.25, 0.125, 0.125. Można policzyć, że na każdy przesyłany sygnał (tutaj – znak) ilość informacji wynosi 7/4 bitów. Tymczasem standartowo rozrzutnie używa się 8 bitów. Można zakodować(„drzewo Huffmana”)

A = 0 B = 10 C = 110 D = 111

(jako ćwiczenie proszę przeliczyć, że traktując ten skompresowany zbiór jako zbiór zer oraz jedynek, stosując wzór na entropię pięć folii wstecz, otrzyma się również 1750 bitów!)

Kompresja danych

Poziom semantyczny informacji

Weźmy pod uwagę liczbe π

Ile bitów informacji należy przesłać, aby przesłać ją bezstratnie? nieskończenie wiele (traktując informację **na poziomie sygnałowym**)

A na poziomie semantycznym można przesłać,że jest to stosunek obwodu koła do jego średnicy, bezstratnie.

W teorii informacji nie znaleziono sposobu wyliczania ilości informacji ”merytorycznej”. Tym bardziej ilości informacji związanej z rozumowaniem logicznym.

Dygresja: rozumowanie logiczne

Czy liczbę ”pierwiastek z 2” da się zapisać jako ułamek, gdzie licznik i mianownik to liczby naturalne ? Udowadnia się, że nie można – ile jest informacji uzyskanej dzięki znajomości dowodu tej własności ?

Nie znamy sposobu wyliczenia ilości tej informacji.

Dygresja – czy tak by można kompresować?

mamy np. pewien tekst do przesłania, w języku angielskim

znajdujemy pewną liczbę 1.xxxxxxxxxx..... ,

przesyłamy do odbiornika definicję tej liczby

(teoretycznie taki sposób nie jest sprzeczny z semantycznym pojęciem informacji)

Kompresja danych – algorytm RLE

RLE – Run Length Encoding

Zalety: niskie zapotrzebowanie na moc obliczeniową, w wybranych sytuacjach algorytm najlepszy

Idea: zamień sekwencję danych na liczbę powtórzeń pewnych znaków

ABBBBBBBBBBCDEFFFFFF to chcemy skompresować

A$9BCDE$5F użyty został znak kontrolny $

Kompresja danych – algorytm RLE

Co zrobić jeśli w danych do skompresowania jest znak identyczny z wybranym znakiem kontrolnym?

Można np. zapisac $ jako $% , to już pozwoli na jednoznaczne zidentyfikowanie takiej sytuacji.

Kompresja danych – LZ77

pewną wadą kodowania Huffmana jest to, że potrzebne do kodowania drzewo może w przypadku ciągłej transmisji okazać się w pewnym momencie przestarzałe. Niestety zmieniać trzeba go w sposób rewolucyjny czyli trzeba zmienić całe drzewo.

Istnieją algorytmy adaptacyjne, pomyślane jako ”nadążające za sytuacją”.

LZ77 (Lempel-Ziv-1977) korzysta z bufora z podglądem oraz z okna przesuwnego.

Podstawową częścią algorytmu jest stałe wyszukiwanie w buforze najdłuższych zdań, które są już w oknie przesuwnym.

Typowe długości okna i bufora – odpowiednio kilka tysięcy bajtów i sto bajtów.

LZ77 (Lempel-Ziv-1977) to metoda kompresji na poziomie merytorycznym informacji (poszukiwanie zdań)

Kompresja danych – LZ77

okno 8b	dane (bufor 4b)	dane skompresowane
00000000	ABABCBABABCA	
*0000000*A	BABCBABABCA	A
*000000*AB	ABCBABABCA	AB
*000*ABABC	BABABCA	AB(6,2,C)
BABCBABA	BCA	AB(6,2,C)(4,3,A)
CBABABCA	(koniec danych)	AB(6,2,C)(4,3,A)(2,2,A)

(czyli okno, pełniące role słownika, nieustannie się modyfikuje)

Dekompresja danych – LZ77

okno 8b	dane skompresowane
00000000	AB(6,2,C)(4,3,A)(2,2,A)
*000000*AB	(6,2,C)(4,3,A)(2,2,A)
*000*ABABC	(4,3,A)(2,2,A)
BABCBABA	(2,2,A)
CBABABCA	(koniec danych skompresowanych)

(czyli okno, pełniące role słownika, nieustannie się modyfikuje)

LZ77 – zalety,wady

- przy dobrej implementacji wydajniejsze od kompresji uzyskiwanej kodowaniem Huffmana

- wolniejsze od kodowania Huffmana

- w miarę szybkie odkodowywanie

kodowanie arytmetyczne

-wykorzystanie ułamków bita

-Literatura:

http://www.zipworld.com.au/~isanta/uni/arithmetic.htm

http://www.cs.cf.ac.uk/Dave/Multimedia/node213.html

http://compressions.sourceforge.net/Arithmetic.html