Построяване на оптимални кодове

1. Ефективност на кода и степен на компресия

Ефективност на кода : Отношението на ентропията на източника I към цената на кода K за този източник **H(I)/L(K)**.

Очевидно 0&LTH(I)/L(K)<=1.

В компютърните компресиращи програми една по-практична оценка за тяхната ефективност е **степента на компресия**

Степен на компресия

$$(1-L_{compressed}/L_{uncompressed})*100\%$$

където $L_{uncompressed}$ и $L_{compressed}$ са съответно дължината на съобщението (*например файл*) преди и след компресия.

Понякога вместо горната величина се дава просто отношението **некомпресирана:компресирана** дължина (*Например: 10:1*). Това е по-удобно при големи степени на компресия - основно при компресия със загуби.

2. Метод на Шенон-Фано

Алгоритъмът на Шенон-Фано се базира на следното:

- За двубуквена азбука оптималния код е K={0,1} с цена L(K) = 1.
- Горната цена на кода ще бъде равна на ентропията на източника (*т.е. кодът ще има максимална ефективност*), ако източника е с вероятности на буквите P={0.5,0.5}.
  - *Вж. задача 1.*

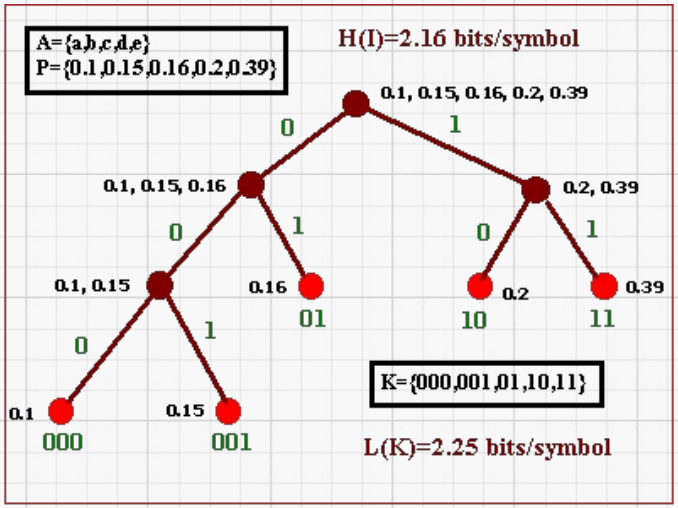
- Източник с n буквена азбука $A=\{a_1,a_2..a_j..a_n\}$ с вероятности $P=\{p_1,p_2..p_j..p_n\}$ може да бъде сведен до нов източник I_1 с двубуквена азбука $A_1=\{a_1$ **или** a_2 **или ..** a_j, a_{j+1} **или ..** $a_n\}$ и вероятности $P=\{p_1 + p_2 + .. p_j, p_{j+1} + .. p_n\}$.

$$\sum_{i=1}^{j} p_i \text{ и } \sum_{i=j+1}^{n} p_i \text{ са максимално близки до } 0.5, \text{т.е. ако } j \text{ е такова, че}$$

$$\text{минимизира} \left| \sum_{i=1}^{j} p_i - \sum_{i=j+1}^{n} p_i \right|$$

- Оптималният код за I_1 ще бъде максимално ефективен ако

Пример за кодиране по Шенон-Фано



⚠ Методът на Шенон-Фано не гарантира получаването на оптимален код.

---

3. Метод на Хафмън

Методът на Хафмън позволява получаване на **оптимален** код за зададен източник. Той се базира на следните две теореми:

Ако I е n буквен източник с вероятности

$$p_1>=p_2>=..p_j>=..>=p_n$$

то съществува **оптимален префиксен** код с дължини на кодовите думи

$$l_1<=l_2<=..l_j<=..<=l_n$$

като последните две кодови думи имат вида a0 и a1 (*а е двоична поредица*).

Ако K е оптимален код с кодови думи $\{a_1,a_2,...a_n\}$ за **n** буквен източник с вероятности

$$p_1>=p_2>=..p_i>=..>=p_n$$

то кодът K' с кодови думи

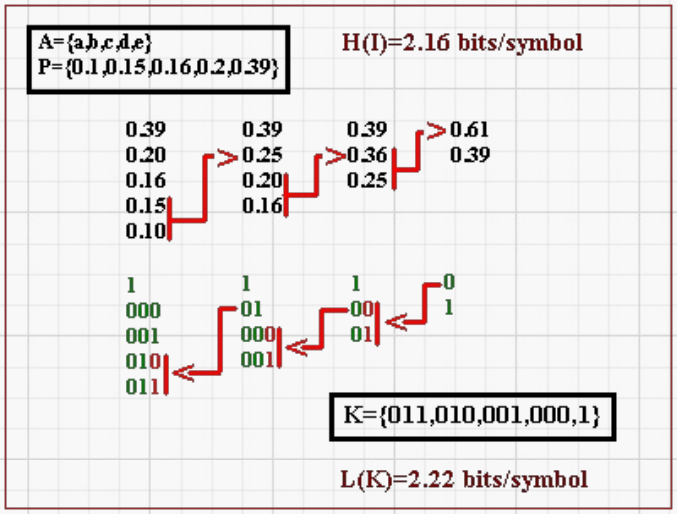
$$a_1, a_2,... a_{i-1}, a_{i+1},..., a_i0, a_i1$$

е оптимален за **n+1** буквен източник с вероятности

$$p_1>=p_2>=..p_{i-1}>=,p_{i+1} >= ..>= p_n >= q >= s$$

където $q+s=p_i$.

Пример за кодиране по Хафмън



4. Предимства и недостатъци на побуквеното кодиране.

- **Предимства**
  - Кодирането е без загуба на информация.
  - Ако източника е известен, то съществува алгоритъм за построяване на оптимален за този източник код

(Хафмън).
  - Кодирането и декодирането са прости и бързи алгоритми, които лесно се реализират софтуерно и хардуерно.
- **Недостатъци**
  - Кодирането се базира на предварително зададен източник (**статичен модел**). "Построяването" на източника (модела) е извън алгоритъма на Хафмън.

> ***Пример*** *Модел на английската азбука с фиксирани вероятности, използван като източник за а) литературен текст. b) текст на Pascal.*

  - Кодирането може да е далече от оптималното ако първичната азбука на източника е с малък брой букви.

> ***Пример*** *При предаване по факс. Азбука от две букви: {черно, бяло}. Цена на оптималния код е 1 бит/символ. Ентропия на източника е* $H=p \log_2 p+(1-p) \log_2 (1-p) <= 1$

  - Алгоритъма на Хафмън е за т.н. **модел от 0-лев порядък** (*отчитат се само вероятностите на буквите на първичната азбука* p_i, *но не и 'взаимните' вероятности* p_{ij}, p_{ijk} *и т.н*).

- **Преодоляване на недостатъците**
  - Използване на **динамични модели**.

> ***Пример*** *Оценка на вероятностите от честотата на срещане на символите в самото съобщение.* *За достатъчно дълги съобщения.* ***Недостатък: трябва да се предават и вероятностите.***

  - Използване на **модели от по-висок порядък** . Трудности: 1. Броят на отчитаните вероятностите расте много бързо. 2. Построяването на адекватен модел е трудно.

Пример *Параметри на модели от различен порядък:*

Порядък на модела	**Отчитани вероятности на буквите**	**Брой вероятности**	**Брой вероятности** Пример за азбука с 30 букви
-1	$p=1/n$	1	1
0	$p_1,p_2..p_n$	n	30
1	$p_1, p_2..p_n$ $p_{11}, p_{21}..p_{n1}$ $p_{12}, p_{22}..p_{n2}$ ..... $p_{1n}, p_{2n}..p_{nn}$	$n+n^2$	930
2	$p_1,p_2..p_n$ $p_{11}, p_{21}..p_{n1}$ $p_{12}, p_{22}..p_{n2}$ ..... $p_{1n}, p_{2n}..p_{nn}$ ..... $p_{111}, p_{121}..p_{1n1}$	$n+n^2+n^3$	27930

	.....		

- Използване на кодиране, което **не е побуквено**.
- Използване на **компресия със загуби** (*ако е допустимо*).
- Комбинация на различни методи и подходи.

4. Класификация на методите за компресия

- **Според начина на възстановяване на първичното съобщение:**
  - Компресия без загуби
  - Компресия със загуби
- **Според начина на кодиране:**
  - Побуквено кодиране
  - Непобуквено кодиране - конволюционни и др. методи (*общото - еднакви участъци от съобщението могат да се кодират по различен начин в зависимост от предисторията*).
- **Според модела на източника**
  - Фиксиран модел.
  - Динамичен модел.
  - Порядъка на модела (от 0-ев, 1-ви и т.н. порядък).

6. Методи за компресия без загуби.

- **Хафмън** (вж. по-горе)
- **Run length encoding (RLE)**.

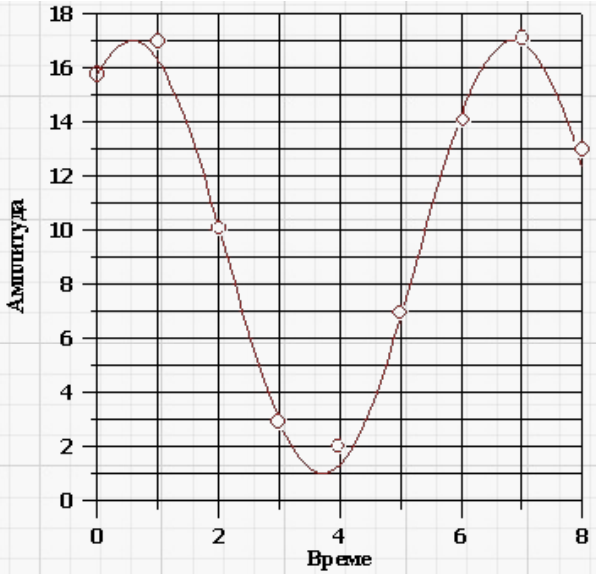
Пример
първично съобщение: 00001111111100000101011011111111
кодирано съобщение: #40#81#50!1010110#71
 # - мета символ : #nx - n пъти символ x
! - мета символ : следва некомпресирана част.

Предимства: много бързо. Пример за използване: ECP port.

- **Relative encoding (Δ кодиране)**

Алгоритъм
Първи кодиран байт = Първи байт от съобщението.
Всеки следващ кодиран байт = Текущия байт от съобщението - Предишния байт от съобщението.

Пример



Време	Извадка	Кодирано
0	16	16
1	17	+1
2	10	-6
3	3	-7
4	2	-1
5	7	+5
6	14	+7
7	17	+3
8	13	-4

Развитие - разлики от по-висок порядък.
Примери за приложение: ADC & DAC; звук; научен експеримент (аналогови сигнали).

- **Алгоритми на базата на речници**

  Най-известни - **LZ (Лемпел-Зиф)** алгоритми.

  Пример

  Входно съобщение: **ABRACADABRA**
  ABRACADABRA
  Кодирано съобщение: **ABRACAD*74***
  74 е препратка (цитиране) 7 символа назад, 4 символа дължина

- **Аритметично кодиране**

  Пример

Буква	Вероятност	Интервал
Space	0.1	0.0 - 0.1
A	0.1	0.1 - 0.2
B	0.1	0.2 - 0.3
E	0.1	0.3 - 0.4
G	0.1	0.4 - 0.5
I	0.1	0.5 - 0.6
L	0.2	0.6 - 0.8
S	0.1	0.8 - 0.9
T	0.1	0.9 - 1.0

Вероятности на буквите в съобщението BILL GATES

Следваща буква	Долна граница	Горна граница
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
Space	0.2572	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Кодиране на съобщението BILL GATES

Число	Буква	Граници		Интервал
0.2572167752	B	0.2	0.3	0.1
0.572167752	I	0.5	0.6	0.1
0.72167752	L	0.6	0.8	0.2
0.6083876	L	0.6	0.8	0.2
0.041938	Space	0.0	0.1	0.1
0.41938	G	0.4	0.5	0.1
0.1938	A	0.2	0.3	0.1
0.938	T	0.9	1.0	0.1
0.38	E	0.3	0.4	0.1
0.8	S	0.8	0.9	0.1
0.0				

Декодиране на съобщението BILL GATES

- **Математически трансформации - Фурие и др.**

7. Методи за компресия със загуби.

Къде е допустимо?: Информация, свързана с човешки възприятия (5-те сетива).

Пример *Звук: 20 Hz - 20 kHz, но най-голяма чувствителност в средната честотна област и нормален интензитет.*

Методи:

- Нелинейни трансформации.
- FFT + филтрации + IFFT.
- Фрактални.
- "Алгоритмично" кодиране. *Пример:* MIDI (vocoder)

8. Смесени методи за компресия.

Смесени = компресия със загуби + без загуби.

Пример
MP3 (Звук):

- ***Нелинейно AD преобразуване***
- ***Joint stereo***
- ***"Резервоар" - участъци за компресия без загуби***
- ***Хафмън***

9. Граници за компресия???

- *Алгоритмична сложност*
- *Некомпресируемост*