Information Compression

مروری بر نظریه‌ی اطلاعات

کدگذاری بی‌اتلاف

فشرده‌سازی اطلاعات

۴۰-۱۰-۰۰۲-۰۱

بخش نخست

دانشگاه شهید بهشتی
پژوهشکده‌ی فضای مجازی
زمستان ۱۴۰۰
احمد محمودی ازناوه

فهرست مطالب

- انواع فشرده‌سازی
- اندازه‌گیری اطلاعات
- مفهوم آنتروپی
- نیازمندی‌های یک کدگذار
- نامساوی Kraft-McMillan
- کدهای پیشوندی
- کدگذاری Shannon–Fano
- کدگذاری Huffman
- کدگذاری Unary
- کدگذاری Golomb
- کدگذاری محاسباتی

انواع فشرده‌سازی

فشرده‌سازی

بی‌اتلاف

Lossless
noiseless coding
entropy coding

داده‌ی فشرده‌شده پس از بازیابی کاملا با داده‌ی اصلی یکسان است.

فشرده‌سازی متن
تصاویر پزشکی-تصاویر نظامی

بااتلاف

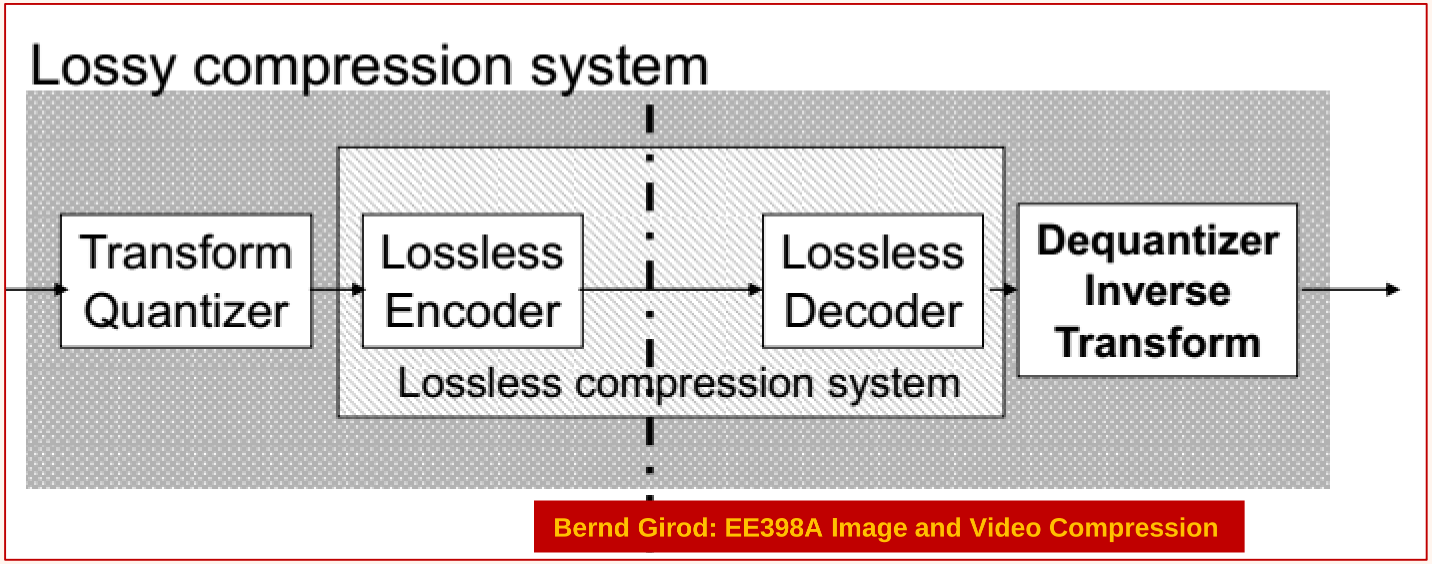
Lossy

داده‌ی فشرده‌شده پس از بازیابی با داده‌ی اصلی یکسان نیست!

نرخ فشردگی بالا

فشرده‌سازی بی‌اتلاف

- تقریبا در بطن همه‌ی سیستم‌های فشرده‌سازی «بااتلاف» یک سیستم «بی‌اتلاف» قرار دارد:



- از این رو ابتدا به بررسی سیستم‌های «فشرده‌سازی بی‌اتلاف» می‌پردازیم.

تا چه میزان امکان فشرده‌سازی بی‌اتلاف وجود دارد؟

اندازه‌گیری اطلاعات

- اگر تعدادی نماد در دست باشد، به وسیله‌ی رابطه‌ی *Shannon* کمینه‌ی تعداد بیت‌ها برای ارسال نمادهای مورد نظر محاسبه می‌گردد.
- اگر M تعداد نمادها باشد:

$$Ent = -\sum_{i=1}^{M} \rho_i \log_2^{\rho_i} \qquad 0 \le \rho_i \le 1$$

$$Avrage \;\; Numbit = Ent$$

$$0 \le Ent \le \log_2^{M}$$

- اگر داشته باشیم: *AAAAABBCDE*

A = 0.5 B = 0.2 C = 0.1 D = 0.1 E = 0.1

$$\text{Ent} = -[(0.5\log_2^{0.5} + 0.2\log_2^{0.2} + (0.1\log_2^{0.1}) \times 3)]$$

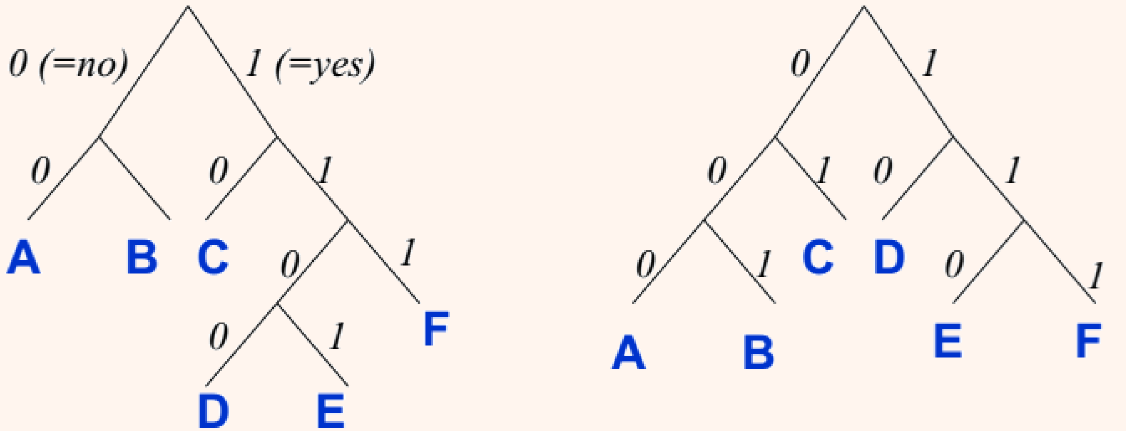
$$\text{Ent} = -[-0.5 + (-0.46438) + (-0.9965)]$$

$$\text{Ent} = -[-1.9] = 1.9$$

یک مثال ساده

- یک مسابقه‌ی «بیست سؤالی» را در نظر بگیرید:
- یک خروجی از یک مجموعه‌ی محدود در نظر گرفته می‌شود: $\{A, B, C, D, E, F\}$
- کدام استراتژی برای رسیدن به پاسخ بهتر است؟

Bernd Girod: EE398A Image and Video Compression



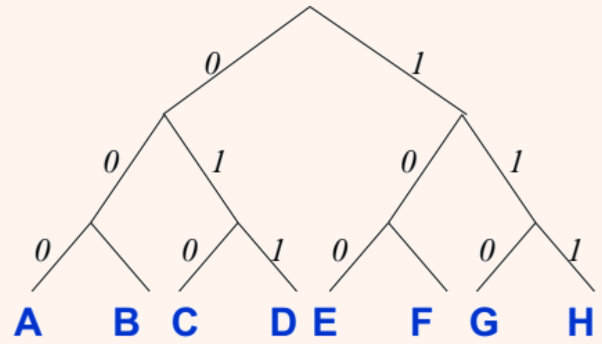
نحوه‌ی انتخاب با یک رشته بیتی قابل نمایش است.

http://dsw.users.sonic.net/entropy.html

نمایش با طول ثابت

طول متوسط این رشته‌ی **بیتی** توصیف‌کننده برای *M* نماد:

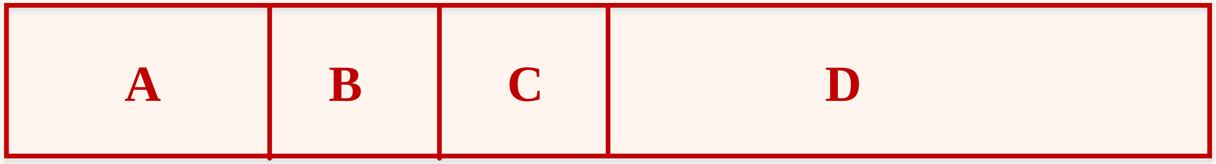
$$l_{av} = \log_2 M$$



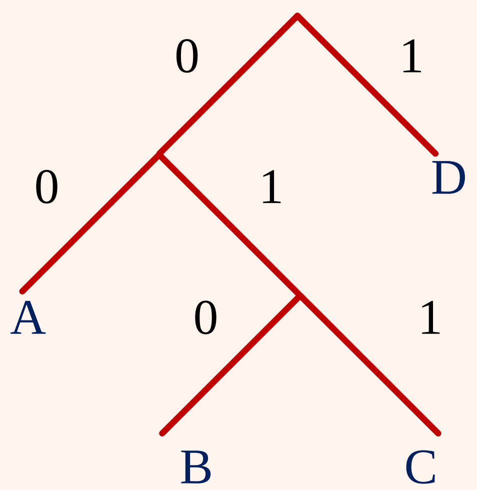
در صورتی که احتمال وقوع هر نماد *1/M* باشد، این شیوه‌ی کدگذاری بهینه است.

نمایش با طول متغیر

- در صورتی که احتمال انتخاب نمادها یکسان نباشد:



Binary Code Trees



  - **بهتر است** نمادهایی که احتمال رخداد بالاتری دارند، با رشته‌ی کوتاه‌تری توصیف شوند و بالعکس

کدهای با طول متغیر

- با افزودن یک بیت، میزان اطلاعات قابل نمایش دو برابر می‌شود(رابطه‌ی نمایی).
- اگر میزان اطلاعات یک متغیر تصادفی دو برابر شود(احتمال وقوع نصف شود)، به یک بیت بیشتر برای نمایش احتیاج دارد(رابطه‌ی لگاریتمی).
- در این حالت تعداد بیت برای نمایش یک نماد با احتمال وقوع p:

$$\log_2\left(\frac{1}{p}\right)$$

آنتروپی یک متغیر تصادفی

- فرض کنیم یک متغیر تصادفی شامل تعداد محدودی الفبا باشد:

$$A_x = \{\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{M-1}\}$$

$$f_X(x) = P(X = x); PMF$$

- اطلاعات متناسب با X=x

$$h_X(x) = -\log_2 f_X(x)$$

واحد اندازه‌گیری: بیت

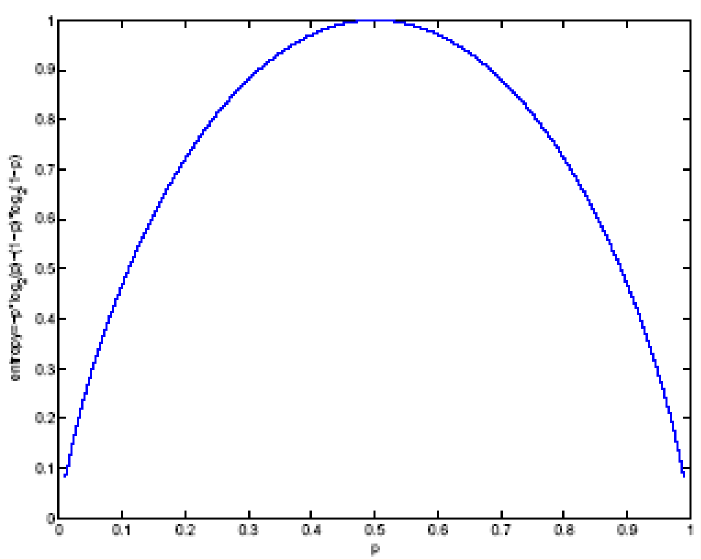
- آنتروپی، مقدار متوسط(امید ریاضی) اطلاعات نمادهاست:

$$H(X) = E[h_X(X)] = -\sum_{x \in A_x} f_X(x) \log_2 f_X(x)$$

آنتروپی برای دو دسته

– آنتروپی اطلاعات: رخداد اتفاقی تا چه حد تصادفی است.

$$H\left(X\right)=-p\log_2 p-\left(1-p\right)\log_2\left(1-p\right)$$



$$\rho = R - H\left(X\right) \geq 0; \quad R: \text{aveage length code}$$

میزان افزونگی

Variable length codes کدهای با طول متغیر

- با توجه به این‌که اطلاعات موجود یا نمادهای تولید شده در یک پیام دارای احتمال وقوع یکسان نیستند، می‌توان اطلاعات با احتمال وقوع کم‌تر را با تعداد بیت بیشتر و اطلاعات با احتمال وقوع بیشتر را با تعداد بیت کم‌تر کد کرد.

Coding

کدگذاری

Alphabet

Letter

- مجموعه نمادهایی (حروف) که باید ذخیره(ارسال) شوند، «الفبا» نامیده می‌شود.
- به رشته‌ی بیتی که به هر «نماد» (حرف) نسبت داده می‌شود، «کلمه‌کد» یا «کدواژه» گفته می‌شود.

Code word

  - هر کلمه‌کد دارای معنی واحد می باشد و طبق یک قاعده خاص با یک کد متناظر می باشد.

Code

- به مجموعه‌ی «کدواژه‌ها»، «کد» گفته می‌شود.
- به فرآیند انتساب «کدواژه‌ها» به حروف الفبا «کدگذاری» گفته می‌شود.

نیازمندی‌های یک کدگذار

- یک کد خوب باید شرایط زیر را داشته باشد:

Non-singular

کد اختصاص یافته به هر نماد یکتا باشد.

Uniquely decodable

دنباله‌ی نمادهای کد شده با هر ترتیبی قابل کدگشایی باشد.

Instantaneously decodable

دنباله‌ی نمادهای کد شده دریافت شده بدون توجه به مابقی داده‌ها (هنوز دیده نشده) قابل کدگشایی باشد.

مثال ۱

کدگذاری نمادها

α_i	Code word
α_0	0
α_1	01
α_2	10
α_3	11

$$\boxed{\alpha_0,\alpha_2}\ \alpha_3,\alpha_0,\alpha_1$$

$$\boxed{0\ 10}\ 11\ 0\ 0\ 1$$

$$\boxed{\alpha_1,\alpha_0}\ \alpha_3,\alpha_0,\alpha_1$$

$$\boxed{01\ 0}\ 11\ 0\ 0\ 1$$

کدگشایی به صورت یکتا ممکن نیست!

مثال ۲

α_i	$p(\alpha_i)$	Code A	Code B	Code C	Code D	Code E
α_0	0.5	0	0	0	00	0
α_1	0.25	10	01	01	01	10
α_2	0.125	11	010	011	10	110
α_3	0.125	11	011	111	110	111
	l_{av}	1.5	1.75	1.75	2.125	1.75
non-singular		✗	✓	✓	✓	✓
uniquely decodable			✗	✓	✓	✓
instantaneously decodable				✗	✓	✓

تمرین

کد یک

α_i	Code word
α_0	0
α_1	10
α_2	110
α_3	111

کد دو

α_i	Code word
α_0	0
α_1	01
α_2	100
α_3	011

با توجه به کدهای فوق رشته‌ی بیتی زیر را کدگشایی کنید.

0 0 1 1 0 1 0 1 1 0 1 0 0

کد یک: 0 | 0 | 1 1 0 | 1 0 | 1 1 0 | 1 0 | 0 → $\alpha_0\ \alpha_0\ \alpha_2\ \alpha_1\ \alpha_2\ \alpha_1\ \alpha_0$

کد دو: 0 | 0 1 1 | 0 1 | 0 1 1 | 0 | 1 0 0 → $\alpha_0\ \alpha_3\ \alpha_1\ \alpha_3\ \alpha_0\ \alpha_2$

$\alpha_0\ \alpha_3\ \alpha_1\ \alpha_3\ \alpha_1\ \alpha_0\ \alpha_0$

به صورت یکتا کدگشایی نمی‌شود!

در صورتی که یک کلمه‌کد پیشوند دیگری باشد، به پسوند کلمه کد دوم «پسوند معلق» می‌گویند.

Dangling Suffix

شیوه‌ی تشخیص قابلیت کدگشایی

Sardinas–Patterson algorithm(1953)

- یک لیست از کلمه‌کدها ایجاد کنید.
- کلمه‌کدها را دوبه‌دو بررسی کنید که آیا یکی پیشوند دیگری است؟
  - در این صورت، پسوند معلق را به لیست بیفزایید(چنان‌چه پیش از این اضافه نکرده باشید).
  - این فرآیند را تا زمانی ادامه دهید که:
    - به یک پسوند معلق برسید که کلمه‌کد باشد.
    - پسوند معلق دیگری نباشد.

قابل کدگشایی نیست.

به صورت یکتا قابل کدگشایی است.

Sardinas, A. A. and G. W. Patterson (1953). A necessary and sufficient condition for unique decomposition of coded messages. PROCEEDINGS OF THE INSTITUTE OF RADIO ENGINEERS.

مثال

$$\{\boxed{0},\boxed{0}\boxed{1},11\}$$

$$\{0,01,11,\boxed{1}\}$$

قابل کدگشایی به صورت یکتاست!

$$\{\boxed{0},\boxed{0}\boxed{1},10\}$$

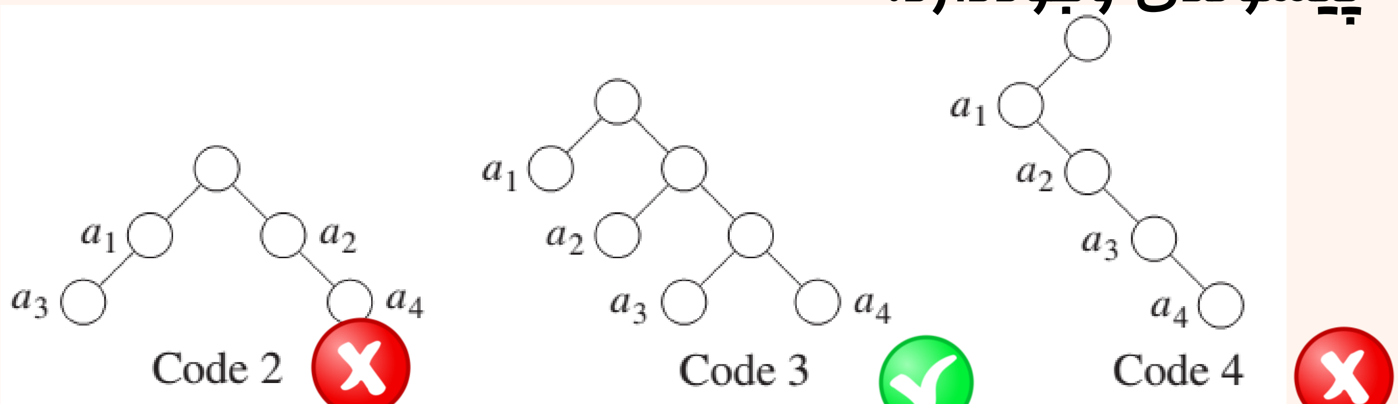
$$\{\boxed{0},01,\boxed{1}\boxed{0},\boxed{1}\}$$

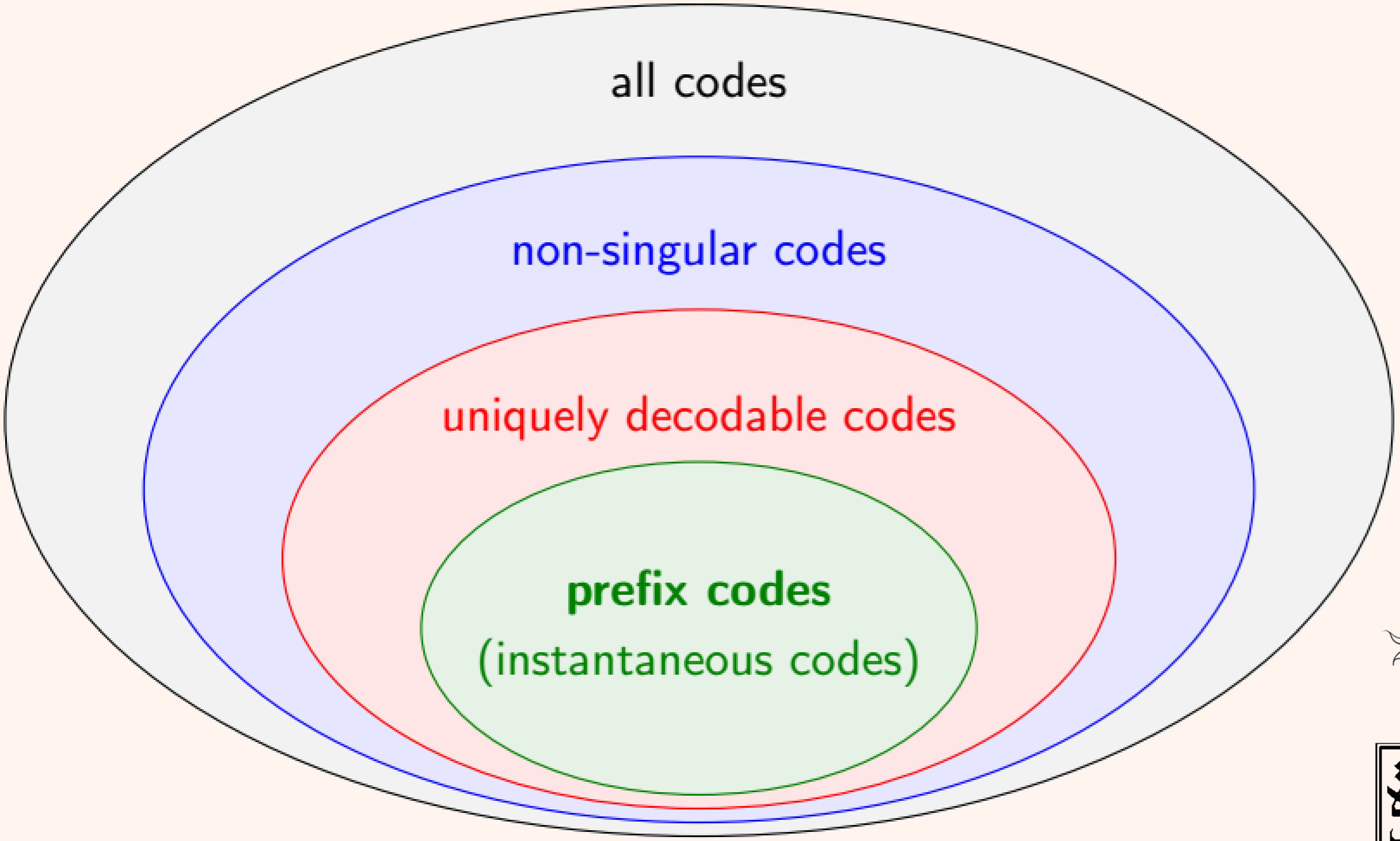
پسوند معلق یک کلمه‌کد است، درنتیجه قابل کدگشایی نیست.

Prefix code

کدهای پیشوندی

- به کدی که هیچ کلمه‌کدی از آن پیشوند دیگری نباشد، «کد پیشوندی» می‌گویند.
  - با توجه نحوه‌ی تشخیص قابلیت کدگشایی، چنین کدی همواره قابل کدگشایی خواهد بود.
- به ازای هر کد پیشوندی یک درخت دودویی می‌توان رسم کرد که به هر برگ یک کلمه‌کد نسبت داده شده است.
- پیشوندی بودن کد باعث ایجاد هیچ محدودیتی نمی‌شود. ثابت می‌شود به ازای هر کد قابل کدگشایی، یک کد پیشوندی وجوددارد.




all codes
non-singular codes
uniquely decodable codes
prefix codes
(instantaneous codes)

Kraft-McMillan inequality

قضیه ۱

- شرط لازم برای این که یک کد به صورت یکتا قابل کدگشایی باشد:

$$\sum_{i=0}^{M-1} 2^{-l(\alpha_i)} \leq 1$$

اثبات در Sayood, K. (2012) آمده است.

قضیه ۲

- چنان‌چه یک کد قابل کدگشایی باشد(در شرایط فوق صدق کند)، یک کد پیشوندی با کلمه‌کد با اندازه‌های مورد نظر وجود خواهد داشت.

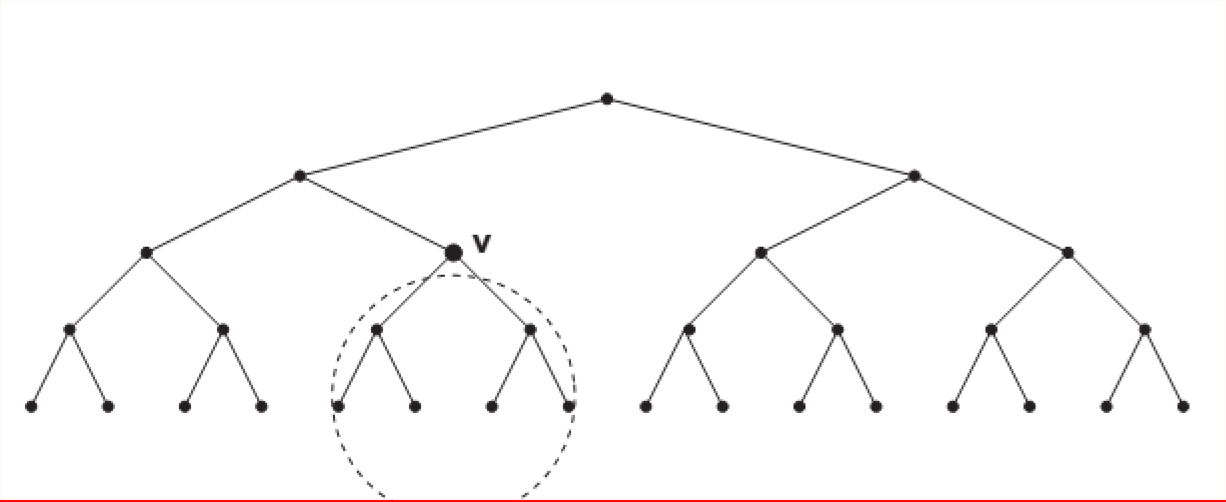
$$l_0 \leq l_1 \leq \ldots \leq l_{M-1}$$

اندازه‌ی کلمه‌کدها

$$\sum_{i=0}^{M-1} 2^{-l_i} \leq 1$$

یک درخت دودویی کامل با عمق l ایجاد می‌کنیم:

$$l = \max\{l_0, l_1, \ldots, l_{M-1}\}$$



در صورت حذف یک زیردرخت از عمق k، 2^{l-k} گره‌ی برگ حذف می‌شوند.

مجموع برگ‌هایی که برای تشکیل کد باید حذف شوند:

$$\sum_{i=0}^{M-1} 2^{l-l_i} = 2^l \sum_{i=0}^{M-1} 2^{-l_i} \leq 2^l$$

در نتیجه تعداد گره‌های برگ برای ساخت چنین کدی کفایت می‌کند ✦

Shannon–Fano coding

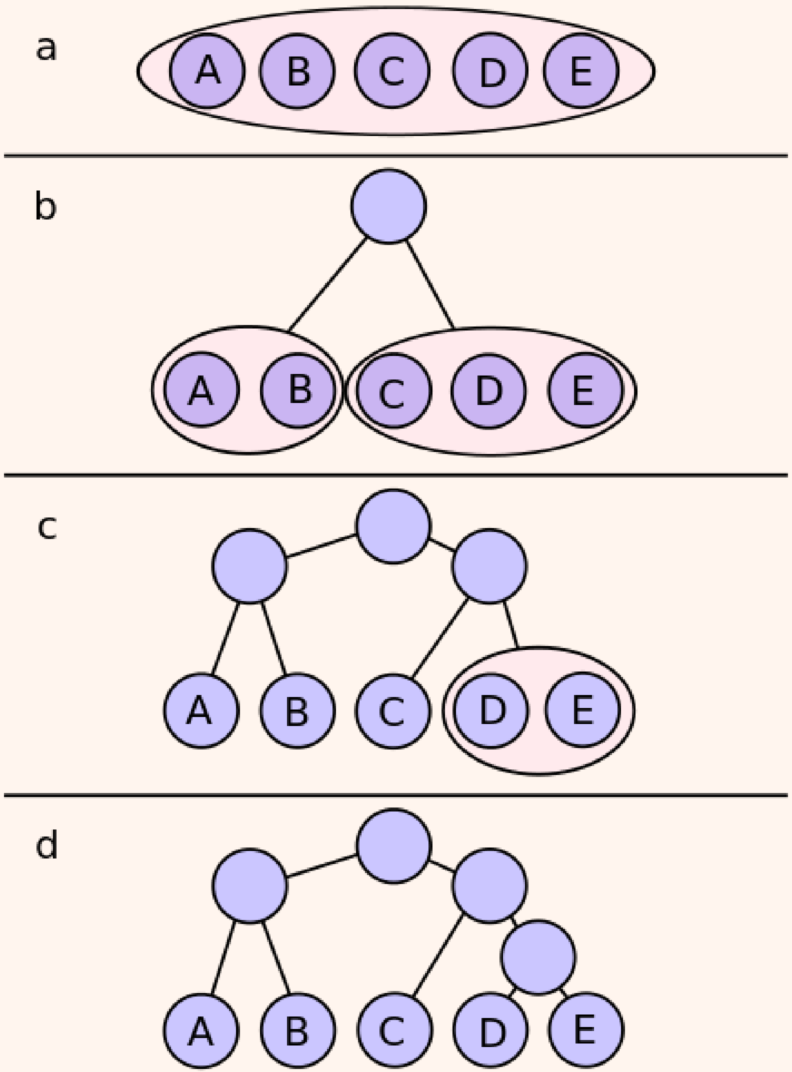
- در این شیوه بر اساس احتمال وقوع نمادها پس از مرتب کردن به دو دسته‌ی تقریبا با احتمال مساوی تقسیم می‌شوند.
  - به یک دسته بیت صفر و به دسته‌ی دیگر بیت یک اختصاص می‌یابد.
- این تقسیم‌بندی به همین شیوه ادامه می‌یابد.
- این شیوه‌ی کدگذاری بهینه نیست.

a_i	$p(a_i)$	1	2	3	4	Code
a_1	0.36	0	00			00
a_2	0.18		01			01
a_3	0.18	1	10			10
a_4	0.12		11	110		110
a_5	0.09			111	1110	1110
a_6	0.07				1111	1111

wikipedia

Fano, R. M. (1949). The transmission of information, Massachusetts Institute of Technology, Research Laboratory of Electronics Cambridge, Mass, USA.

Shannon–Fano coding



- در این شیوه (بر خلاف روش Huffman) درخت از سمت ریشه رشد می‌کند.

wikipedia

کدگذاری Huffman

- در سال ۱۹۵۲ توسط David Huffman مطرح شد.
- این شیوه prefix است و متوسط طول کد در آن بهینه است.
- در این شیوه ابتدا اصولی برای کدهای بهینه مطرح و سپس بر اساس آن شیوه‌ی کدگذاری به دست می‌آید:
  - کد اختصاص یافته به هر نماد یکتاست.
  - اطلاعات با احتمال وقوع کم‌تر را با تعداد بیت بیشتر و اطلاعات با احتمال وقوع بیشتر را با تعداد بیت کم‌تر کد شوند.
  - دنباله‌ی نمادهای کد شده دریافت شده بدون توجه به مابقی داده‌ها (هنوز دیده نشده) قابل کدگشایی باشد.
  - دو نماد با کم‌ترین احتمال با طول یکسان کد می‌شوند.
  - کلمه‌کدهای دو نماد با کم‌ترین احتمال تنها دربیت آخر متفاوت هستند.

$$P(1) \geq P(2) \geq \cdots \geq P(N-1) \geq P(N)$$
$$L(1) \leq L(2) \leq \cdots \leq L(N-1) = L(N)$$

"A Method for the Construction of Minimum-Redundancy Codes"

مثال

Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a_4	0.1	$c(a_4)$
a_5	0.1	$c(a_5)$

- دو نماد α_4 و α_5 کم‌ترین احتمال وقوع را دارند، پس تنها در یک بیت اختلاف خواهند داشت:

$$c(a_4) = \alpha_1 * 0$$
$$c(a_5) = \alpha_1 * 1$$

Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a'_4	0.2	α_1

- به همین ترتیب خواهیم داشت:

$$c(a_4) = \alpha_2 * 10$$
$$c(a_5) = \alpha_2 * 11$$

$$\alpha_1 = \alpha_2 * 1$$

$$c(a_3) = \alpha_2 * 0$$
$$c(a'_4) = \alpha_2 * 1$$

مثال(ا

Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a_3'	0.4	α_2
a_1	0.2	$c(a_1)$

$$c(a_3') = \alpha_3 * 0$$
$$c(a_1) = \alpha_3 * 1$$

$$c(a_3) = \alpha_3 * 00$$
$$c(a_4) = \alpha_3 * 010$$
$$c(a_5) = \alpha_3 * 011$$

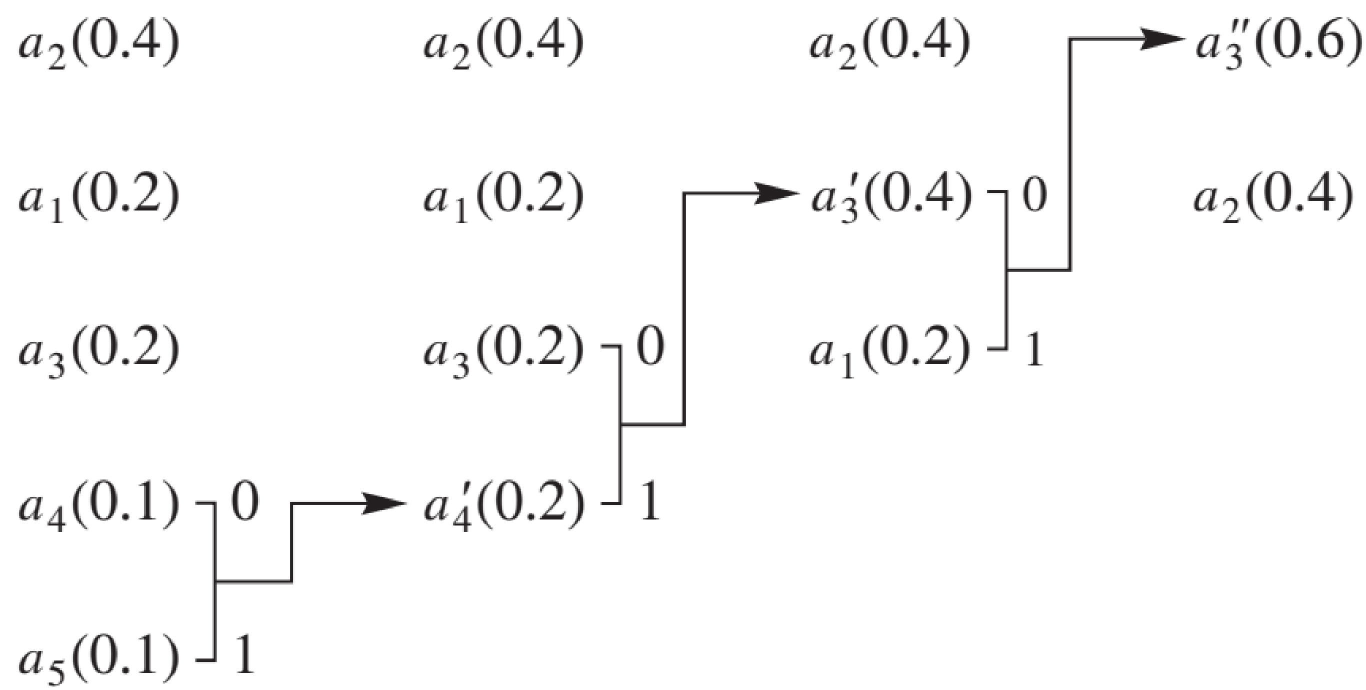
Letter	Probability	Codeword
a_3''	0.6	α_3
a_2	0.4	$c(a_2)$

$$c(a_3'') = 0$$
$$c(a_2) = 1$$

$$\alpha_2 = \alpha_3 * 0$$

Letter	Probability	Codeword
a_2	0.4	1
a_1	0.2	01
a_3	0.2	000
a_4	0.1	0010
a_5	0.1	0011

مثال(ادامه...)

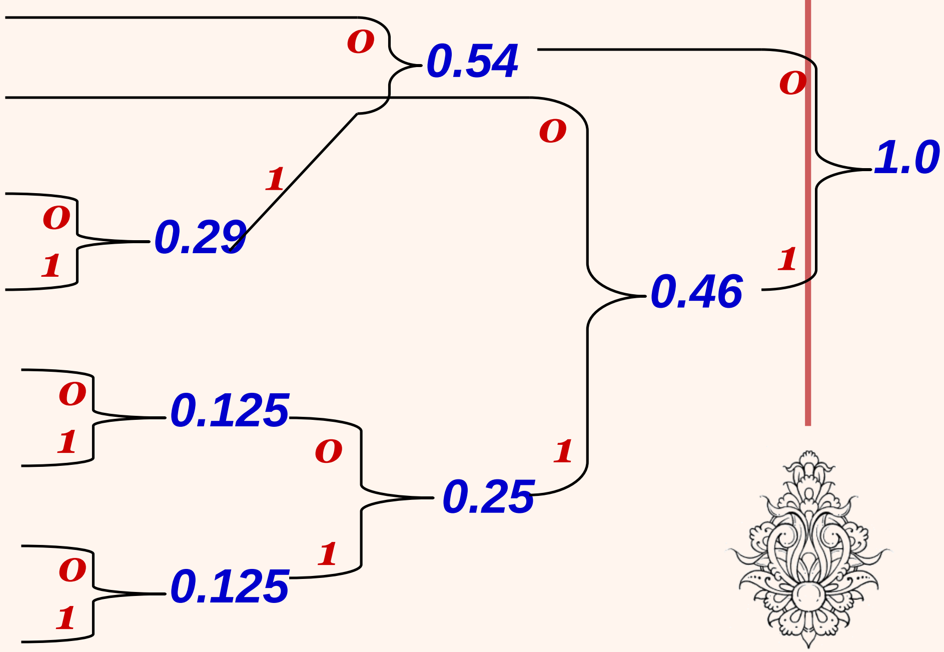


$$l = .4 \times 1 + .2 \times 2 + .2 \times 3 + .1 \times 4 + .1 \times 4 = 2.2 \text{ bits/symbol}$$

the redundancy is 0.078 bits/symbol.

مثال ۲

Binary	Huffman		
000	00	S0	0.25
001	10	S1	0.21
010	010	S2	0.15
011	011	S3	0.14
100	1100	S4	0.0625
101	1101	S5	0.0625
110	1110	S6	0.0625
111	1111	S7	0.0625

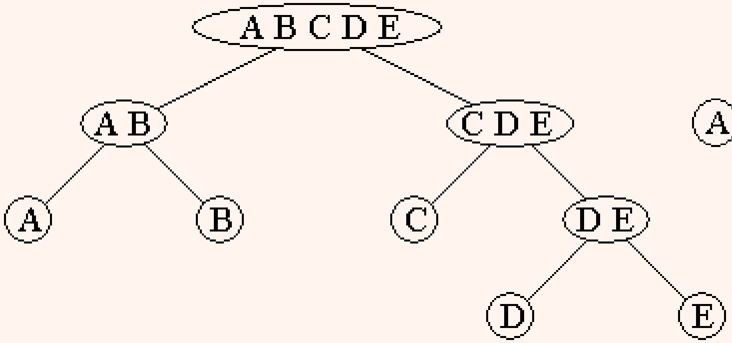


(trace from root)

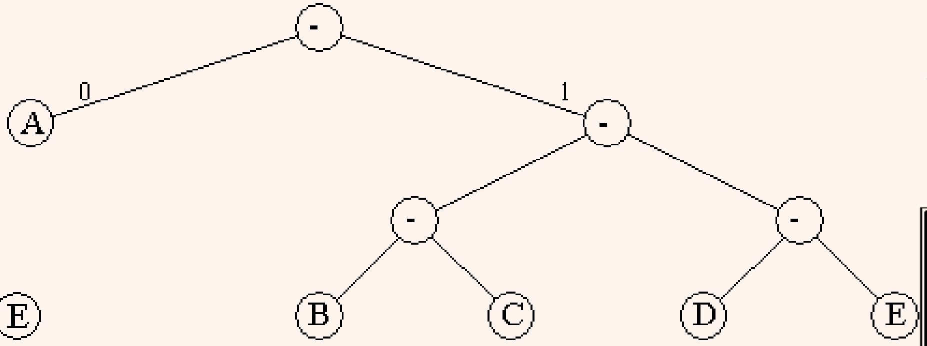
تمرین

- جدول نمادهای زیر را به دو روش Huffman و Shannon-Fano کد کنید.

α_i	frequency
A	24
B	12
C	10
D	8
E	8



Shannon-Fano



Huffman

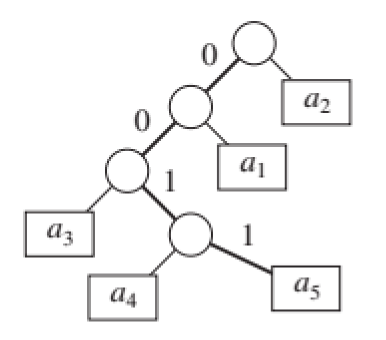
Minimum Variance Huffman Codes

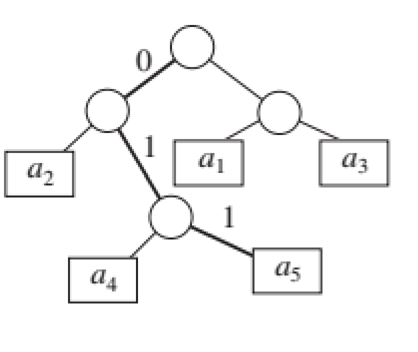
Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a'_4	0.2	α_1

Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a'_4	0.2	α_1
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$

Letter	Probability	Codeword
a_2	0.4	1
a_1	0.2	01
a_3	0.2	000
a_4	0.1	0010
a_5	0.1	0011

Letter	Probability	Codeword
a_1	0.2	10
a_2	0.4	00
a_3	0.2	11
a_4	0.1	010
a_5	0.1	011





Minimum Variance Huffman Codes

- با تغییر در ترتیب اجرای الگوریتم Huffman کدهای متفاوتی به دست می‌آید، بسته به این که نمادها در هر مرحله چگونه مرتب شوند.
- هرچند متوسط نرخ بیت در دو حالت با هم تفاوتی ندارد، اما واریانس طول کلمه‌کدها متفاوت خواهد بود.
- در بیشتر کاربردها با وجود این که می‌توان از کدهای با طول متغیر استفاده کرد، ولی پهنای باند ثابت است.
  - به عنوان مثال در صورتی که ما قصد داشته باشیم ۱۰،۰۰۰ نماد در ثانیه ارسال کنیم و پهنای باند ۲۲،۰۰۰bps باشد، امکان ارسال بیش از ۲۲،۰۰۰ بیت بر ثانیه وجود نخواهد داشت.

Minimum Variance Huffman Codes

- چنان‌چه در یک بازه‌ی کوتاه دو نماد α_4 و α_5 پیوسته ارسال شود:
  - برای کد اول، داده‌ی ارسالی حدود ۴۰،۰۰۰bps خواهد بود(۱۸۰۰۰ بیت باید بافر شوند).
  - برای کد دوم، داده‌ی ارسالی حدود ۳۰،۰۰۰bps خواهد بود(۸۰۰۰ بیت باید بافر شوند).
  - در صورتی که به جای این دو نماد از α_2 استفاده شود، داده‌ی ارسالی در کد اول نرخ ۱۰،۰۰۰ بیت بر ثانیه خواهد داشت.
- در همه‌ی این حالات گیرنده انتظار دریافت ۲۲،۰۰۰ بیت بر ثانیه را خواهد داشت.

برای به دست آوردن کد Huffman با کم‌ترین واریانس، حرف ترکیبی در مرتب‌سازی بالاتر از حروف تکی با احتمال یکسان قرار می‌گیرند.

- در صورتی که خصوصیات آماری داده در حال تغییر باشد، برای کدگذاری، هر بار کلمه‌کدها باید محاسبه شده و همراه با داده‌ها ارسال(ذخیره) گردد:
  - مثال:

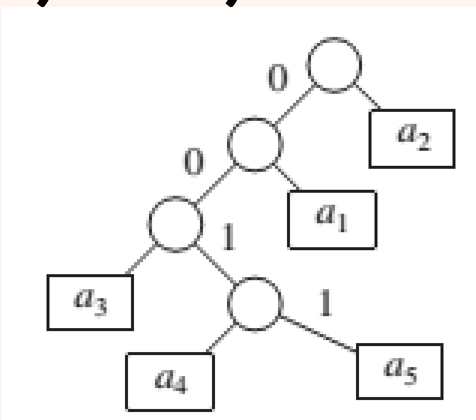
letter	probability	codeword
a_2	0.4	1
a_1	0.2	01
a_3	0.2	000
a_4	0.1	0010
a_5	0.1	0011

- به عنوان مثال کلمه‌کدها به ترتیب الفبایی مرتب شده و ارسال شوند.

$$\left[2,01,1,1,3,000,4,0010,4,0011\right]$$

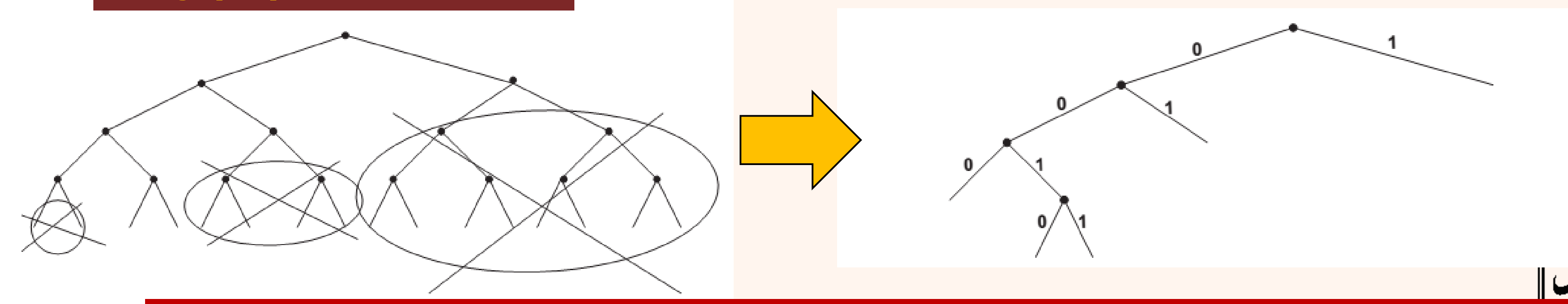
- هرچند چنین‌کاری برای الفبای کوچک شدنی است، اما در صورتی که تعداد نمادها افزایش یابد، حتی ممکن است اثر فشرده‌سازی را خنثی کند.

- با توجه به ساختار درخت‌های Huffman می‌توان با در اختیار داشتن طول کلمه‌کدها درخت را بازسازی کرد:



$$3, 4, 4, 2, 1$$

fully populated tree



با دانستن طول کلمه‌کدها می‌توان کد Huffman را بازتولید کرد.

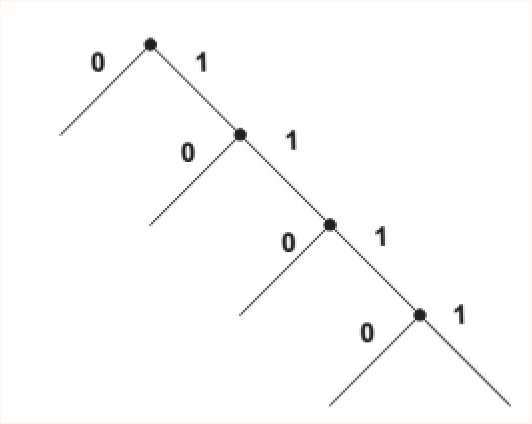
تنها مشکل این است که نمی‌دانیم که هر کدام متعلق به کدام نماد است!

Canonical Huffman Codes

- برای چیرگی بر این مشکل نوعی خاصی از کدگذاری پیشنهاد می‌شود:
  - نمادها به ترتیب الفبایی مرتب شوند.
  - کلمه‌کدهای کوتاه‌تر، پیش از کلمه‌کدهای بلند در ساختار درخت قرار گیرند.

$$\{2,1,3,4,4\}$$

letter	probability	codeword
a_2	0.4	0
a_1	0.2	10
a_3	0.2	110
a_4	0.1	1110
a_5	0.1	1111



در zlib از این شیوه استفاده شده است.

افزونگی کدگذاری Huffman

$$\rho = R - H\left(X\right) \geq 0; \quad R: \text{aveage code length}$$

افزونگی کد Huffman همیشه کم‌تر از یک بیت به ازای هر نماد است:

$$0 \leq \rho < 1$$

به عبارت دیگر

$$H\left(X\right) \leq R < H\left(X\right) + 1$$

نامساوی سمت چپ که براساس رابطه‌ی Shannon برقرارست.

نامساوی سمت راست:

$$R = \sum_{i=1}^{M} \rho_i \left\lceil -\log_2^{\rho_i} \right\rceil < \sum_{i=1}^{M} \rho_i \left(1 - \log_2^{\rho_i}\right) = H\left(X\right) + 1$$

سایر شیوه‌های کدگذاری

- کدگذاری Huffman منجر به متوسط نرخ بیت کمینه می‌شود، اما در مورد حداکثر طول کلمه‌کد هیچ محدودیتی ندارد.
  - در برخی کاربردها به دلیل جلوگیری از پیچیدگی سخت‌افزار و یا محدودیت طول ثبات‌ها نوعی از کدگذاری مطرح می‌شود که هرچند بهینه نیست، اما طول کلمه‌کدها دارای محدودیت است.

Length- Limited Huffman Codes

جزییات بیشتر در Sayood, K. (2012)

سایر شیوه‌های کدگذاری(ادامه...)

- در شرایطی که الفبای کوچکی داشته باشیم و احتمال وقوع نمادها متوازن نباشد، کد Huffman از کارایی کم‌تری برخوردار خواهد بود:

$$A=\{a_1,a_2,a_3\}$$

$$p(a_1)=0.8 \quad p(a_2)=0.02 \quad p(a_3)=0.18$$

$$Ent=0.816$$

Letter	Codeword
a_1	0
a_2	11
a_3	10

$$l_{av}=1.2\text{ bits/symbol}$$

نرخ بیت ۴۷٪ بیش از آنتروپی است

سایر شیوه‌های کدگذاری(ادامه...)

- در چنین شرایطی پیشنهاد می‌شود به جای کدکردن نمادها، مجموعه‌ای از نمادها را کد کنیم.

$$\mathcal{A}^{(n)} = \{\overbrace{a_1a_1\ldots a_1}^{\text{n times}}, a_1a_1\ldots a_2, \ldots, a_1a_1\ldots a_m, a_1a_1\ldots a_2a_1, \ldots, a_ma_m\ldots a_m\}$$

Letter	Probability	Code
a_1a_1	0.64	0
a_1a_2	0.016	10101
a_1a_3	0.144	11
a_2a_1	0.016	101000
a_2a_2	0.0004	10100101
a_2a_3	0.0036	1010011
a_3a_1	0.1440	100
a_3a_2	0.0036	10100100
a_3a_3	0.0324	1011

$$H(X) \le R < H(X) + \frac{1}{n}$$

$$H\left(X^{(n)}\right) \le R^{(n)} < H\left(X^{(n)}\right) + 1$$

$$R = \frac{1}{n} R^{(n)}$$

$$l_{av} = 1.7228 \text{ bits/digram} = 0.8614 \text{ bits/symbols}$$

افزایش اندازه‌ی الفبا می‌تواند منجر به ناکارایی کدگذاری شود.

- میزان اطلاعات یک دنباله به طول n (با الفبایی شامل m نماد) با فرض مستقل بودن و یکسان بودن توزیع نمادها

$$H\left(\mathbf{x}^n\right)=nH\left(x\right)$$

- اثبات:

$$H\left(\mathbf{x}^n\right)=-\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m}P\left(a_{i_1},a_{i_2},\ldots,a_{i_n}\right)\log\left[P\left(a_{i_1},a_{i_2},\ldots,a_{i_n}\right)\right]$$

- با در نظر گفتن iid بودن منبع:

$$H\left(\mathbf{x}^n\right)=-\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m}P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n})\log\left[P\left(a_{i_1},a_{i_2},\ldots,a_{i_n}\right)\right]$$

$$H\left(\mathbf{x}^n\right)=-\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m}P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n})\sum_{j=1}^{n}\log P(a_{i_j})$$

$$H\left(\mathbf{x}^n\right) = -\sum_{i_1=1}^{m} P(a_{i_1})\log P(a_{i_1})\boxed{\left\{\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_2})\cdots P(a_{i_n})\right\}}$$

$$-\sum_{i_2=1}^{m} P(a_{i_2})\log P(a_{i_2})\boxed{\left\{\sum_{i_1=1}^{m}\sum_{i_3=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1})P(a_{i_2})\cdots P(a_{i_n})\right\}}$$

$$\vdots$$

$$-\sum_{i_n=1}^{m} P(a_{i_n})\log P(a_{i_n})\boxed{\left\{\sum_{i_1=1}^{m}\sum_{i_3=1}^{m}\cdots\sum_{i_{n-1}=1}^{m} P(a_{i_1})P(a_{i_2})\cdots P(a_{i_{n-1}})\right\}}$$

برابر با یک

$$\Rightarrow \boxed{H\left(\mathbf{x}^n\right) = nH\left(x\right)}$$

پیاده‌سازی کد Huffman

- کدگذاری به سادگی و با کمک یک «جدول مراجعه» انجام می‌شود.
- کدگشایی، با چنین شیوه‌ای به یک جدول بزرگ احتیاج خواهد داشت.
  - متناسب با بزرگ‌ترین کلمه‌کد
- می‌توان از ساختار درخت دودیی استفاده کرد.
  - پیمایش درخت زمان‌بر است
- معمولا از ترکیب این دو استفاده می‌شود.

سایر شیوه‌های کدگذاری(ادامه...)

- در صورتی که مشخصات آماری نمادها در طول زمان تغییر کند، استفاده از کدگذاری وفقی Huffman پیشنهاد می‌شود.

Adaptive Huffman Coding

- به علت حجم محاسبات بالا کم‌تر مورد استفاده قرار می‌گیرد.

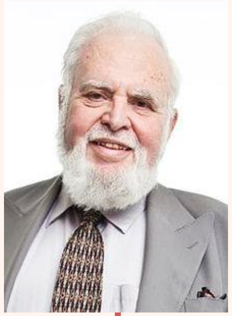
جزییات بیشتر در Sayood, K. (2012)

Unary coding

- این شیوه‌ی کدگذاری برای اعداد طبیعی پیشنهاد شده است:
  - برای کدکردن عدد n، به تعداد n عدد یک و در پایان صفر قرار می‌دهیم.
  - در صورتی که $p(n)=1/2^{n+1}$ این شیوه بهینه است.(معادل Huffman)
  - با وجود این که تابع احتمال فوق در موارد محدودی بهینه است، اما پیاده‌سازی آن بسیار ساده است.

n	Unary code
0	0
1	10
2	110
3	1110
4	11110
5	111110

کدگذاری Golomb



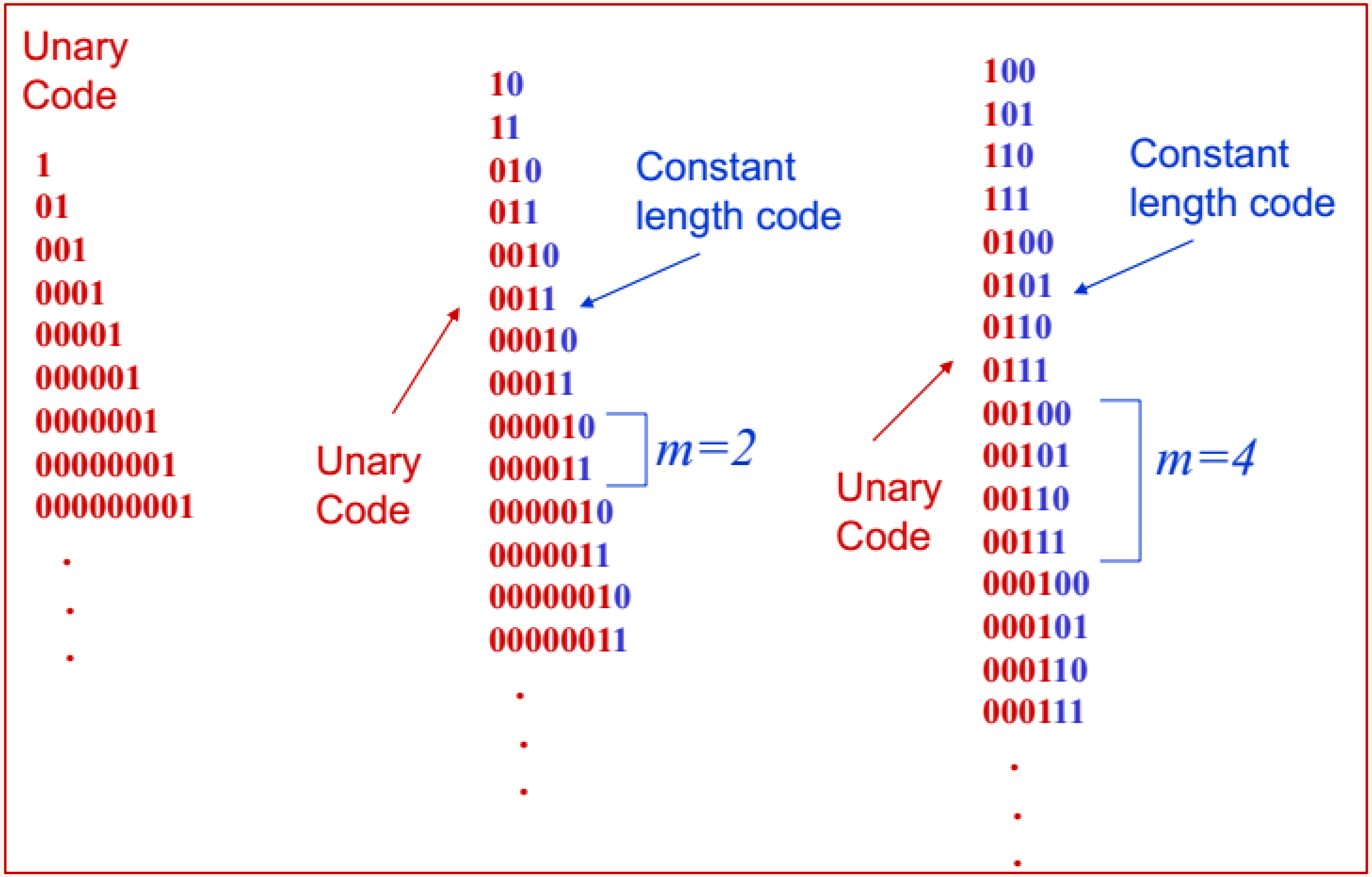
با پارامتر m

- ابتدا خارج قسمت و باقیمانده نسبت به m محاسبه می‌شود:

$$q = \left\lfloor \frac{n}{m} \right\rfloor \qquad\qquad r = n - qm$$

- q(خارج قسمت)، با استفاده از کدهای unary کد می‌شوند.
- باقیمانده‌ها در بازه‌ی 0 تا m-1 قرار دارد.
- در صورتی که m توانی از ۲ باشد، می‌توان از کدهای باینری استفاده کرد.

مثال



Bernd Girod: EE398A Image and Video Compression

به صورت مشابه می‌توان کدها را به صورت معکوس هم تعریف کرد.

Truncated binary encoding

در غیر این‌صورت از Truncated binary encoding استفاده می‌شود.

در این حالت هدف کدگذاری m نماد است.
چنان‌چه m توانی از ۲ می‌بود و از کدگذاری دودویی معمولی استفاده می‌شد، به $\log_2 m$ بیت نیاز می‌بود.
در این حالت که $2^n < m < 2^{n+1}$
برای $2^{n+1}-m$ نماد اول از n بیت
و برای مابقی نمادها از $n+1$ بیت استفاده شود.

truncated binary encoding for numbers from 0 to 5

0	00	4	110
1	01	5	111
2	100		
3	101		

Truncated binary encoding

- این شیوه برای نمادهایی که توزیع یکنواخت دارند، مناسب است.
- نمادهای بزرگ‌تر(از نظر الفبایی) با رشته‌ی بزرگ‌تر کد می‌شود.
- هیچ کد n بیتی پیشوند کد n+1 بیتی نیست.

تمرین: برای m=5 کلمه‌کدها را به دست آورید.

0	**00**
1	01
2	10
3	110
4	111

مثال-کدگذاری Golomb

$m=5$

n	q	r	Codeword	n	q	r	Codeword
0	0	0	000	08	1	3	10110
1	0	1	001	09	1	4	10111
2	0	2	010	10	2	0	11000
3	0	3	0110	11	2	1	11001
4	0	4	0111	12	2	2	11010
5	1	0	1000	13	2	3	110110
6	1	1	1001	14	2	4	110111
7	1	2	1010	15	3	0	111000

الفبای نامتوازن

مثال

- در مواردی که با الفبایی سروکار داریم که در آن حروف دارای احتمال وقوع نامتوازن هستند:

α_i	Prob.	Code word
α_1	0.95	0
α_2	0.02	10
α_3	0.03	11

$$Ent = 0.335$$

$$\text{Avrage Bit Rate}=0.95\times 1+0.05\times 2=1.05 \text{ bits/symbol}$$

نرخ بیت بیش از دو برابر بیشتر از حدی که آنتروپی تعیین کرده

کدگذاری دنباله‌ای از نمادها

Letter	Probability	Code
a_1a_1	0.9025	0
a_1a_2	0.0190	111
a_1a_3	0.0285	100
a_2a_1	0.0190	1101
a_2a_2	0.0004	110011
a_2a_3	0.0006	110001
a_3a_1	0.0285	101
a_3a_2	0.0006	110010
a_3a_3	0.0009	110000

Avrage Bit Rate=0.611 bits/symbols

با ادامه دادن همین روند (بلوک‌های هشت‌تایی) به حداقل نرخ بیت نزدیک خواهیم شد، در این صورت جدولی بزرگ خواهیم داشت(۶۵۶۱).

- به حجم حافظه‌ی بالایی نیاز است که در بسیاری از کاربردها عملی نیست.
- کدگشایی هزینه‌ی محاسباتی بالایی دارد.
- در صورتی که احتمال رخداد نمادها تغییر کند، اثر آن بر روی کارایی بالا خواهد بود.

کدگذاری دنباله‌ای از نمادها(ادامه...)

تولید یک کلمه‌کد برای گروهی از نمادها منجر به افزایش کارایی می‌شود.

پیاده‌سازی Huffman در چنین حالتی دشوار است.
چرا که باید برای همه‌ی ترکیب‌های نمادها معادل کد آن را در اختیار داشت. رشد تعداد کلمه‌کدها با گروه نمادها نمایی خواهد بود.

کدهای محاسباتی راهی است که بتوان مجموعه‌ای از نمادها برچسب زد بدون این که نیاز باشد برای همه‌ی ترکیب‌های ممکن برچسب تهیه کرد.

کدهای محاسباتی

۱۹۴۸

- برای اولین بار در مقاله‌ی معروف Shannon استفاده از تابع توزیع تجمعی مطرح شد.

۱۹۶۳

- اولین بار توسط Elias مطرح شد، هرچند این مطلب را منتشر نکرد.
  - در کتاب Abramson این مسأله آورده شد.

۱۹۷۶

- به شیوه‌ی کنونی به صورت جداگانه توسط Pasco و Rissanen کشف شد.
- در این شیوه به جای اختصاص کلمه‌کد برای هر نماد، به هر دنباله از نمادها یک برچسب نسبت داده می‌شود.

فرآیند کدگذاری

- به هر دنباله‌ای از نمادها باید یک «برچسب یکتا» نسبت داده شود.
- این برچسب یک عدد در بازه‌ی [0,1) است.
- برای محاسبه‌ی این برچسب از تابع توزیع تجمعی(cdf) استفاده می‌شود.
- در این روش کل نمادهای موجود در نظر گرفته می‌شود و بر حسب احتمال وقوع به بازه‌ی [0 1] تقسیم می‌شود.

$$A = \{a_0, a_1, \ldots, a_{M-1}\}$$

$$P(X = i) = P(a_i) = \rho_i \quad \text{probabilty density function}$$

$$F_X(i) = \sum_{k=1}^{i} \rho_k \quad \text{cumulative density function}$$

arithmetic coding is a practical way of implementing entropy coding

- فرض کنید M نماد با احتمال وقوع ρ_i برای هر یک وجود دارد.
- در این صورت بازه‌ی [0 1] را می‌توان به صورت زیر تقسیم نمود:

$$[0, \rho_1, \rho_1 + \rho_2, \ldots\ldots, \sum_{i=1}^{M-1} \rho_i, 1]$$

- در این صورت نماد k–ام در محدوده زیر قرار دارد:

$$(\sum_{i=1}^{k-1} \rho_i, \sum_{i=1}^{k} \rho_i)$$

یا

$$(F_X(k-1), F_X(k))$$

مثال ۱

- فرض کنید دنباله‌ای از نمادها به صورت زیر در دست باشد و بخواهیم از این روش کد گردد:

aab

$$[0.343, 0.441]$$

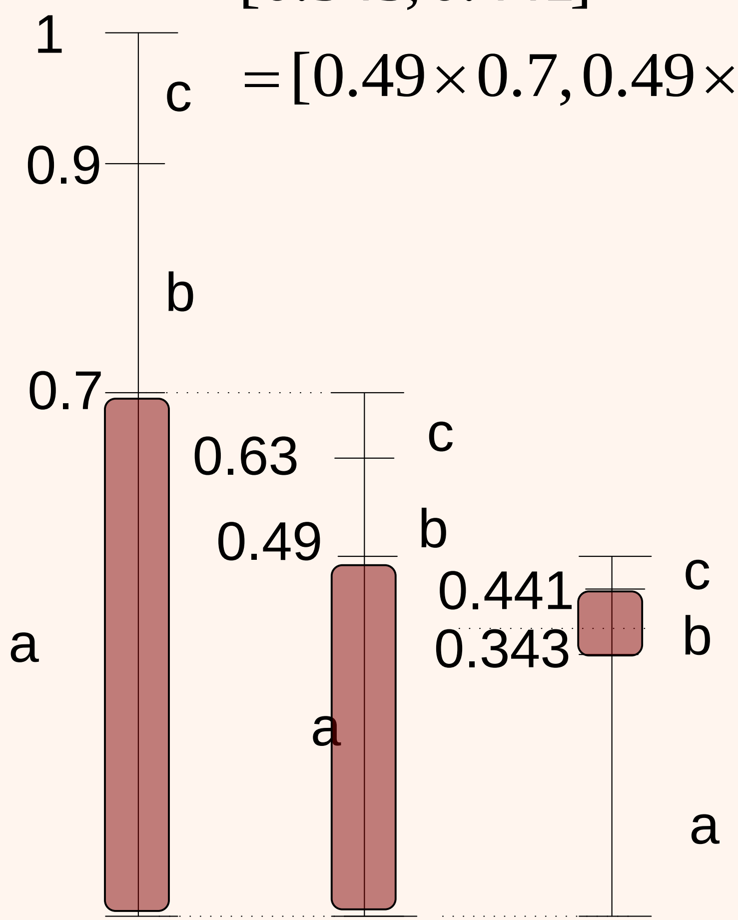
$$= [0.49\times 0.7, 0.49\times 0.7 + 0.49\times 0.2]$$

احتمال وقوع

$a \rightarrow 0.7$

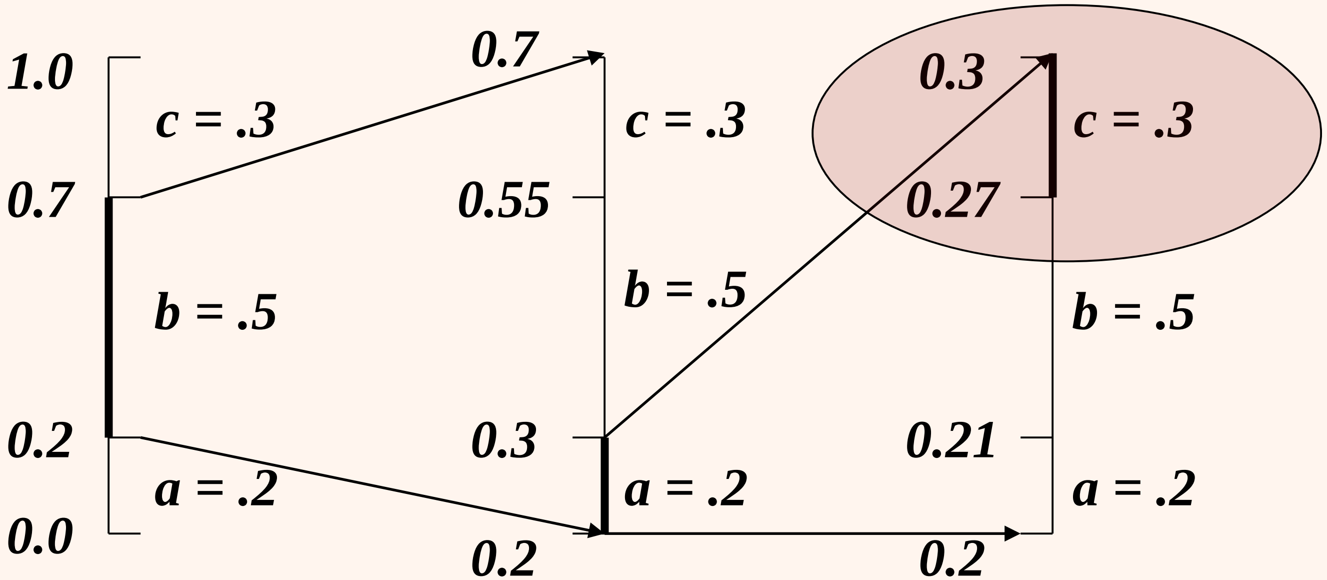
$b \rightarrow 0.2$

$c \rightarrow 0.1$



مثال ۲

کدگذاری رشته: bac

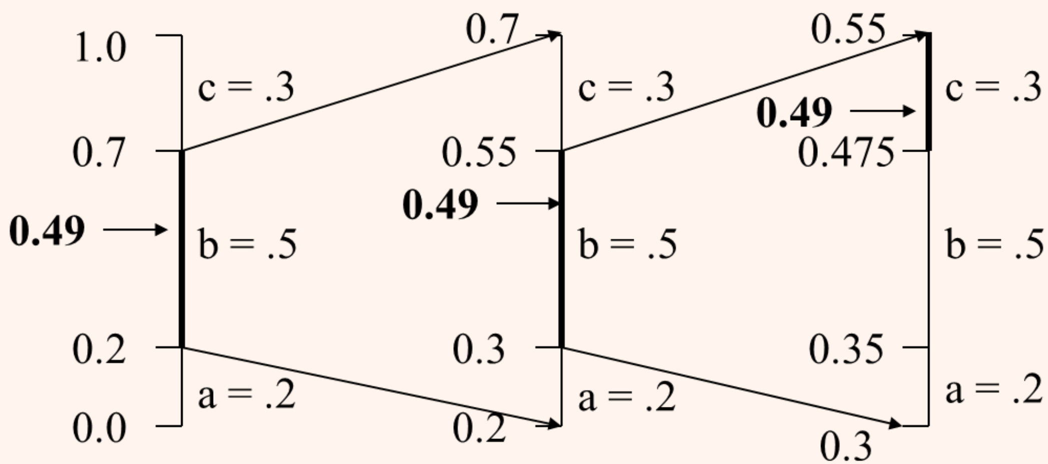


محدوده‌ی نهایی [0.27,0.3)

محدوده‌ی به دست آمده برای هر جمله یکتا، به صورت non overlapped خواهد بود
برای بازگشایی کد نیز از این خاصیت استفاده می‌شود.

کدگشایی-مثال

کدگشایی عدد ۰.۴۹ برای جریان داده با طول ۳



bbc.

فرآیند کدگذاری(ادامه...)

- هر نقطه‌ای از این بازه را می‌توان به عنوان برچسب یکتا در نظر گرفت.
  - نقطه‌ی میانی این بازه می‌تواند انتخاب مناسبی باشد.
- برای یک نماد، این عدد را می‌توان به صورت زیر در نظر گرفت:

$$\overline{T}_X(a_i) = \sum_{k=1}^{i-1} P(X=k) + \frac{1}{2}P(X=i)$$
$$= F_X(i-1) + \frac{1}{2}P(X=i)$$

داده‌ها به صورت الفبایی مرتب شده‌اند

- و برای دنباله‌ای به طول m

$$\overline{T}_{\mathbf{X}}^{(m)}(\mathbf{x}_i) = \sum_{\mathbf{y}<\mathbf{x}_i} P(\mathbf{y}) + \frac{1}{2}P(\mathbf{x}_i)$$

مثال

- در صورتی که الفبا شامل شش نماد با احتمال یکسان باشد، خواهیم داشت:

$$\overline{T}_X(2) = P(X=1) + \frac{1}{2}P(X=2) = \frac{1}{6} + \frac{1}{12} = 0.25$$

$$\overline{T}_X(5) = \sum_{k=1}^{4} P(X=k) + \frac{1}{2}P(X=5) = 0.75$$

$$\begin{aligned}\overline{T}_X(13) &= P(\mathbf{x}=11) + P(\mathbf{x}=12) + 1/2P(\mathbf{x}=13)\\ &= 1/36 + 1/36 + 1/2(1/36)\\ &= 5/72\end{aligned}$$

محاسبه‌ی برچسب

- برای هر دنباله باید کران پایین و بالای بازه را مشخص کرد.
  - برای نماد اول

$$(l^{(1)}, u^{(1)}) = (F_X(x_1 - 1), F_X(x_1))$$

و برای سایر نمادها:

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n - 1)$$
$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n)$$

و در نهایت نقطه‌ی میانی:

$$\overline{T}_X(\mathbf{x}) = \frac{u^{(n)} + l^{(n)}}{2}$$

برای کدگذاری تنها تابع توزیع تجمعی مورد نیاز است.

مثال

- الفبای زیر را در نظر بگیرید:

$$A = \{a_1, a_2, a_3\}$$

$$P(a_1) = 0.8 \quad P(a_2) = 0.02 \quad P(a_3) = 0.18$$

هدف کد کردن دنباله‌ی 1 3 2 1

$$F_X(k) = 0, \; k \leqslant 0, \quad F_X(1) = 0.8, \quad F_X(2) = 0.82, \quad F_X(3) = 1$$

$$l^{(1)} = 0 + (1-0)0 = 0$$
$$u^{(1)} = 0 + (1-0)(0.8) = 0.8$$

$$l^{(2)} = 0 + (0.8-0)F_X(2) = 0.8 \times 0.82 = 0.656$$
$$u^{(2)} = 0 + (0.8-0)F_X(3) = 0.8 \times 1.0 = 0.8$$

$$l^{(3)} = 0.656 + (0.8-0.656)F_X(1) = 0.656 + 0.144 \times 0.8 = 0.7712$$
$$u^{(3)} = 0.656 + (0.8-0.656)F_X(2) = 0.656 + 0.144 \times 0.82 = 0.77408$$

$$l^{(4)} = 0.7712 + (0.77408 - 0.7712)F_X(0) = 0.7712 + 0.00288 \times 0.0 = 0.7712$$
$$u^{(4)} = 0.7712 + (0.77408 - 0.1152)F_X(1) = 0.7712 + 0.00288 \times 0.8 = 0.773504$$

$$\overline{T}_X(1321) = \frac{0.7712 + 0.773504}{2} = 0.772352$$

تولید کد دودویی

- هدف یافتن یک کد دودویی یکتا به روشی کاراست.
- در برخی موارد تعداد ارقام برچسب به دست آمده، بی‌شمار است.
- چنان چه $p(x)$ احتمال رخداد یک دنباله از نمادها باشد. طول معادل دودویی آن از رابطه‌ی زیر به دست می‌آید:

$$l(\mathbf{x}) = \left\lceil \log_2 \left(\frac{1}{p(\mathbf{x})} \right) \right\rceil + 1$$

- می‌توان نشان داد پس از این نوع نمایش کد یکتا و قابل کدگشایی است.

مثال

$$\mathcal{A} = \{a_1, a_2, a_3, a_4\}$$

$$P(a_1) = \frac{1}{2}, \quad P(a_2) = \frac{1}{4}, \quad P(a_3) = \frac{1}{8}, \quad P(a_4) = \frac{1}{8}$$

Symbol	F_X	$\overline{T}_X$	In Binary	$\lceil \log \frac{1}{P(x)} \rceil + 1$	Code
1	.500	.2500	.0100	2	01
2	.750	.6250	.1010	3	101
3	.875	.8125	.1101	4	1101
4	1.000	.9375	.1111	4	1111

یکتایی کد محاسباتی

- برچسب به دست آمده با طول $l(x)$ بریده شود.

$$T_X\left(x\right) \to \left\lfloor T_X\left(x\right)\right\rfloor_{l(x)}$$

- در این صورت $\left\lfloor T_X\left(x\right)\right\rfloor_{l(x)}$ نیز یکتاست.
- باید ثابت کنیم:

$$F_X\left(\mathbf{x}-1\right) < \left\lfloor T_X\left(x\right)\right\rfloor_{l(x)} < F_X\left(\mathbf{x}\right)$$

- درستی بخش راست این نامساوی واضح است!

یکتایی کد محاسباتی

$$F_X(\mathbf{x}-1) < \lfloor T_X(x) \rfloor_{l(x)}$$

$$0 \le T_X(x) - \lfloor T_X(x) \rfloor_{l(x)} < \frac{1}{2^{l(\mathbf{x})}}$$

۱

$$\lfloor T_X(x) \rfloor_{l(x)} > T_X(x) - \frac{1}{2^{l(\mathbf{x})}}$$

۲

$$\frac{1}{2^{l(\mathbf{x})}} = \frac{1}{2^{\left\lceil \log_2\left(\frac{1}{p(\mathbf{x})}\right)\right\rceil+1}} < \frac{1}{2^{\log_2\left(\frac{1}{p(\mathbf{x})}\right)+1}} = \frac{P(\mathbf{x})}{2}$$

۳

$$\frac{P(\mathbf{x})}{2} = T_X(x) - F_X(\mathbf{x}-1)$$

۴

$$T_X(x) = F_X(\mathbf{x}-1) + \frac{P(\mathbf{x})}{2}$$

$$T_X(x) - F_X(\mathbf{x}-1) > \frac{1}{2^{l(\mathbf{x})}}$$

۵

$$F_X(\mathbf{x}-1) < T_X(x) - \frac{1}{2^{l(\mathbf{x})}}$$

۶

۲ و ۶

$$F_X(\mathbf{x}-1) < \lfloor T_X(x) \rfloor_{l(x)}$$

۷

کد پیشوندی

- ثابت می‌شود که کد به دست آمده پیشوندی است:
  - در صورتی که a یک عدد در بازه‌ی $[0,1)$ با نمایش دودویی $[b_1b_2...b_n]$ باشد، اگر عدد دیگری مانند b دارای پیشوند یکسان $[b_1b_2...b_n]$ با a باشد خواهیم داشت:

$$b \in [a, a + \frac{1}{2^n})$$

کد پیشوندی

می‌خواهیم ثابت کنیم که تمام کدهایی که پیشوند آن‌ها کد به دست آمده‌است در همین بازه قرار می‌گیرند.

- می‌دانیم $F_X(\mathbf{x}-1) < \lfloor T_X(x) \rfloor_{l(x)} < F_X(\mathbf{x})$

$$F_X(\mathbf{x}) - \lfloor T_X(x) \rfloor_{l(x)} > F_X(\mathbf{x}) - T_X(x) = \frac{P(x)}{2} > \frac{1}{2^{l(\mathbf{x})}}$$

$$\boxed{\frac{P(\mathbf{x})}{2} = T_X(x) - F_X(\mathbf{x}-1)}$$

$$\Downarrow$$

$$F_X(\mathbf{x}) - \lfloor T_X(x) \rfloor_{l(x)} > \frac{1}{2^{l(\mathbf{x})}}$$

در نتیجه

$$\boxed{[\lfloor T_X(x) \rfloor_{l(x)}, \lfloor T_X(x) \rfloor_{l(x)} + \frac{1}{2^{l(\mathbf{x})}}) \in [F_X(\mathbf{x}-1), F_X(\mathbf{x}))}$$

کد به دست آمده پیشوندی است!

کارایی کد محاسباتی

طول متوسط دنباله‌های کد شده به طول m

- پیش از به کارگیری کدهای محاسباتی باید میزان سودمندی آن‌ها را بسنجید:

$$l(\mathbf{x}) = \left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil + 1$$

$$\begin{aligned} l_{A^{(m)}} &= \sum P(\mathbf{x}) l(\mathbf{x}) \\ &= \sum P(\mathbf{x}) \left[\left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil + 1 \right] \\ &< \sum P(\mathbf{x}) \left[\log \frac{1}{P(\mathbf{x})} + 1 + 1 \right] \\ &= \sum P(\mathbf{x}) \log \frac{1}{P(\mathbf{x})} + 2 \sum P(\mathbf{x}) \end{aligned}$$

$$\Rightarrow \boxed{l_{A^{(m)}} < H\left(\mathbf{x}^m\right) + 2}$$

کارایی کد محاسباتی(ادامه...)

- با توجه به این که طول متوسط همواره از آنتروپی بزرگ‌تر است، خواهیم داشت:

$$H\left(\mathbf{x}^m\right) < l_{A^{(m)}} < H\left(\mathbf{x}^m\right) + 2$$

- میزان متوسط نرخ بیت به ازای هر نماد:

$$l_A = \frac{l_{A^{(m)}}}{m}$$

- در نتیجه

$$\boxed{H\left(\mathbf{x}^m\right) = mH\left(x\right)}$$

$$\Rightarrow \quad \frac{H\left(\mathbf{x}^m\right)}{m} < l_A < \frac{H\left(\mathbf{x}^m\right) + 2}{m}$$

$$\boxed{H\left(x\right) < l_A < H\left(x\right) + \frac{2}{m}}$$

کدهای محاسباتی وفقی

Adaptive Arithmetic Coding

- در بسیاری از کاربردها تابع احتمال مشاهده‌ی نمادها(توزیع تجمعی) از پیش مشخص نیست.
  - در این حالت در ابتدا فرض می‌شود همه‌ی نمادها احتمال وقوع یکسان دارند.
  - با مشاهده‌ی هر نماد احتمال وقوع مربوط آن به روز شود.
  - این کار به طور همزمان در فرستنده و گیرنده به صورت مشابه انجام شود.