

# بازیابی اطلاعات

دکتر امین گلزاری اسکویی

[a.golzari@azaruniv.ac.ir](mailto:a.golzari@azaruniv.ac.ir)

[a.golzari@tabrizu.ac.ir](mailto:a.golzari@tabrizu.ac.ir)

<https://github.com/Amin-Golzari-Oskoue>



دانشگاه صنعتی ارومیه

پاییز ۱۴۰۲

# فصل ۵

## فشرده سازی شاخص

مطالب این فصل

- هدف از فشرده سازی در سیستم‌های بازیابی اطلاعات
- چگونه مولفه شاخص معکوس لغت‌نامه را می‌توان فشرده‌سازی کرد؟
- چگونه مولفه شاخص معکوس پست‌ها را می‌توان فشرده سازی کرد؟
- ویژگی آماری اصطلاحات: اصطلاحات در مجموعه اسناد چگونه توزیع می‌شوند؟

## چرا فشرده‌سازی انجام می‌دهیم؟ (به صورت کلی)

❖ از فضای دیسک کمتری استفاده می‌شود. (صرفه جویی در هزینه)

❖ داده‌های بیشتری را می‌توان در حافظه اصلی نگه‌داری کرد. (افزایش سرعت)

❖ سرعت انتقال داده‌ها از دیسک افزایش می‌یابد.

. فشرده‌سازی و از حالت فشرده خارج کردن در حافظه اصلی، سریع‌تر از داده‌ی غیرفشرده‌سازی شده است.

❖ فرض: الگوریتم‌های Decompression سریع‌تر هستند.

❖ ما از الگوریتم‌های Decompression استفاده خواهیم کرد.

## چرا در بازیابی اطلاعات فشرده‌سازی انجام می‌دهیم؟

- ❖ ابتدا، فضایی را برای لغت‌نامه در نظر می‌گیریم.
  - انگیزه اصلی برای فشرده‌سازی لغت‌نامه: تا مدی حجم آن را کوچک کنیم که بتوانیم درون حافظه اصلی نگه داریم.
- ❖ برای فایل پست‌ها:
  - انگیزه: کاهش فضای مورد نیاز، افزایش زمان مورد نیاز برای خواندن از دیسک.
  - توجه: موتورهای جستجو بزرگ بخش قابل توجهی از پست‌ها را داخل حافظه اصلی نگه می‌دارند.
- ❖ مدل های فشرده‌سازی گوناگونی برای لغت‌نامه و پست‌ها ایجاد خواهیم کرد.

# فشرده سازی بی‌اتلاف VS فشرده سازی با‌اتلاف

- ❖ فشرده سازی با‌اتلاف: برخی اطلاعات دور ریخته می‌شوند.
- ❖ بسیاری از مراحل پیش پردازش که ما مکرراً انجام می‌دهیم می‌تواند به عنوان فشرده سازی با‌اتلاف باشد:
  - تبدیل مروف بزرگ به کوچک، کلمات توقف، ریشه‌گیری، حذف اعداد
- ❖ فشرده سازی بی‌اتلاف: تمامی اطلاعات محفوظ می‌مانند.
  - کاری که اغلب در فشرده سازی شفاف انجام می‌دهیم.

## مجموعه RCV1

symbol	statistics	value
$N$	documents	800,000
$L$	avg. # tokens per document	200
$M$	word types	400,000
	avg. # bytes per token (incl. spaces/punct.)	6
	avg. # bytes per token (without spaces/punct.)	4.5
	avg. # bytes per term (= word type)	7.5
$T$	non-positional postings	100,000,000

## تأثیر پارامترها بر اندازه شاخص ساخته شده

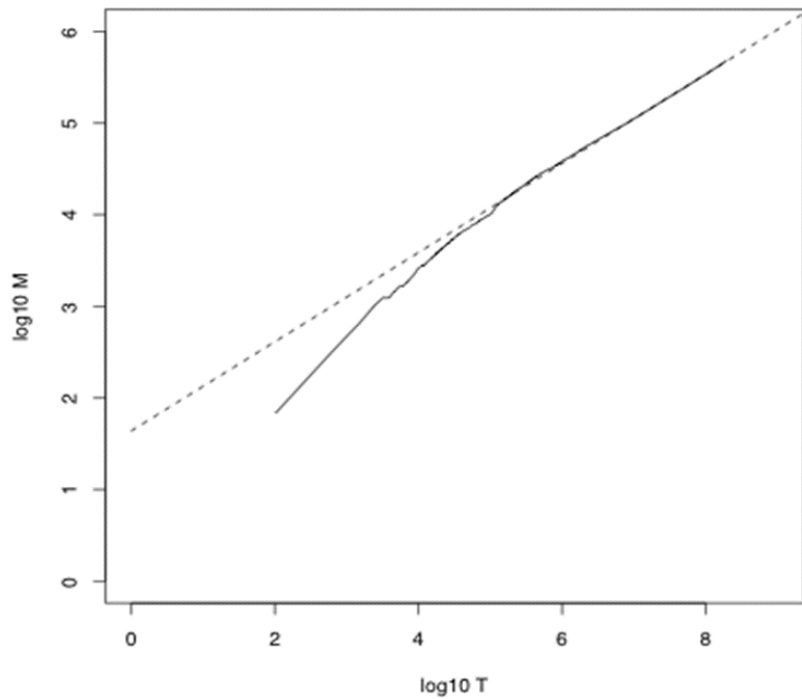
size of	word types (term)			non-positional postings			positional postings (word tokens)		
	dictionary			non-positional index			positional index		
	size	$\Delta$	cml..	size	$\Delta$	cml..	size	$\Delta$	cml..
unfiltered	484,494			109,971,179			197,879,290		
no numbers	473,723	-2%	-2%	100,680,242	-8%	-8%	179,158,204	-9%	-9%
case folding	391,523	-17%	-19%	96,969,056	-3%	-12%	179,158,204	-0%	-9%
30 stop w's	391,493	-0%	-19%	83,390,443	-14%	-24%	121,857,825	-31%	-38%
150 stop w's	391,373	-0%	-19%	67,001,847	-30%	-39%	94,516,599	-47%	-52%
stemming	322,383	-17%	-33%	63,812,300	-4%	-42%	94,516,599	-0%	-52%

تأثیر پیش پردازش روی تعداد عبارات، پست‌های غیر موقعیتی، و نشانه‌های Reuters-RCV1. “ $\Delta\%$ ” نشان‌دهنده‌ی کاهش اندازه از خط قبلی است، به جز “30 کلمه توقف” و “150 کلمه توقف” که هر دو غیر محاسب کردن به مروف کوچک و بزرگ را به عنوان خط مرجع‌شان به کار می‌برند. “ $T\%$ ” کاهش تجمعی (“کلی”) از حالت فیلتر نشده است. ما ریشه‌گیری را با ریشه‌گیر Porter انجام دادیم.

## لیست اصطلاحات تا چه اندازه‌ای می‌تواند بزرگ باشد؟

- ❖ چه تعداد کلمات متمایز از هم وجود دارد؟
- ❖ آیا می‌توان کران بالایی را در نظر گرفت؟
- ❖ Not really: At least  $70^{20} \approx 10^{37}$  different words of length 20.
- ❖ تعداد واژگان (لغت‌نامه) با افزایش اندازه مجموعه افزایش خواهد یافت.
- ❖ Heaps' law:  $M = kT^b$
- ❖  $M$  اندازه تعداد واژگان،  $T$  تعداد token در مجموعه است.
- ❖ مقادیر معمولی برای پارامترهای  $k$  و  $b$  شامل  $30 \leq k \leq 100$  و  $b \approx 0.5$  است.
- ❖ قانون Heap در فضای log-log خطی است.
- ❖ این ساده‌ترین رابطه ممکن بین اندازه مجموعه و اندازه لغت‌نامه در فضای log-log است. (به صورت تجربی)





قانون Heap، اندازه‌ی مجموعه واژگان  $M$  به عنوان تابعی از اندازه‌ی مجموعه  $T$  (تعداد نشانه‌ها) برای Reuters-RCV1. برای این داده‌ها، خط چین  $\log_{10} M = 0.49 * \log_{10} T + 1.64$  بهترین برازش کمترین مربعات است. بنابراین  $K = 10^{1.64} \approx 44$  و  $b = 0.49$  است.



“

❖ **مثال :** برای 1,000,020 توکن قانون heap، 38,323 اصطلاح پیش‌بینی می‌کند.

❖ تعداد واقعی برابر 38,365 اصطلاح، که نزدیک به پیش‌بینی است.

❖ بر اساس تجربه **fit** در کل خوب است.

## تمرین

① تأثیر گنجاندن اشتباهات املایی در مقابل تصمیع خودکار اشتباهات املایی در قانون Heaps چیست؟

② اندازه واژگان  $M$  را محاسبه کنید.

□ همچنین برای 1,000,000 صفحه اول، اندازه دیکشنری برابر با 30000 عبارت است.

□ فرض کنید که موتور جستجو 20 میلیارد صفحه وب را شاخص گذاری می‌کند. هر کدام از صفحات به طور متوسط دارای 200 توکن هستند.

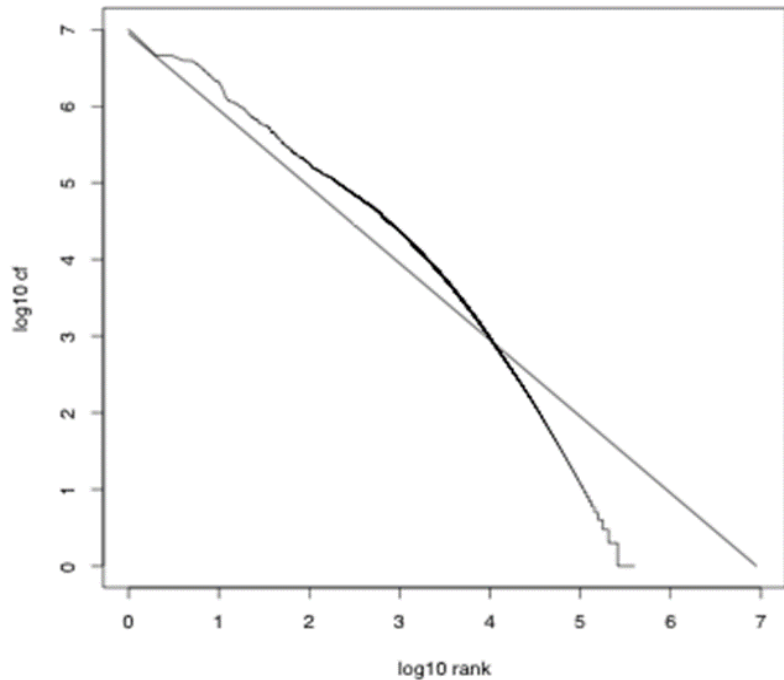
□ اندازه دیکشنری  $M$  برای این مجموعه صفحات را با استفاده از قانون Heap محاسبه کنید.

## قانون Zipf

❖ مدل رایج توزیع عبارات در یک مجموعه، قانون Zipf است. این قانون بیان می‌کند که اگر  $t_1$  رایج‌ترین عبارت در مجموعه،  $t_2$  عبارت رایج دوم باشد، و به همین ترتیب، آنگاه فراوانی مجموعه‌ی  $i$ امین رایج‌ترین عبارت،  $cf_i$  با

$$\frac{1}{i} \text{ متناسب است: } cf_i \propto \frac{1}{i}$$

❖ بنابراین، اگر عبارت با بیشترین فراوانی،  $cf_i$  بار اتفاق بیفتد، پس دومین عبارت با فراوانی زیاد نیمی از وقوع‌های اولی را خواهد داشت، سومین عبارت با فراوانی زیاد یک سوم وقوع‌های اولی را خواهد داشت، به همین ترتیب پیش می‌رود. مفهوم این است که فراوانی نسبت به رتبه، بسیار سریع کاهش می‌یابد. معادله  $cf_i \propto \frac{1}{i}$  یکی از ساده‌ترین فرمول‌های فرموله کردن چنین کاهش سریعی است و همچنین مشخص شده است که این مدل بسیار معقول است. به طور مشابه، می‌توانیم قانون Zipf را به صورت  $cf_i = ci^k$  یا به صورت  $\log cf_i = \log c + k \log i$  بنویسیم در صورتیکه  $k=-1$  و  $c$  یک عدد ثابت باشد.



فراوانی به عنوان تابعی از رتبه‌ی فراوانی، برای عبارات در مجموعه رسم شده است. خط، توزیع پیش‌بینی شده توسط

قانون Zipf است (برازش وزندار کمترین مربعات؛ نقطه تلاقی 6.95 است).

## فشرده سازی لغت نامه

❖ لغت نامه ها در مقایسه با فایل پست ها کوچک هستند.

❖ به این دلیل که ما می خواهیم آن ها را در حافظه اصلی نگه داری کنیم.

❖ با فشرده سازی دیکشنری فضای خالی برای برنامه های دیگر در حافظه اصلی فراهم می شود. حتی اگر دیکشنری در حافظه اصلی نباشد، برای اینکه سرعت جستجو بالاتر شود نیاز به کوچک کردن اندازه دیکشنری داریم.

❖ فشرده سازی لغت نامه مهم است.

term	document frequency	pointer to postings list
a	656,265	→
aachen	65	→
...	...	...
zulu	221	→

Space needed: 20 bytes      4 bytes      4 bytes

for Reuters:  $(20+4+4)*400,000 = 11.2 \text{ MB}$

اگر آرایه ای از مدفل های با عرض ثابت به عنوان لغت نامه در نظر بگیریم آنگاه برای مجموعه واژگان به تعداد 400 هزار کلمه نیاز به 11.2MB حافظه داریم.

## ورودی های با طول ثابت (عرض ثابت) مناسب نیستند.

❖ بیشتر بایت های ستون اصطلاح هدر می روند.

❖ 20 بایت برای هر اصطلاح به طول یک اختصاص می دهیم.

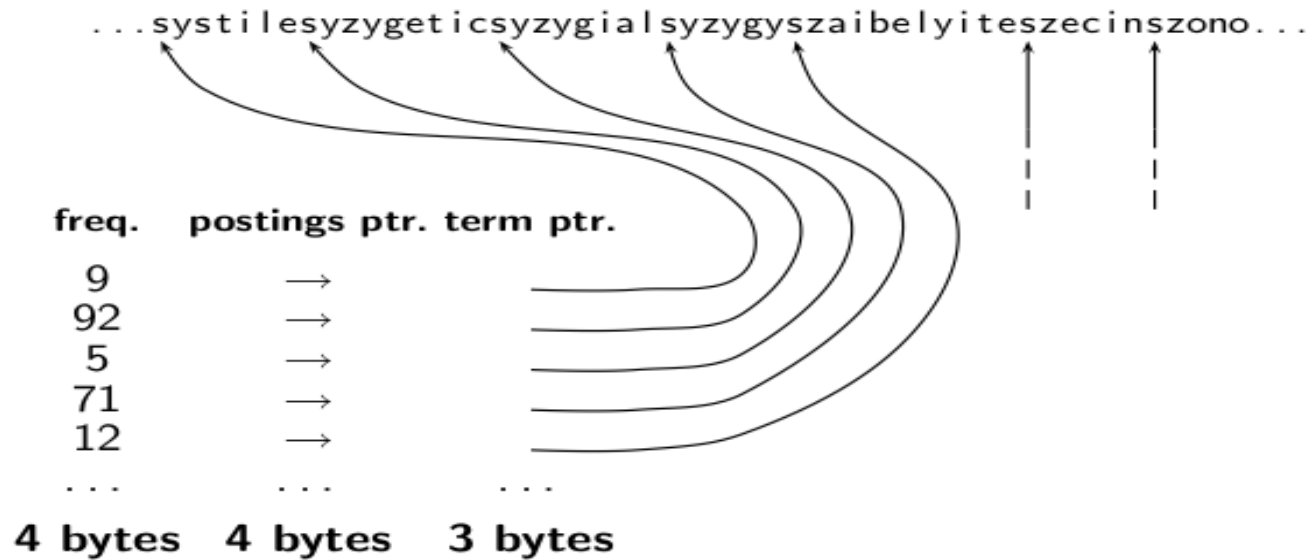
❖ همچنین هیچ راهی برای ذخیره دو عبارت با طول بیش از 20 کاراکتر مانند **HYDROCHLOROFLUOROCARBONS** and **SUPERCALIFRAGILISTICEXPIALIDOCIOUS** نداریم.

❖ میانگین طول هر اصطلاح در انگلیسی 8 کاراکتر است.

❖ چگونه می توانیم به طور میانگین از هشت کاراکتر به ازای هر اصطلاح استفاده کنیم؟



## Dictionary as a string



❖ 4 بایت برای فراوانی سند.

❖ 4 بایت برای اشاره‌گر به لیست پست‌ها.

❖ به طور میانگین 8 بایت برای هر اصطلاح در یک رشته است.

❖ این مشکل می‌تواند با ذخیره عبارات در لغت‌نامه به صورت یک **رشته بلند** از کاراکترها برطرف کرد.

❖ 3 بایت برای هر اشاره‌گر داخل رشته (need  $\log_2 8 \cdot 400000 < 24$  bits to resolve  $8 \cdot 400,000$  positions)

❖ در این طرح جدید فضایی برابر با  $400,000 \times (4 + 4 + 3 + 8) = 7.6\text{MB}$  برای لغت‌نامه Reuters-rcv1 نیاز داریم.

## Dictionary as a string with blocking

...7systile9syzygetic8syzygial6syzygy11szaibelyite6szecin...

freq.	postings ptr.	term ptr.
-------	---------------	-----------

9	→	
---	---	--

92	→	
----	---	--

5	→	
---	---	--

71	→	
----	---	--

12	→	
----	---	--

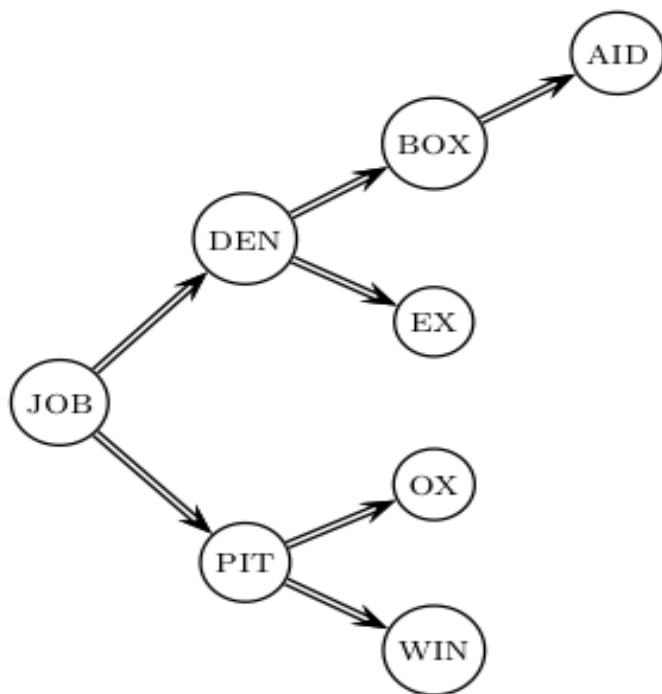
...

...

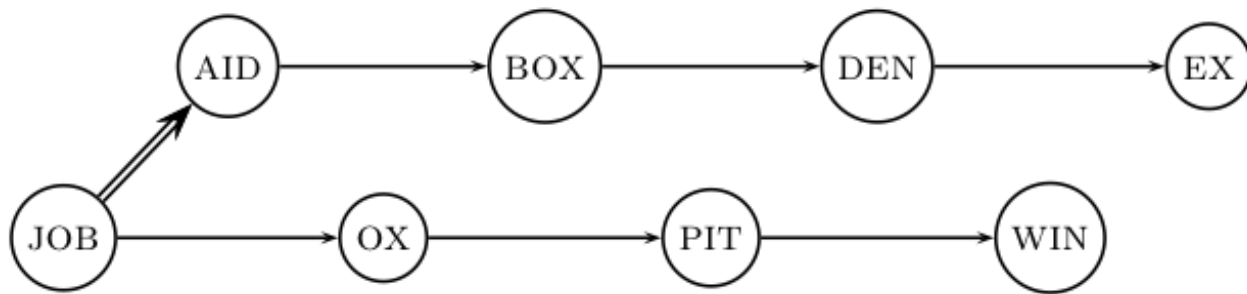
...

## فضای مورد نیاز برای لغت‌نامه در ذخیره‌سازی بلوکی

❖ ما می‌توانیم لغت‌نامه را با گروه‌بندی عبارات در رشته به بلوک‌هایی به اندازه‌ی  $K$  فشرده کرده و اشاره‌گر عبارت اول هر بلوک (مانند شکل اسلاید 24) نگهداریم. طول عبارت در رشته را، به عنوان بایت اضافی در ابتدای هر عبارت ذخیره می‌کنیم. بنابراین اشاره‌گرهای  $k-1$  عبارت را حذف می‌کنیم، اما به یک  $k$  بایت اضافی برای ذخیره طول هر عبارت نیاز داریم. برای  $k=4$ ، ما  $(k-1) \times 3 = 9$  بایت برای اشاره‌گر عبارات آزاد می‌کنیم، اما نیاز به  $K=4$  بایت اضافه برای طول عبارات داریم. بنابراین کل فضای مورد نیاز برای لغت‌نامه REUTERS-RCV1 تا 5 بایت در هر بلوک 4 عبارتی کاهش یافته است، یا در مجموع  $0.5 = 1.4 \times 5 \times 400,000$  مگابایت، که تا 7.1 مگابایت فضا را کاهش می‌دهد.



جستجو لغت نامه فشرده نشده



جستجو لغت نامه فشرده شده با بلوک بندی  $k=4$

## Front coding

One block in blocked compression ( $k = 4$ ) . . .

**8 a u t o m a t a 8 a u t o m a t e 9 a u t o m a t i c 10 a u t o m a t i o n**



. . . further compressed with front coding.

**8 a u t o m a t \* a 1 ◊ e 2 ◊ i c 3 ◊ i o n**

## فشرده سازی لغت نامه برای Reuters

data structure	size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
~, with blocking, $k = 4$	7.1
~, with blocking & front coding	5.9



## تمرین

❖ کدام پیشنهادهای باید در کدام طرف جلو استفاده شوند؟

❖ ورودی : فهرستی از اصطلاحات (لغتنامه واژگان)

❖ خروجی: فهرستی از پیشنهادها که در کدام طرف جلو استفاده خواهند شد.

## فشرده سازی پست ها

- ❖ فایل پست ها حداقل 10 برابر بزرگتر از دیکشنری ها هستند.
- ❖ نیاز کلیدی: ذخیره هر پست به صورت فشرده.
- ❖ منظور از پست همان docID است.
- ❖ برای Reuters (800,000 سند)، هنگام استفاده از اعداد صحیح (4-byte)، از 32 بیت به ازای هر docID استفاده می کنیم.
- ❖ یا می توانیم از 20 بیت به ازای هر docID ( $\log_2^{800000} \approx 19.6 < 20 \text{ bit per docID}$ ) استفاده کنیم.
- ❖ هدف: استفاده کمتر از 20 بیت به ازای هر docID.

## ایده اصلی: ذخیره سازی فاصله بین دو docID به جای خود آن ها

- ❖ هر لیست پستی به ترتیب افزایش docID مرتب شده است.
- ❖ برای ذخیره فاصله ها کافی است این کار را انجام دهید:  $283159 - 283154 = 5$ ,  $283202 - 283154 = 43$
- ❖ مثال استفاده از فاصله ها در لیست پست ها: **COMPUTER: 283154, 5, 43, ...**
- ❖ فاصله ها (Gaps) برای اصطلاحات مکرر کوچک هستند.
- ❖ به این ترتیب، فاصله های کوچک را می توانیم با کمتر از 20 بیت encode کنیم.

## Gap encoding

	encoding	postings list					
THE	docIDs	...	283042	283043	283044	283045	...
	gaps		1	1	1		...
COMPUTER	docIDs	...	283047	283154	283159	283202	...
	gaps		107	5	43		...
ARACHNOCENTRIC	docIDs	252000	500100				
	gaps	252000	248100				

# Variable length encoding

❖ هدف:

- برای کلمه ARACHNOCENTRIC از مدخل 20 bits per gap استفاده می‌کنیم.
  - برای کلمه the یا امثال آن از 1 bits per gap استفاده می‌کنیم.
- ❖ به منظور اجرای این کار، ما باید چند فرم از (کدگذاری طول متغیر) **variable length encoding** طراحی کنیم.
- ❖ کدهای طول متغیر از بیت های کمی برای فاصله های کوچک و بیت های زیاد برای فاصله های بیشتر استفاده می‌کند.

“

docIDs	824	829	215406
gaps		5	214577
VB	00000110 10111000	10000101	00001101 00001100 10110001
code			

VBENCODENUMBER( $n$ )

```
1  $bytes \leftarrow \langle \rangle$ 
2 while  $true$ 
3   do PREPEND( $bytes, n \bmod 128$ )
4     if  $n < 128$ 
5       then BREAK
6      $n \leftarrow n \text{ div } 128$ 
7    $bytes[\text{LENGTH}(bytes)] += 128$ 
8   return  $bytes$ 
```

VBENCODE( $numbers$ )

```
1  $bytestream \leftarrow \langle \rangle$ 
2 for each  $n \in numbers$ 
3   do  $bytes \leftarrow \text{VBENCODENUMBER}(n)$ 
4      $bytestream \leftarrow \text{EXTEND}(bytestream, bytes)$ 
5   return  $bytestream$ 
```

VBDECODE(*bytestream*)

```
1  numbers  $\leftarrow \langle \rangle$ 
2  n  $\leftarrow 0$ 
3  for i  $\leftarrow 1$  to LENGTH(bytestream)
4  do if bytestream[i] < 128
5      then n  $\leftarrow 128 \times n + \text{bytestream}[i]$ 
6      else n  $\leftarrow 128 \times n + (\text{bytestream}[i] - 128)$ 
7          APPEND(numbers, n)
8          n  $\leftarrow 0$ 
9  return numbers
```

کدگذاری و کدگشایی بایت متغیر. توابع *div* و *mod* تقسیم صحیح و باقیمانده را بعد از تقسیم صحیح به ترتیب مناسبه می‌کنند. *PREPEND* مولفه‌ای را به ابتدای لیست می‌افزاید، *EXTEND* یک لیست را بسط می‌دهد.



## سایر کدهای متغیر

- ❖ به جای بایت‌ها می‌توانیم از واحدهای دیگری مانند 32 بیتی (words)، 16 بیتی، 4 بیتی استفاده کنیم.
- ❖ استفاده از کدگذاری بایت‌های متغیر باعث می‌شود تا در صورت وجود gap های کوچک فضای حافظه هدر رود.
- ❖ کار اخیر روی کدهای word-aligned که به طور موثر تعداد متغیری از gap های داخل یک کلمه را “pack” می‌کنند.

## کدهای Gamma برای کدگذاری gap

- [illegible]

## کد Gamma

❖ این کد G gap را با دو مولفه طول (length) و آفست (offset) نگهداری می‌کند.

✓ منظور از آفست همان نمایش باینری G است با برش بیت با بالاترین ارزش.

• For example  $13 \rightarrow 1101 \rightarrow 101 = \text{offset}$

✓ طول همان طول آفست است.

• For 13 (offset 101), this is 3.

✓ کدگذاری طول با کد unary : 1110

✓ کد Gamma عدد 13 برابر با اتصال طول و آفست بدست آمده است : 1110101

## مثالی از کد Gamma

number	unary code	length	offset	$\gamma$ code
0	0			
1	10	0		0
2	110	10	0	10,0
3	1110	10	1	10,1
4	11110	110	00	110,00
9	1111111110	1110	001	1110,001
13		1110	101	1110,101
24		11110	1000	11110,1000
511		1111111110	11111111	111111110,11111111
1025		11111111110	0000000001	11111111110,0000000001

## تمرین

✓ برای عدد 130، کد بایت متغیر را محاسبه کنید.

✓ برای عدد 130، کد  $\gamma$  را محاسبه کنید.

## طول کد Gamma

❖ اندازه آفست برابر با  $\lceil \log_2 G \rceil$  بیت است.

❖ اندازه طول برابر با  $\lceil \log_2 G \rceil + 1$  بیت است.


❖ اندازه طول کل کد برابر با  $2 \times \lceil \log_2 G \rceil + 1$  بیت است.

❖ کدهای gamma همیشه طولی فرد داشته‌اند.

❖ کدهای gamma با فاکتور 2 نسبت به طول کدگذاری بهینه  $G / \log_2 G$  است.

• این مقدار بهینه را از فرضی که  $2^n$  فاصله بین 1 و  $2^n$  است، برداشت می‌کنیم،

اما همیشه این مدنظر نیست.



❖ کدهای gamma پیشنهاد آزاد هستند، یعنی هیچ کد پیشنهاد دیگری نیست.

❖ کدگذاری با ضریب 3 بهینه است. (و با ضریب 2 مفروضات اضافی ایجاد می‌شود).

❖ این نتیجه مستقل از توزیع شکاف هاست.

❖ ما می‌توانیم کدهای gamma را برای هر توزیعی استفاده کنیم. (این کد کلی است)

❖ کد gamma بدون پارامتر است.


## هم ترازى

- ❖ ماشین ها سرمدى برای اندازه کلمه دارند. (8, 16, 32 bits)
- ❖ فشردہ سازی و دستکاری **granularity of bits** می تواند کند باشد.
- ❖ کدگذاری بایت متغیر هم تراز است و به همین خاطر بسیار کارآمد است.
- ❖ بایت متغیر به صورت مفهومی اندکی در هزینه فضای اضافی ساده تر است



“

data structure	size in MB
لغت نامه ، عرض ثابت	11.2
لغت نامه ، اشاره گر عبارات در رشته	7.6
به صورت بلوکی ، $k = 4$	7.1
به صورت بلوکی و با کدگذاری به طرف جلو ، ~	5.9
مجموعه (text, xml markup etc)	3600.0
مجموعه (text)	960.0
ماتریس تلاقی عبارت	40,000.0
فشرده نشده (32-bit words)	400.0
فشرده نشده (20 bits)	250.0
کدگذاری بایت متغیر	116.0
کدگذاری گاما	101.0



	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*. Entry is 0 if term doesn't occur. Example: CALPURNIA doesn't occur in *The tempest*.

در صورت وجود اصطلاح، ورودی 1 است، برای مثال: Calpurnia در نمایشنامه Julius Caesar وجود دارد. در صورت عدم وجود اصطلاح، ورودی 0 است، برای مثال: Calpurnia در نمایشنامه The Tempest وجود ندارد.

## خلاصه

- ❖ اکنون می‌توانیم یک شافص برای بازیابی بولی بسیار کارآمد ایجاد کنیم که فضای بسیار کارآمدی دارد.
- ❖ فقط 10-15٪ از کل اندازه متن در مجموعه وجود دارد.
- ❖ به هر حال، اطلاعات موقعیتی و تکراری را نادیده گرفتیم.
- ❖ به همین دلیل، صرفه جویی در فضا در واقعیت کمتر است.



تشکر

سوال؟

[a.golzari@azaruniv.ac.ir](mailto:a.golzari@azaruniv.ac.ir)

[a.golzari@tabrizu.ac.ir](mailto:a.golzari@tabrizu.ac.ir)

<https://github.com/Amin-Golzari-Oskouei>