Useful Things To Know

Chapter 1.3 of Textbook 05 “Mining of Massive Datasets”

Importance of Words in Document

Term Frequency times Inverse Document Frequency (TF.IDF) is: $\mathrm{TF.IDF}_{ij} = \mathrm{TF}_{ij} \times \mathrm{IDF}_{i}$

Suppose f_{ij} to be the frequency (number of occurrences) of term (word) i in document j, the *term frequency* TF_{ij} is:

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Suppose term i appears in n_i of N documents of the collection under consideration, then the *inverse document frequency* is

$$IDF_i = \log_2\left(\frac{N}{n_i}\right)$$

Importance of Words in Document (cont'd)

EXAMPLES

Suppose our collection includes $2^{20}=1048576$ documents, and word w appears in $2^{10}=1024$ of them, then $IDF_w=\log_2(2^{20}/2^{10})=10$,

consider a document j in which w appears 20 times, and that is the maximum number of times in which any words appear (after removing *stop words*), and then $TF_{wj}=1$, and $TF.IDF$ score for w in document j is 10,

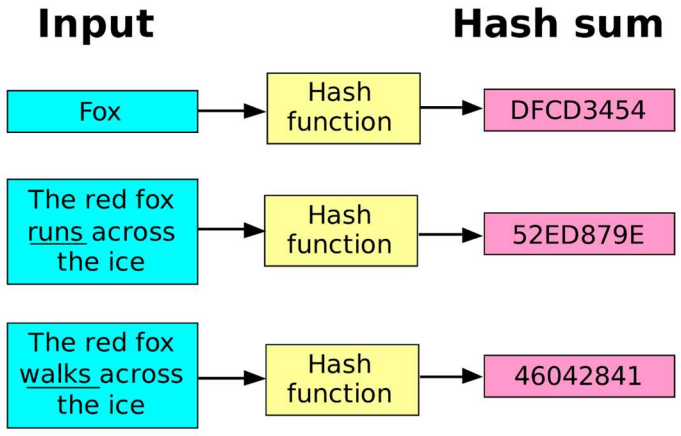
suppose in document k, the word w appears once, and the maximum number of occurrences of any word in this document is 20, then $TF_{wk}=1/20$, and the $TF.IDF$ score for w in document k is $1/2$.

Hash Functions

A hash function *h* takes a *hash-key* value as an argument and produces a bucket *number* as a result, i.e., *h* "*randomizes*" hash-keys

A common and simple one: $h(x) = x \bmod B$

It is preferred to choose B to be an even, odd, or prime number?



Hash Functions (cont'd)

What if hash-keys are not integers?

All data type have values that are composed of *bits*, sequences of bits could be interpreted as integers

- ASCII code

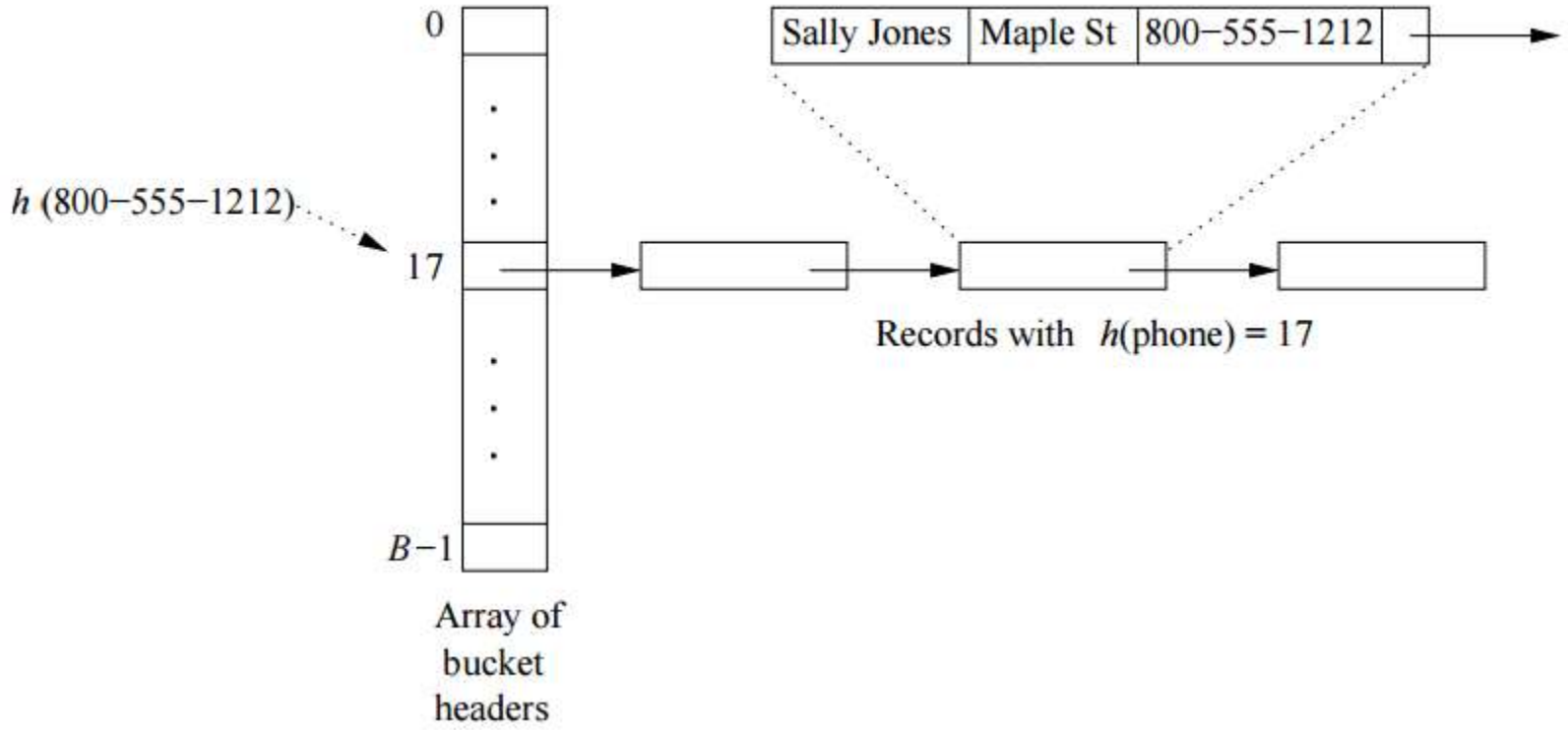
Record, array, set, bag of elements - recursively convert each component to an integer, sum up, and then divide by B

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

Indexes

An index is a data structure that makes it efficient to retrieve objects given the value of one or more elements of those objects, …*retrieve records efficiently*…there are ways to implement indexes



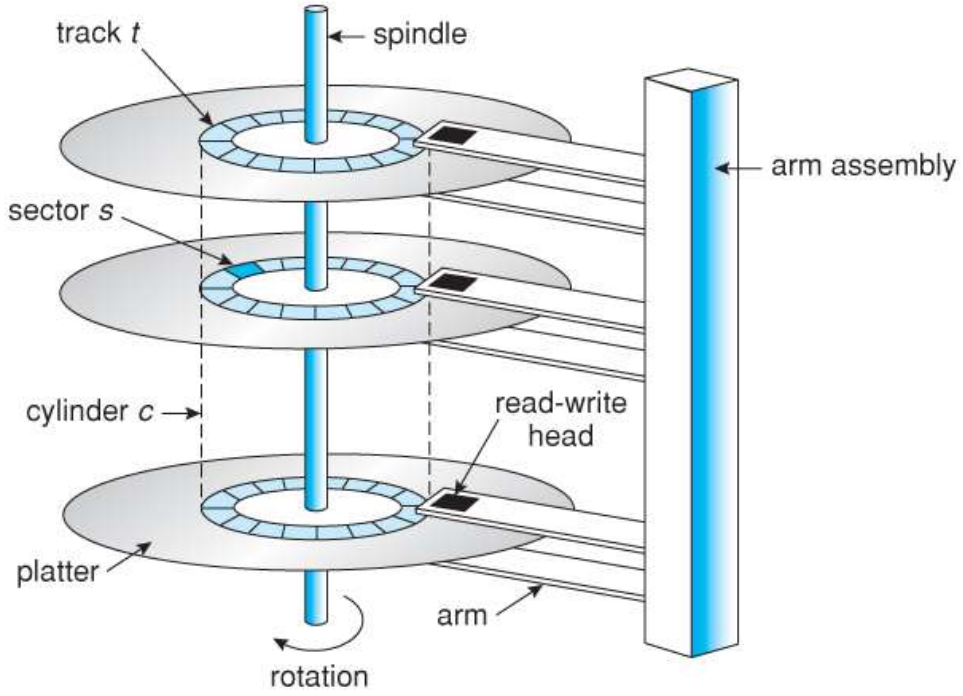
Secondary Storage

Non-volatile memory (does not lose stored data when the device is powered down) that is not directly accessible by the CPU

Slower than main memory, i.e., RAM, a disk cannot transfer data to main memory at more than a *hundred million bytes* per second, no matter how that data is organized

Secondary Storage (cont'd)

OS organizes secondary memory as *blocks*, by organizing our data so that related data is on a single cylinder (the collection of blocks reachable at a fixed radius from the center of the disk, and therefore accessible without moving the disk head), we may be able to improve performance



The Base of Natural Logarithms

Properties of the constant $e = \lim_{x \to \infty} (1 + \frac{1}{x})^x$

Taylor expansion $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots$

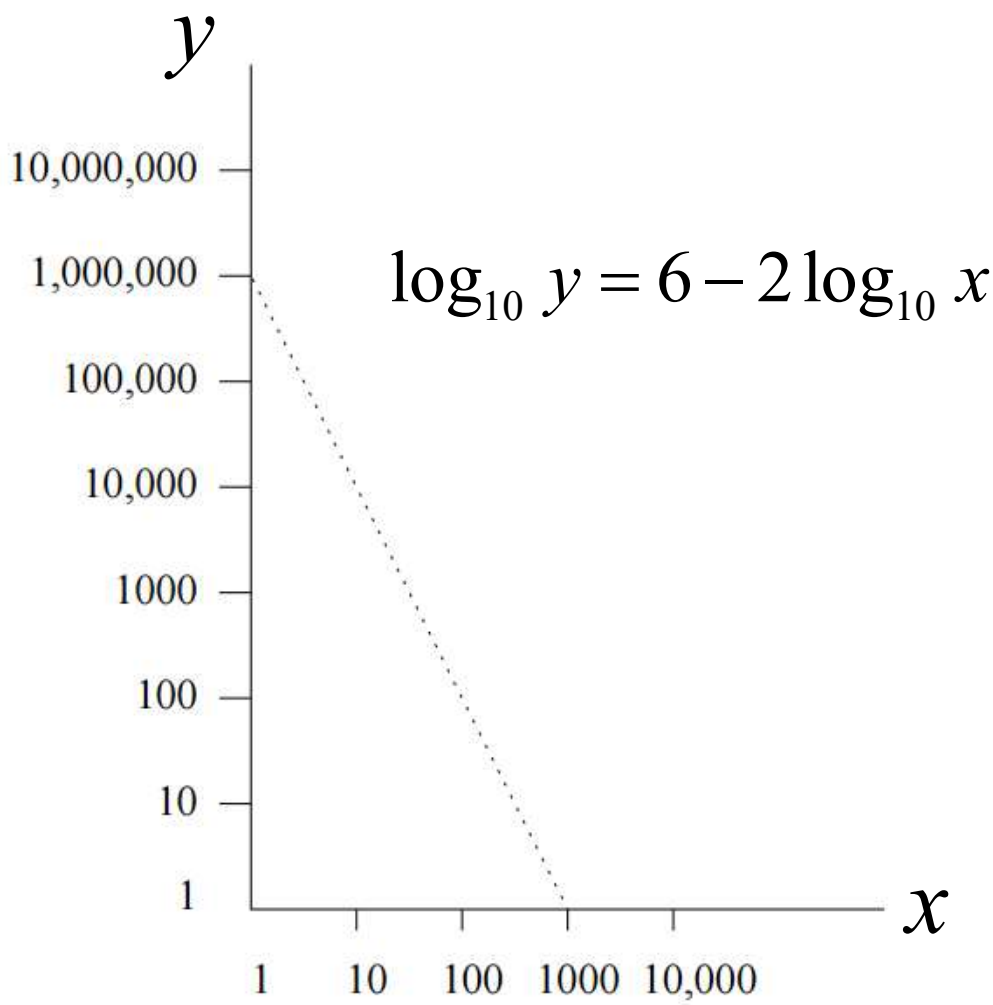
Approximation examples

$$\begin{cases} a = \frac{1}{x} \\ x = \frac{1}{a} \end{cases} \qquad \begin{cases} e = \lim_{x \to \infty} (1 + \frac{1}{x})^x \\ a \text{ is small, } x \text{ is large} \end{cases}$$

$$(1+a)^b = (1+a)^{(\frac{1}{a})(ab)} = (1+\frac{1}{x})^{xab} = \left((1+\frac{1}{x})^x \right)^{ab} = e^{ab}$$

Power Laws

Linear relationship between the logarithms of the variables

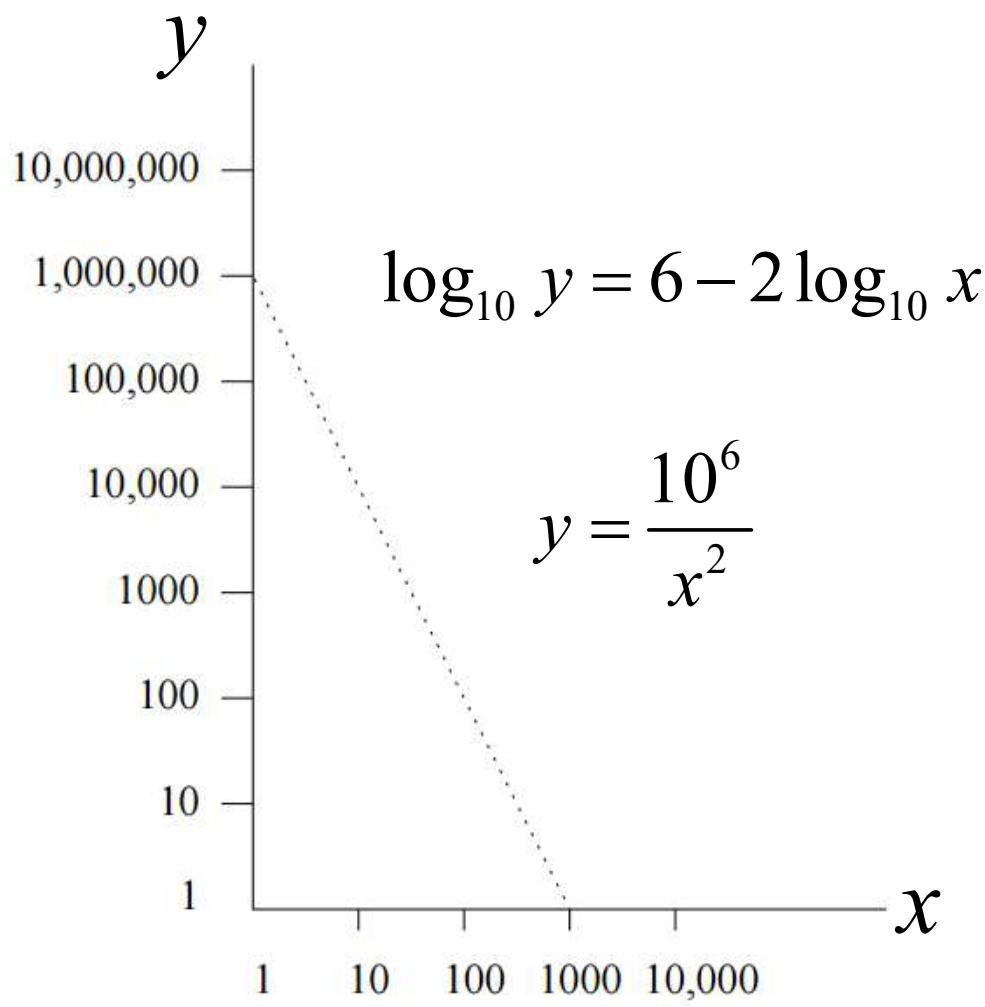


Power Laws (cont'd)

Example

x - rank of books by sale

y - number of sales of the xth best-selling book over some period



Power Laws (cont'd)

Example

x - rank of books by sale

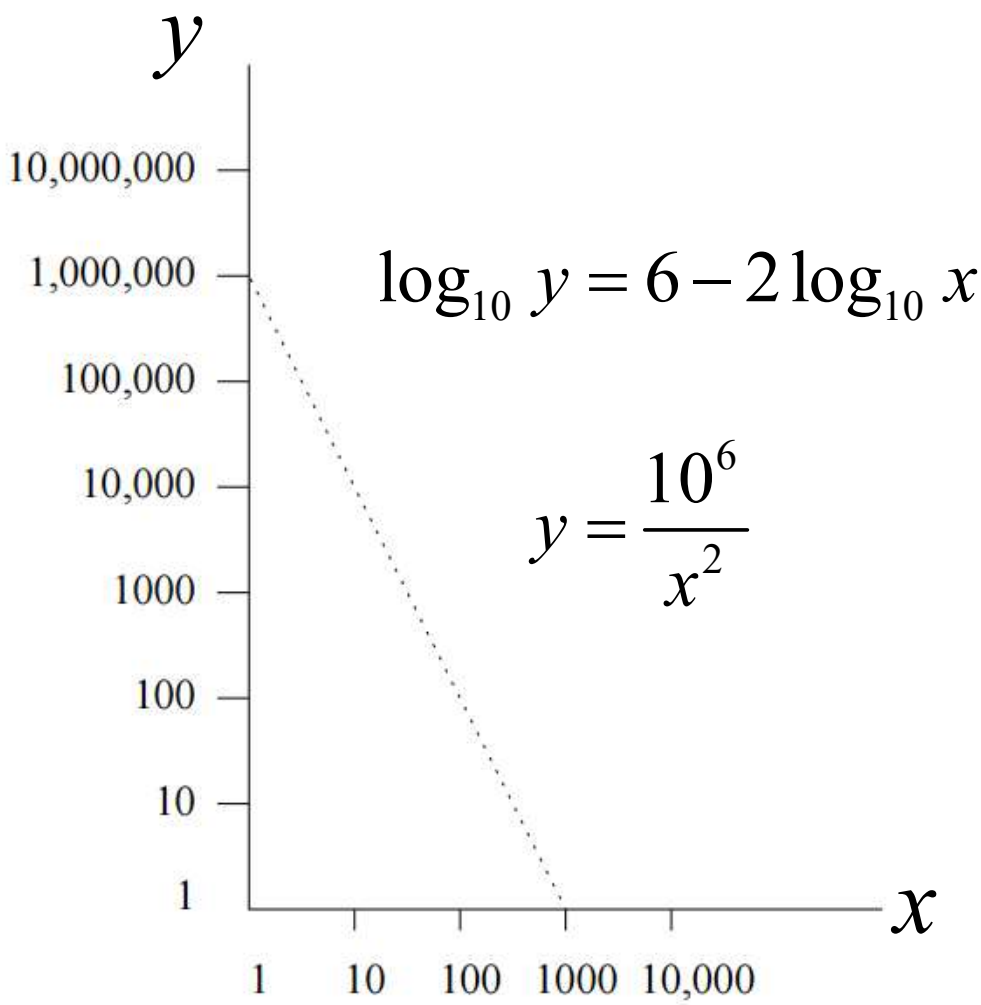
y - number of sales of the xth best-selling book over some period

How many copies are sold

for the best-selling book?

the 10th best-selling one?

the 100th best-selling one?



$$\log_{10} y = 6 - 2\log_{10} x$$

$$y = \frac{10^6}{x^2}$$

Power Laws (cont'd)

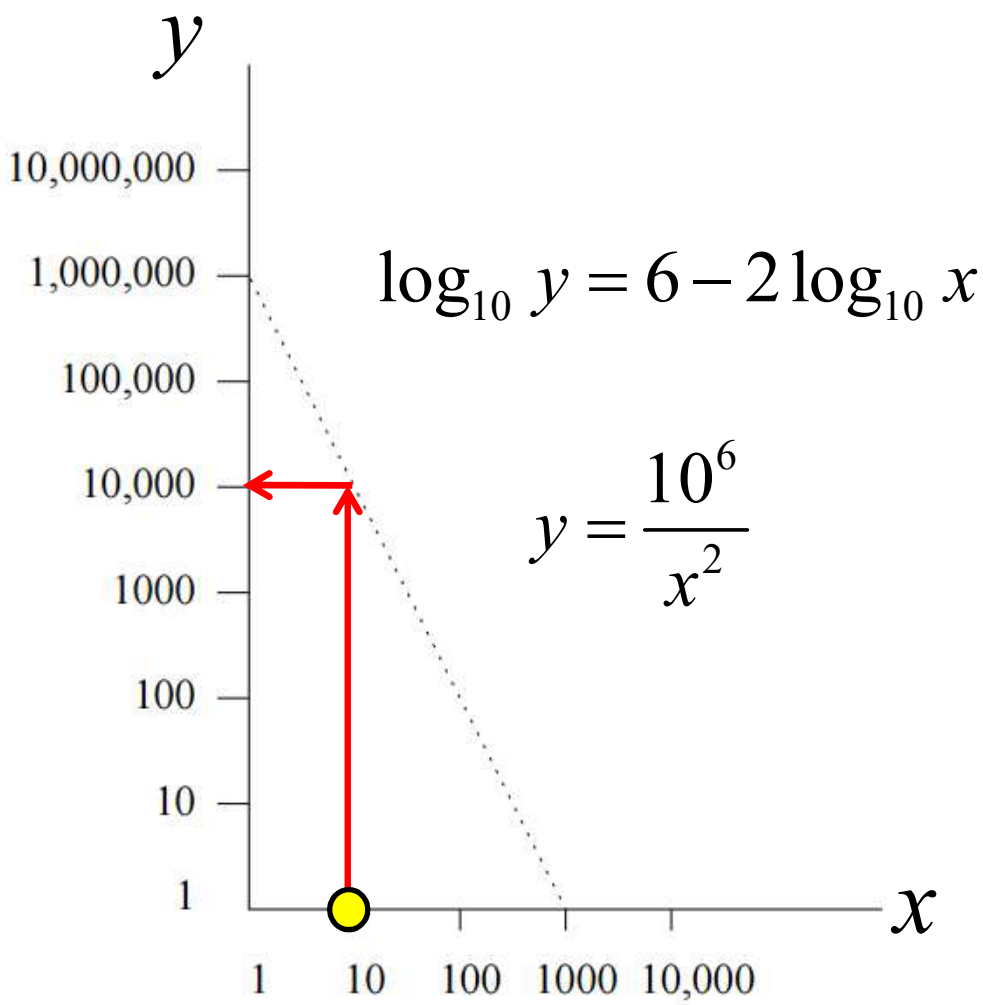
Example

x - rank of books by sale

y - number of sales of the xth best-selling book over some period

How many copies are sold for the best-selling book?

the 10th best-selling one?



Power Laws (cont'd)

Example

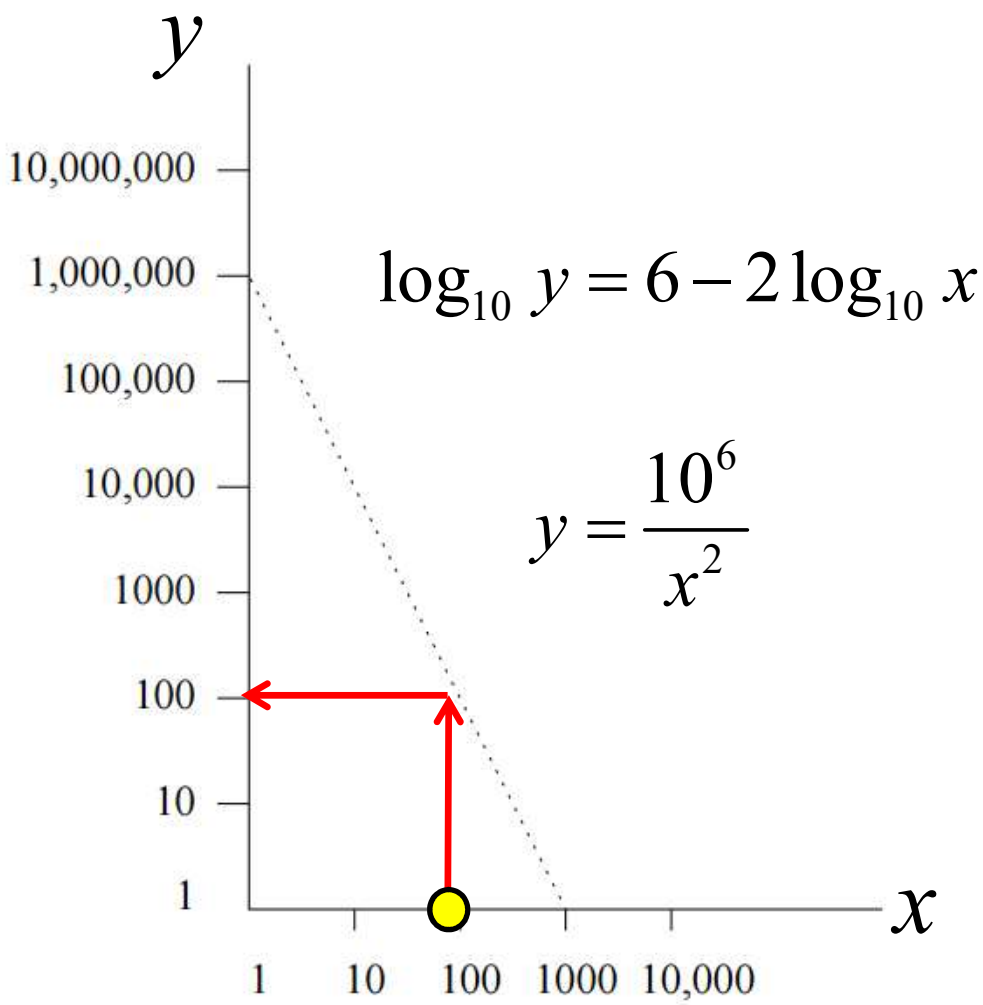
x - rank of books by sale

y - number of sales of the xth best-selling book over some period

How many copies are sold for the best-selling book?

the 10th best-selling one?

the 100th best-selling one?



Power Laws (cont'd)

General form: $\log y = b + a \log x$

if the base is e

$y = e^{b} e^{a \log x}$

e^{b} is just "some constant"

$y = e^{b} e^{a \log x}$

a and c are constants

$y = cx^{a}$

Power Laws (cont'd)

Application examples

- Node Degree in the Web Graph: order all pages by the number of *inlinks* to the page
- Sales of Products: order products, say books at Amazon.com, by their sales over the past year
- Size of Web Sites: count the number of pages at Web sites, and order sites by the number of pages
- Zipf's Law: frequency of words in a collection of documents
- …