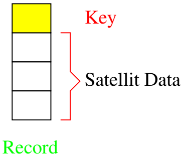
Hash function

RA-MIRI QT Curs 2020-2021

Data Structures: Reminder

Given a universe $\mathcal{U}$, a dynamic set of records, where each record:



- Array
- Linked List (and variations)
- Stack (LIFO): Supports push and pop
- Queue (FIFO): Supports enqueue and dequeue
- Deque: Supports push, pop, enqueue and dequeue
- Heaps: Supports insertions, deletions, find Max and MIN
- Hashing

Recall Dynamic Data Structures

DICTIONARY

Data structure for maintaining $\mathcal{S} \subset \mathcal{U}$ together with operations:

- Search $(\mathcal{S}, k)$: decide if $k \in \mathcal{S}$
- Insert $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \cup \{k\}$
- Delete $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \backslash \{k\}$

PRIORITY QUEUE

Data structure for maintaining $\mathcal{S} \subset \mathcal{U}$ together with operations:

- Insert $(\mathcal{S}, k)$: $\mathcal{S} := \mathcal{S} \cup \{k\}$
- Maximum $(\mathcal{S})$: Returns element of $\mathcal{S}$ with largest k
- Extract-Maximum $(\mathcal{S})$: Returns and erase from $\mathcal{S}$ the element of $\mathcal{S}$ with largest k

Priority Queue

Linked Lists:

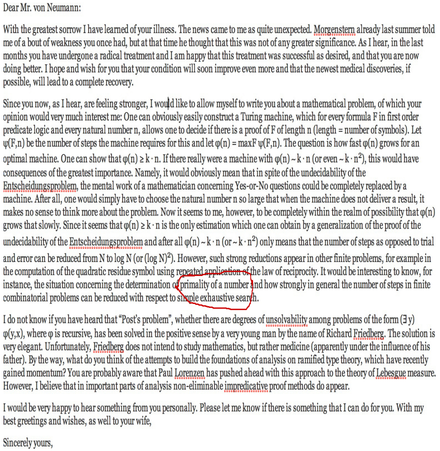
- *INSERT:* $O(n)$
- *EXTRACT-MAX:* $O(1)$

Heaps:

- *INSERT:* $O(\lg n)$
- *EXTRACT-MAX:* $O(\lg n)$

Using a Heap is a good compromise between fast insertion and slow extraction.

String Matching



Dear Mr. von Neumann:

With the greatest sorrow I have learned of your illness. The news came to me as quite unexpected. Morgenstern already last summer told me of a bout of weakness you once had, but at that time he thought that this was not of any greater significance. As I hear, in the last months you have undergone a radical treatment and I am happy that this treatment was successful as desired, and that you are now doing better. I hope and wish for you that your condition will soon improve even more and that the newest medical discoveries, if possible, will lead to a complete recovery.

Since you now, as I hear, are feeling stronger, I would like to allow myself to write you about a mathematical problem, of which your opinion would very much interest me: One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\psi(F,n)$ be the number of steps the machine requires for this and let $\varphi(n) = \max_F \psi(F,n)$. The question is how fast $\varphi(n)$ grows for an optimal machine. One can show that $\varphi(n) \geq k \cdot n$. If there really were a machine with $\varphi(n) \sim k \cdot n$ (or even $\sim k \cdot n^2$), this would have consequences of the greatest importance. Namely, it would obviously mean that in spite of the undecidability of the Entscheidungsproblem, the mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine. After all, one would simply have to choose the natural number n so large that when the machine does not deliver a result, it makes no sense to think more about the problem. Now it seems to me, however, to be completely within the realm of possibility that $\varphi(n)$ grows that slowly. Since it seems that $\varphi(n) \geq k \cdot n$ is the only estimation which one can obtain by a generalization of the proof of the undecidability of the Entscheidungsproblem and after all $\varphi(n) \sim k \cdot n$ (or $\sim k \cdot n^2$) only means that the number of steps as opposed to trial and error can be reduced from N to log N (or $(\log N)^2$). However, such strong reductions appear in other finite problems, for example in the computation of the quadratic residue symbol using repeated application of the law of reciprocity. It would be interesting to know, for instance, the situation concerning the determination of primality of a number and how strongly in general the number of steps in finite combinatorial problems can be reduced with respect to simple exhaustive search.

I do not know if you have heard that "Post's problem", whether there are degrees of unsolvability among problems of the form $(\exists y)\, \varphi(y,x)$, where φ is recursive, has been solved in the positive sense by a very young man by the name of Richard Friedberg. The solution is very elegant. Unfortunately, Friedberg does not intend to study mathematics, but rather medicine (apparently under the influence of his father). By the way, what do you think of the attempts to build the foundations of analysis on ramified type theory, which have recently gained momentum? You are probably aware that Paul Lorenzen has pushed ahead with this approach to the theory of Lebesgue measure. However, I believe that in important parts of analysis non-eliminable impredicative proof methods do appear.

I would be very happy to hear something from you personally. Please let me know if there is something that I can do for you. With my best greetings and wishes, as well to your wife,

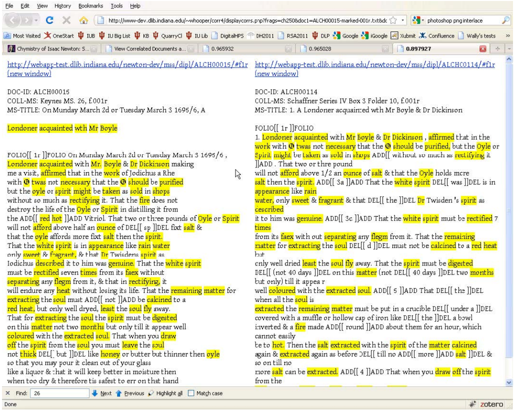
Sincerely yours,

Search: primality of a number

Given a text, find a subtext

- Given two texts, find common subtexts (plagiarism)
- Given two genomes, find common subchains (consecutive characters)

Document similarity


File Edit View History Bookmarks Tools Help
http://www-dev.dlib.indiana.edu/~whooper/corr4/displaycorrs.php?frags=ch250&doc1=ALCH00015-marked-001r.txt&dc
photoshop png interlace
Most Visited OneStart IUB IU Big List KB QuarryCl IU Lib DigitalHPS DH2011 RSA2011 DLP Google iGoogle Xubmit Confluence Wally's tests
Chymistry of Isaac Newton: S...
View Correlated Documents a...
0.965932
0.965028
0.897927

http://webapp-test.dlib.indiana.edu/newton-dev/mss/dipl/ALCH00015/#f1r
(new window)

DOC-ID: ALCH00015
COLL-MS: Keynes MS. 26, f.001r
MS-TITLE: On Munday March 2d or Tuesday March 3 1695/6, A

Londoner acquainted wth Mr Boyle

FOLIO[[1r]]FOLIO On Munday March 2d or Tuesday March 3 1695/6 ,
Londoner acquainted with Mr. Boyle & Dr Dickinson making
me a visit, affirmed that in the work of Jodichus a Rhe
with Φ twas not necessary that the Φ should be purified
but the oyle or spirit might be taken as sold in shops
without so much as rectifying it. That the fire does not
destroy the life of the Oyle or Spirit in distilling it from
the ADD[[red hot]]ADD Vitriol. That two or three pounds of Oyle or Spirit
will not afford above half an ounce of DEL[[sp]]DEL fixt salt &
that the oyle affords more fixt salt then the spirit.
That the white spirit is in appearance like rain water
only sweet & fragrant, & that Dr Twisdens spirit as
Iodichus described it to him was genuine. That the white spirit
must be rectified seven times from its faex without
separating any flegm from it, & that in rectifying, it
will endure any heat without losing its life. That the remaining matter for
extracting the soul must ADD[[not]]ADD be calcined to a
red heat, but only well dryed, least the soul fly away.
That for extracting the soul the spirit must be digested
on this matter not two months but only till it appear well
coloured with the extracted soul. That when you draw
off the spirit from the soul you must leave the soul
not thick DEL[[but]]DEL like honey or butter but thinner then oyle
so that you may pour it clean out of your glass
like a liquor & that it will keep better in moisture then
when too dry & therefore tis safest to err on that hand

http://webapp-test.dlib.indiana.edu/newton-dev/mss/dipl/ALCH00114/#f1r
(new window)

DOC-ID: ALCH00114
COLL-MS: Schaffner Series IV Box 3 Folder 10, f.001r
MS-TITLE: 1. A Londoner acquainted wth Mr Boyle & Dr Dickinson

FOLIO[[1r]]FOLIO
1. Londoner acquainted with Mr Boyle & Dr Dickinson , affirmed that in the work with Φ twas not necessary that the Φ should be purified, but the Oyle or Spirit might be taken as sold in shops ADD[[without so much as rectifying it]]ADD . That two or thre pound
will not afford above 1/2 an ounce of salt & that the Oyle holds more salt then the spirit. ADD[[3a]]ADD That the white spirit DEL[[was]]DEL is in appearance like rain
water, only sweet & fragrant & that DEL[[the]]DEL Dr Twisden 's spirit as described
it to him was genuine. ADD[[3c]]ADD That the white spirit must be rectified 7 times
from its faex with out separating any flegm from it. That the remaining matter for extracting the soul DEL[[d]]DEL must not be calcined to a red heat but
only well dried least the soul fly away. That the spirit must be digested
DEL[[(not 40 days]]DEL on this matter (not DEL[[40 days]]DEL two months but only) till it appea r
well coloured with the extracted soul. ADD[[5]]ADD That DEL[[the]]DEL when all the soul is
extracted the remaining matter must be put in a crucible DEL[[under a]]DEL covered with a muffle or hollow cap of iron like DEL[[the]]DEL a bowl inverted & a fire made ADD[[round]]ADD about them for an hour, which cannot easily
be to hot. Then the salt extracted with the spirit of the matter calcined again & extracted again as before DEL[[till no ADD[[more]]ADD salt]]DEL & so on till no
more salt can be extracted. ADD[[4]]ADD That when you draw off the spirit from the

Finding similar documents in the WWW

- Proliferation of almost identical documents
- Approximately 30% of the pages on the web are (near) duplicates.
- Another way to find plagiarism

Hashing functions

Data Structure that supports *dictionary* operations on an universe of numerical keys.

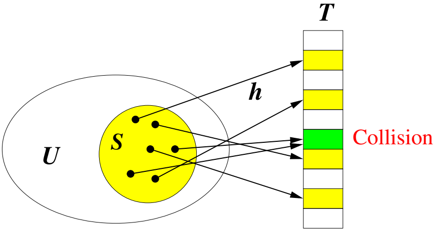
Notice the number of possible keys represented as 64-bit integers is $2^{63} = 18446744073709551616$.

Tradeoff *time/space*

Define a hashing table $T[0, \dots, m-1]$

a hashing function $h : \mathcal{U} \to T[0, \dots, m-1]$

Hans P. Luhn
(1896-1964)



Simple uniform hashing function.

A good hashing function must have the property that $\forall k \in \mathcal{U}$, $h(k)$ must have the same probability of ending in any $T[i]$.

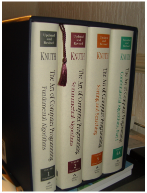
Given a hashing table T with m slots, we want to store $n = |\mathcal{S}|$ keys, as maximum.

Important measure: load factor $\alpha = n/m$, the average number of keys per slot.

The performance of hashing depends on how well h distributes the keys on the m slots: h is simple uniform if it hash any key *with equal probability* into any slot, independently of where other keys go.

How to choose h?

Advice: For an exhaustive treaty on Hashing: D. Knuth, Vol. 3 of *The Art of computing programming*



h depends on the type of key:

- If $k \in \mathbb{R}, 0 \leq k \leq 1$ we can use $h(k) = \lfloor mk \rfloor$.

- If $k \in \mathbb{R}, s \leq k \leq t$ scale by $1/(t-s)$, and use the previous methode: $h(k/(t-s)) = \lfloor mk/(t-s) \rfloor$.

The division method

Choose m prime and as far as possible from a power,

$$\boxed{h(k) = k \mod m}.$$

Fast ($\Theta(1)$) to compute in most languages ($k\%m$)!

Be aware: if $m = 2^r$ the hash does not depend on all the bits of K

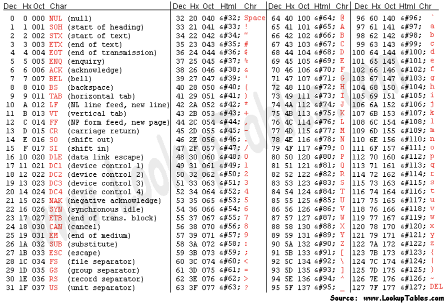
If $r = 6$ with $k = 1011000111\underbrace{011010}_{=h(k)}$

$(45530 \mod 64 = 858 \mod 64)$

- In some applications, the keys may be very large, for instance with alphanumeric keys, which must be converted to ascii:

Example: *averylongkey* is converted via ascii:

$$97 \cdot 128^{11} + 118 \cdot 128^{10} +$$
$$101 \cdot 128^{9} + 114 \cdot 128^{8}$$
$$+121 \cdot 128^{7} + 108 \cdot 126^{6}$$
$$+111 \cdot 128^{5} + 110 \cdot 128^{4}$$
$$+103 \cdot 128^{3} + 107 \cdot 128^{2}$$
$$+101 \cdot 128^{1} + 121 \cdot 128^{0} = n$$



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

which has 84-bits!

Recall mod arithmetic : for $a, b, m \in \mathbb{Z}$,
$(a+b) \bmod m = (a \bmod m + b \bmod m) \bmod m$
$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$
$a(b+c) \bmod m = ab \bmod m + ac \bmod m$
If $a \in \mathbb{Z}_m$ $(a \bmod m) \bmod m = a \bmod m$

Horner's rule: Given a specific value x_0 and a polynomial $A(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 X + \cdots + a_n x^n$ to evaluate $A(x_0)$ in $\Theta(n)$ steps:

$$A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(a_{n-1} + a_n x_0)))$$

How to deal with large n

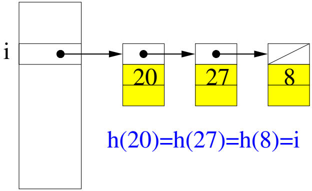
For large n, to compute $h = n \mod m$, we can use mod arithmetic + Horner's method:

$$
\begin{aligned}
&((((((((((97 \cdot 128 + 118) \cdot 128 + 101) \cdot 128 + 114) \cdot 128 + 121) \\
&\cdot 128 + 111) \cdot 128 + 110) \cdot 128 + 103) \cdot 128 + 107) \\
&\cdot 128 + 101) \cdot 128 + 121 \mod m \\
&= (((((((((\underbrace{\underbrace{(97 \cdot 128 + 118 \mod m)}_{} \cdot 128) \mod m + 101)}_{} \cdot \ldots))))))
\end{aligned}
$$

Collision resolution: Separate chaining

For each table address, construct a linked list of the items whose keys hash to that address.

- Every key goes to the same slot
- Time to explore the list = length of the list



Cost of average analysis of chaining

The cost of the dictionary operations using hashing:

- Insertion of a new key: $\Theta(1)$.
- Search of a key: $O($ length of the list)
- Deletion of a key: $O($ length of the list).

Under the hypothesis that h is *simply uniform hashing*, each key x is equally likely to be hashed to any slot of T, independently of where other keys are hashed

Therefore, the expected number of keys falling into $T[i]$ is $\alpha = n/m$.

Cost of search

For an unsuccessful search (x is not in T) therefore we have to explore the all list at $h(x) \to T[i]$ with an the expected time to search the list at $T[i]$ is $O(1+\alpha)$.
(α of searching the list and $\Theta(1)$ of computing $h(x)$ and going to slot $T[i]$)

For an successful search search, we can obtain the same bound, (most of the cases we would have to search a fraction of the list until finding the x element.)

Therefore we have the following result: Under the assumption of simple uniform hashing, in a hash table with chaining, an unsuccessful and successful search takes time $\Theta(1+\frac{n}{m})$ on the average.

Notice that if $n = \theta(m)$ then $\alpha = O(1)$ and search time is $\Theta(1)$.

Universal hashing: Motivation

For every deterministic hash function, there is a set of bad instances.

An adversary can arrange the keys so your function hashes most of them to the same slot.

Create a set $\mathcal{H}$ of hash functions on $\mathcal{U}$ and choose a hashing function at random and independently of the keys.

Must be careful once we choose one particular hashing function for a given key, we always use the same function to deal with the key.

Universal hashing

Let $\mathcal{U}$ be the universe of keys and let $\mathcal{H}$ be a collection of hashing functions with hashing table $T[0, \ldots, m-1]$, $\mathcal{H}$ is universal if $\forall x, y \in \mathcal{U}, x \neq y$, then

$$|\{h \in \mathcal{H} \mid h(x) = h(y)\}| \leq \frac{|\mathcal{H}|}{m}.$$

In an equivalent way, $\mathcal{H}$ is *universal* if $\forall x, y \in \mathcal{U}, x \neq y$, and for any h chosen uniformly from $\mathcal{H}$, we have

$$\mathbf{Pr}\left[h(x) = h(y)\right] \leq \frac{1}{m}.$$

Universality gives good average-case behaviour

Theorem

If we pick a u.a.r. h from a universal $\mathcal{H}$ and build a table using and hash n keys to T with size m, for any given key x let Z_x be a random variable counting the number of collisions with others keys y in T.

$$\mathbf{E}\left[\#\mathit{collisions}\right] \leq n/m.$$

Construction of a universal family: $\mathcal{H}$

To construct a family $\mathcal{H}$ for $N = \max\{\mathcal{U}\}$ and $T[0, \ldots, m-1]$:

- $\mathcal{H} = \emptyset$.
- *Choose a prime p, $N \leq p \leq 2N$. Then $\mathcal{U} \subset \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$.*
- *Choose independently and u.a.r. $a \in \mathbb{Z}_p^+$ and $b \in \mathbb{Z}_p$. Given a key x define $h_{a,b}(x) = \underbrace{((ax+b) \mod p)}_{g_{a,b}(x)} \mod m$.*
- $\mathcal{H} = \{h_{a,b} | a, b \in \mathbb{Z}_p, a \neq 0\}$.

Example: $p = 17, m = 6$ we have $\mathcal{H}_{17,6} = \{h_{a,b} : a \in \mathbb{Z}_p^+, b \in \mathbb{Z}_p\}$
if $x = 8$, $a = 3, b = 4$ then
$h_{3,4}(8) = ((3 \cdot 8 + 4) \mod 17) \mod 6 = 5$

Properties of $\mathcal{H}$

1. $h_{ab} : \mathbb{Z}_p \to \mathbb{Z}_m$.
2. $|\mathcal{H}| = p(p-1)$. (We can select a in $p-1$ ways and b in p ways)
3. Specifying an $h \in \mathcal{H}$ requires $O(\lg p) = O(\lg N)$ bits.
4. To choose $h \in \mathcal{H}$ select a, b independently and u.a.r. from $\mathbb{Z}_p^+$ and $\mathbb{Z}_p$.
5. Evaluating $h(x)$ is fast.

Theorem

The family $\mathcal{H}$ is universal.

For the proof:

Chapter 11 of Cormen. Leiserson, Rivest, Stein: *An introduction to Algorithms*