Hashing Data Structure

↳ Randomized DS

Easy - Moderate

- ↳ Array ⟷ List ✓
- ↳ Linked List ✓
- ↳ Skip List — ADSA (Probabilistic DS) (Randomized DS)
- ↳ Stack ✓
- ↳ Queue ✓
- ↳ Hashing ✓
- ↳ Heap ✓

Python
↳ Dict

Java
↳ Hashmap

Advance

- ↳ Tree
- ↳ Graph
- ↳ Skip List

⟶ Lookups, searching ⇒ $\Theta(1)$ ⇓ Constant

Hashing → (key, value)
⇓
unique

HashTable

key	value
0	
1	
2	2
3	
4	4
5	
6	6
7	7
8	
9	9

Direct Addressing Table

2, 4, 7, 9, 6

2^{10}

2^{10}

Lot of memory wastage ↙

Hash functions

March 2020 → Hash function → 2 numeric key

0		
1		
2	March 2020	768

Type of Hash function

1) Divison Modulo Hash function * Imp

Key = value % m

Remainder

m = size of hash table

m = 10

20 % 10 = 0

Hash Table

0	20
1	
2	52
3	43
4	74
5	
6	56
7	47
8	
9	69

20, 47, 57

52, 69, 79

74, 56, 43

57 collisions

March 2020 → summation % m = key

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

www.alpharithms.com

↳ Collision Error

2) Mid-square

0 to 999

m = 1000

Value = 2042 → key = 697

$(2042)^2 = 4169764$

m = 100

0 to 99

3) Digit folding Method

Value = 124 655 12

H(value) = 124 + 655 + 12

= 791 → key

Good Hash function → 1) Lesser collisions

2) uniform distribution of data in your hash table

3) Easy to understand & simple to compute

https://docs.python.org/3/library/operator.html → **Study Reference**

↳ collision Resolution Techniques

- Chaining (extra space)
  - ↳ Linked List
- Open Addressing (no extra space)
  - change in hash function
    1) Linear Probing
    2) Quadratic Probing
    3) Double Hashing