Hashing

Friday, December 10, 2021 10:18 AM

LINEAR SEARCH [O(n)]

IN ORDER TO FETCH AN ITEM, WE CHECK EVERY ELEMENT IN THE ARRAY

2, 4, 1, 0, 7, 8, 6

BINARY SEARCH O(log n)

IN BINARY SEARCH WE NEED ORGANIZE OUR DATA TO ACHIEVE (log n)

HASHTABLE (ADT)

HAS AN AVERAGE SEARCH TIME O(1). HOWEVER DEPENDING ON THE IMPLEMENTATION IT MIGHT REQUIRE MORE MEMORY.

SUPPORTED OPERATIONS.

- INSERT (item)
- DELETE (item)
- SEARCH (KEY)

WHEN WE INSERT ELEMENTS IN A ARRAY WE CALCULATE THEIR INDEX INTO THAT ARRAY.

HASHING TECHNIQUES

$\boxed{9, 4, 14, 7, 5, 10}$ → KEY SPACE.

$\boxed{h(x) = x}$ → HASHING fuction

				4	5		7		9	10				14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

IN ORDER TO AVOID ALLOCATING LARGE MEMORY SPACE, WE COULD MODIFY THE HASHING FUNCTION.

$h(x) = x \% \text{ size}$.

$h(9) = 9 \% 10$

$h(14) = 14 \% 10$

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

14 → collision.

WE NEED TO RESOLVE COLLISIONS GRACEFULLY.

COLLISION RESOLUTION TECHNIQUES.

- OPEN ADDRESSING (OPEN HASHING) ⇒ CHAINING

- OPEN ADDRESSING (OPEN HASHING) ⇒ CHAINING
- LINEAR PROBING
- QUADRATIC PROBING
- DOUBLE HASHING.
- PERFECT HASHING
- UNIVERSAL HASHING

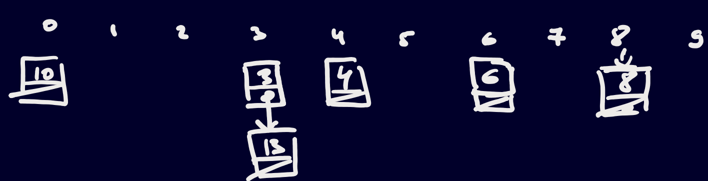
CHAINING

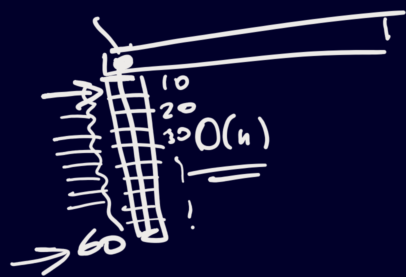
SIMPLIST HASHING TECHNIQUE, WHERE WE ALLOCATE A LINKED LIST AT THE DESIRED ARRAY POSITION

KEY SPACE

8, 3, 13, 6, 4, 10 $h(x) = x \% \text{SIZE}$

$h(60) = 60 \% 10 = 0$





20, 30, 40, 60, 10.

LINEAR PROBING

9, 4, 14, 7, 5, 10 $h(x) = x \% \text{SIZE}$

10				4	14	5	7		9
0	1	2	3	4	5	6	7	8	9

$f(i) = i$

$h'(x) = [\,h(x) + f(i)\,] \% \text{SIZE}$ WHERE $f(i) = 0, 1, 2, 3$ ----

$h'(4) = [\,h(4) + f(0)\,] \% 10 = 4$

$h'(14) = [\,h(14) + f(0)\,] \% 10 = 4$

$h'(14) = [\,h(14) + f(1)\,] \% 10 = 5$

QUADRATIC PROBING

SIMILAR TO LINEAR PROBING BUT HAS A DIFFERENT HASHING FUNCTION TO AVOID CLUSTERING.

HASHING FUNCTION TO AVOID CLUSTRING.

$$h'(x) = [\, h(x) + f(i) \,] \,\%\, size$$

$$f(i) = i^2$$

DOUBLE HASHING

SIMILAR TO QUADRATIC AND LINEAR PROBING, EXCEPT IT RELIES ON TWO HASHING FUNCTIONS INSTEAD OF ONE.

$h_1(x) = x \,\%\, size$ THE SECOND ONE WILL BE

$h_2(x) = m - (x \,\%\, m)$ WHERE M IS THE CLOSEST PRIME NUMBER TO THE SIZE OF THE ARRAY. (ART OF COMPUTER PROGRAMMING)

final hashing function

$$h_1(x) + j\, h_2(x)$$

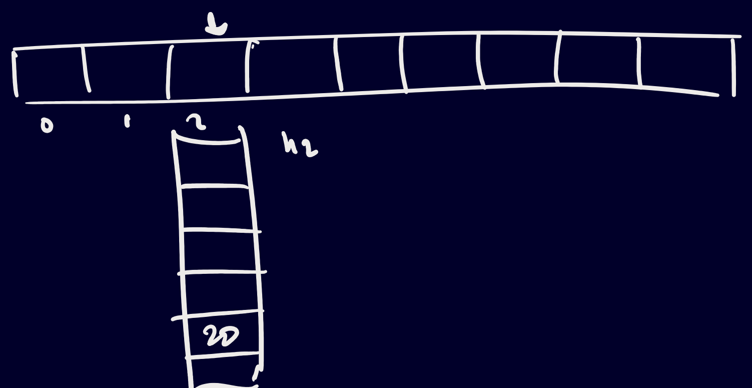
$j \Rightarrow 1, 2, 3, 4, 5$

PERFECT HASHING

IT HAS SIMILAR CONCEPT TO CHAINING, EXCEPT, INSTEAD OF USING LINKED LISTS IT USE HASHTABLE

$h_1(x) \Rightarrow$ TO FIND POSITION IN THE ARRAY

$h_2(x) \Rightarrow$ FIND POSITION IN THE SUB ARRAY.



UNIVERSAL HASHING

IT RELIES ON HAVING A COLLECTION OF HASHING FUNCTIONS ... THROUGHOUT THE

IT RELIES ON HAVING A COLLECTION OF HASHING FUNCTIONS THAT WE PICK AT RANDOM AND WE STICK THROUGHOUT THE RUNTIME OF THE PROGRAM.

1 → CHOOSE A VERY LARGE PRIME NUMBER $\boxed{P}$

2 → $b \in \{0, 1, 2, \ldots, P-1\}$

3 → $a \in \{1, 2, 3, \ldots, P-1\}$

4 → m = SIZE OF THE ARRAY

$$h_{a,b}(X) = [(ax + b) \% P] \% m.$$

DICTIONARY <T, K>