Data Structures

An Introduction

Data Structures – Definition

- Data Structure:
  - It is a structure that can store a limited number of data; provides a means to access any of the data, delete any of the data and insert additional data.
  - Supported Operations:
    - Insertion/Addition of elements
    - Deletion/Removal of elements
    - Accessing of elements
    - Modifying of elements – *optional*.
  - Has specified behavior for each supported operation.

Data Structures

Strong Neighbors

- We have a contiguous strip of houses identified by owner's name
- House numbers starts from zero
- There must be no vacant house
- Walls must be shared

0	1	2	3	4	5	6	7	8	9
SAM	JOE	ADA	SANI	FAITH	TIM	HILDA	AISHA	RAUF	KEN

Strong Neighbors

- Estate owner.

SIZE( )

GET(index)

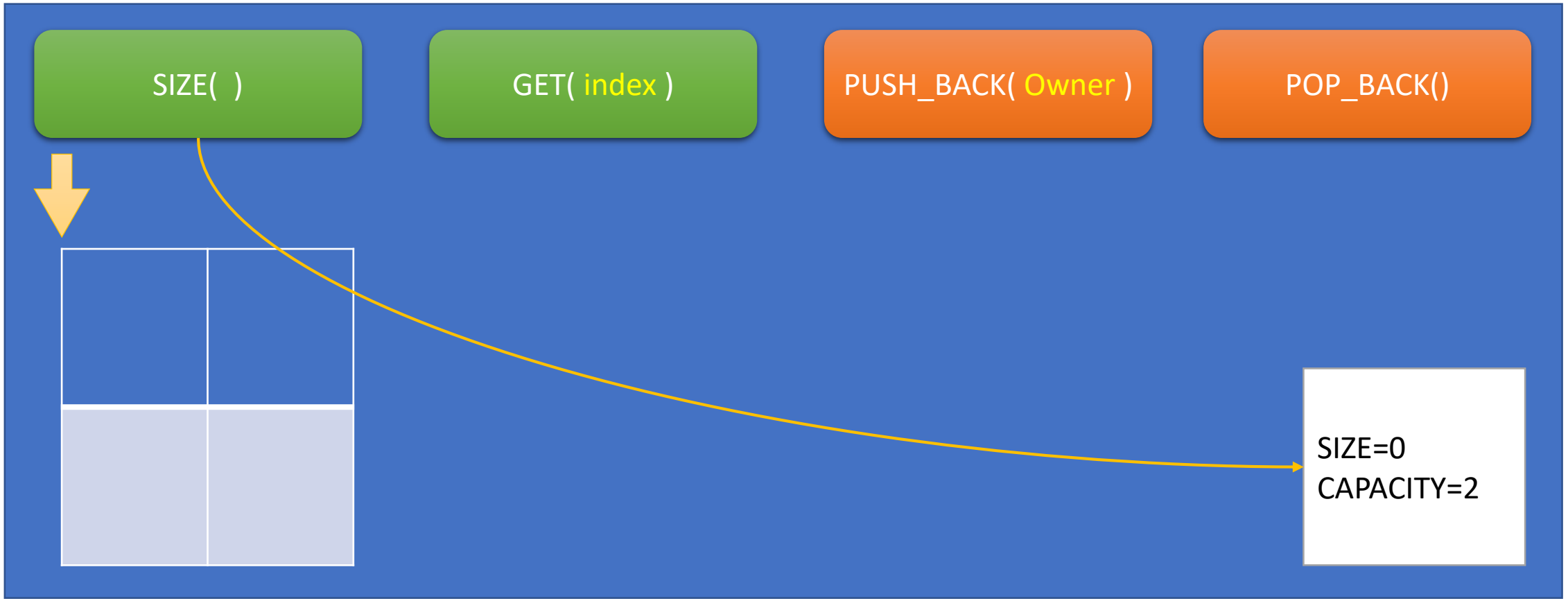
PUSH_BACK(Owner)

POP_BACK()

0	1	2	3	4	5	6	7	8	9
SAM	JOE	ADA	SANI	FAITH	TIM	HILDA	AISHA	RAUF	KEN

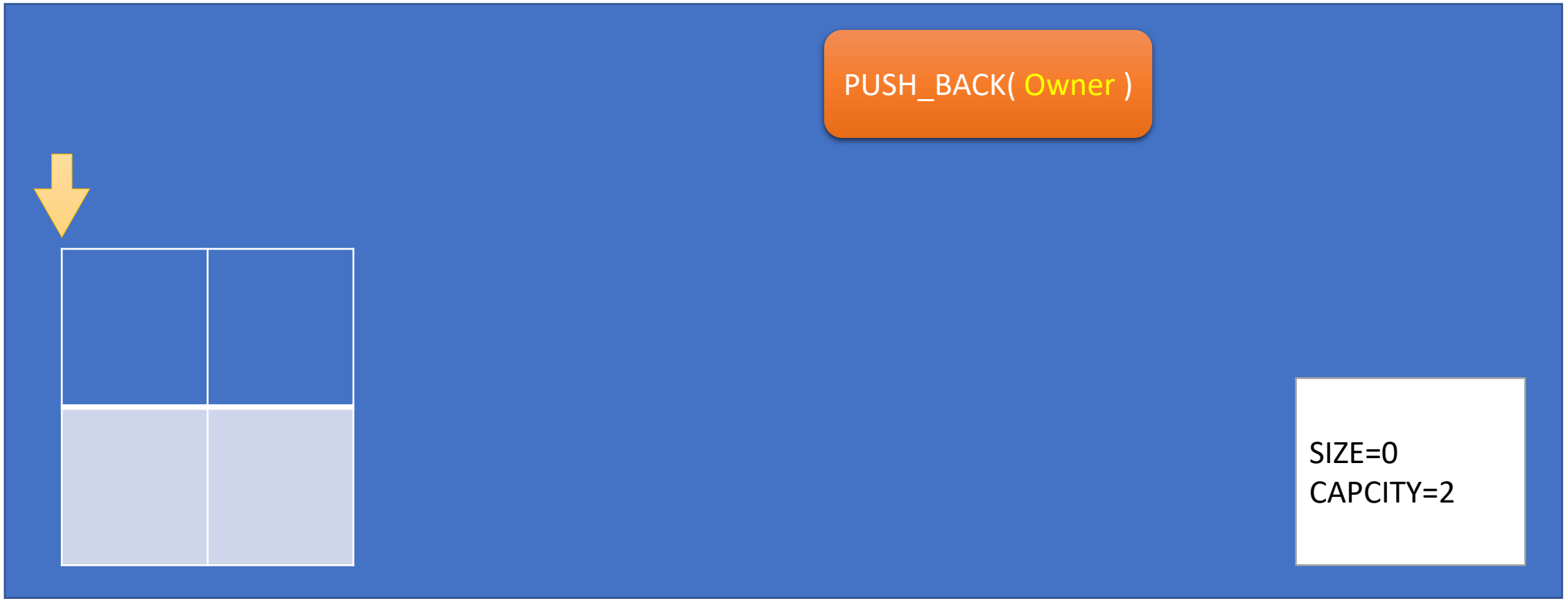
Strong Neighbors

- An Empty Estate…



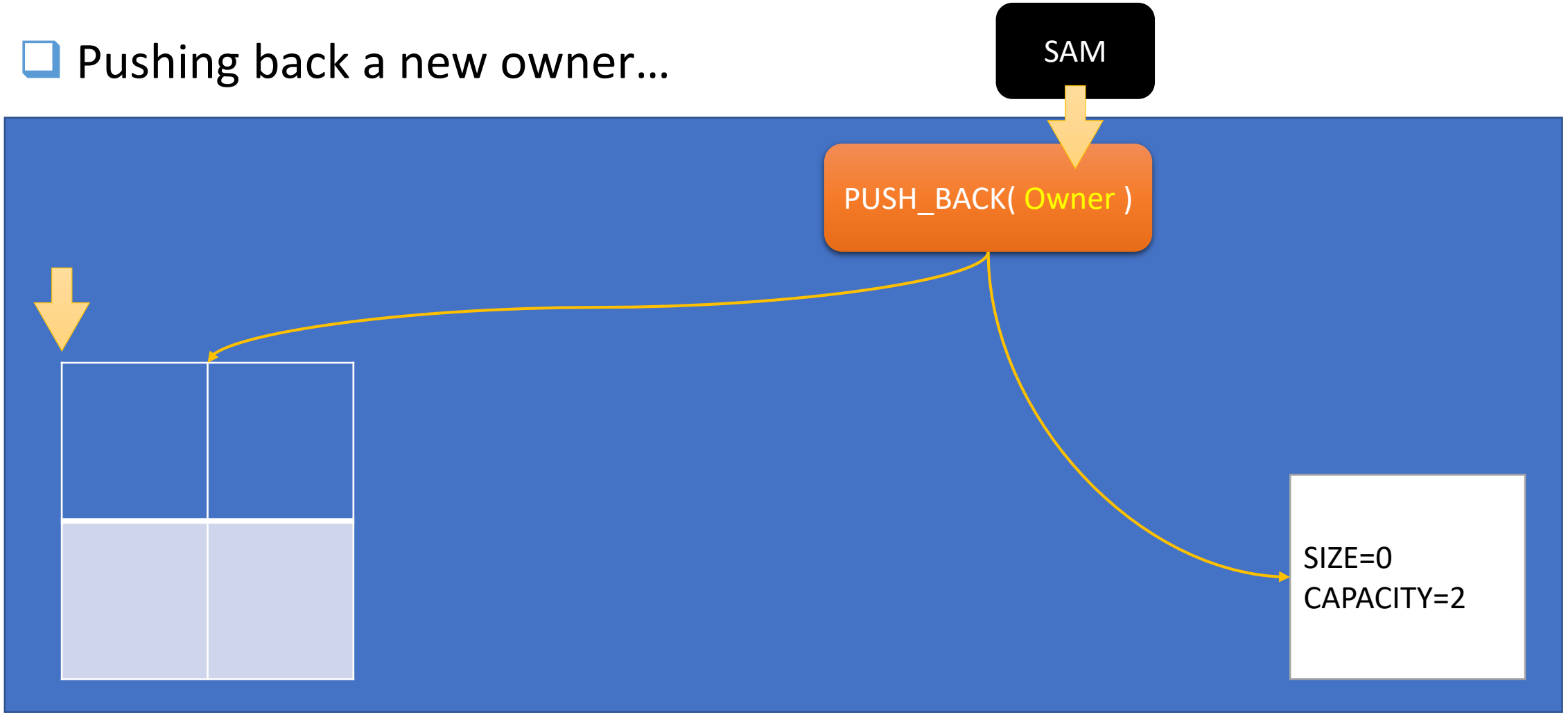
Strong Neighbors

- Pushing back a new owner…



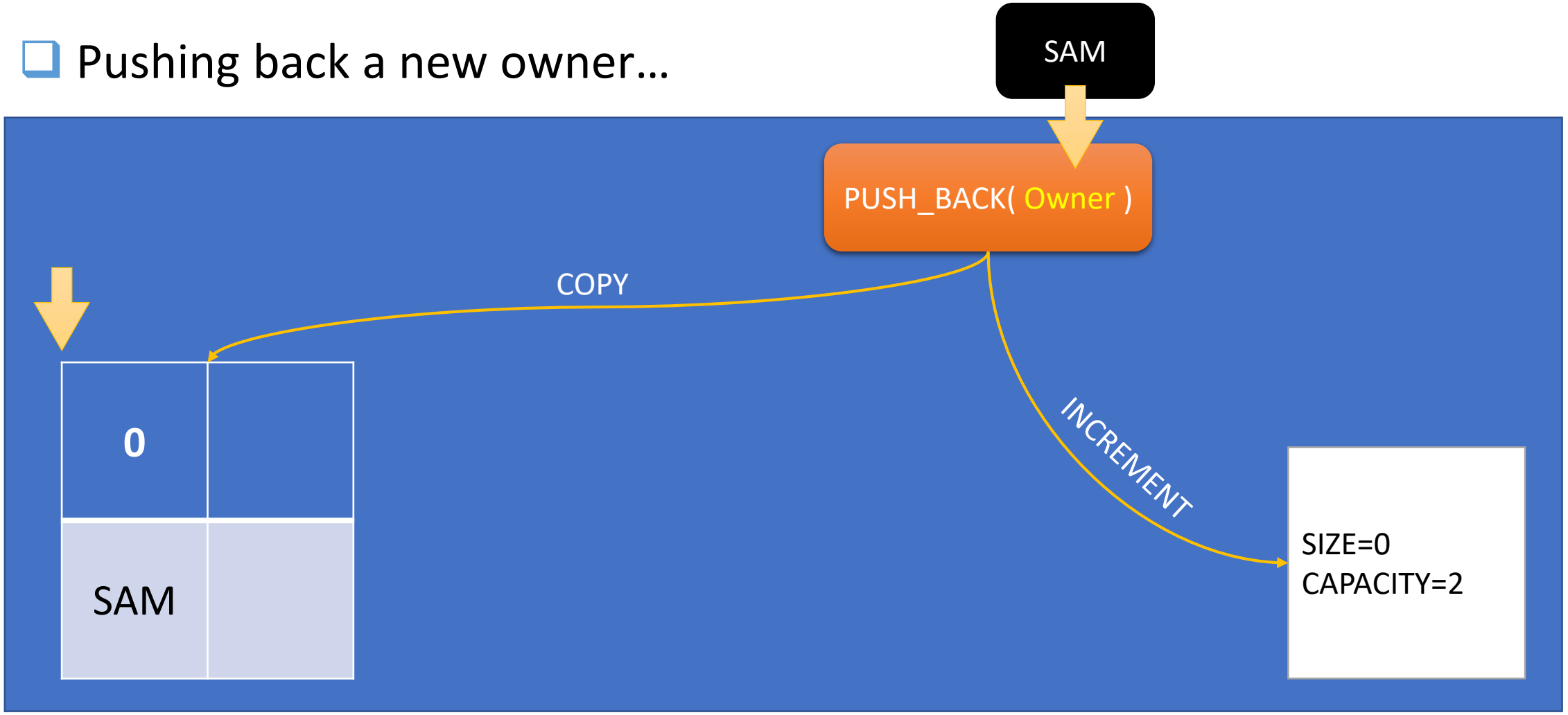
Strong Neighbors

- Pushing back a new owner…



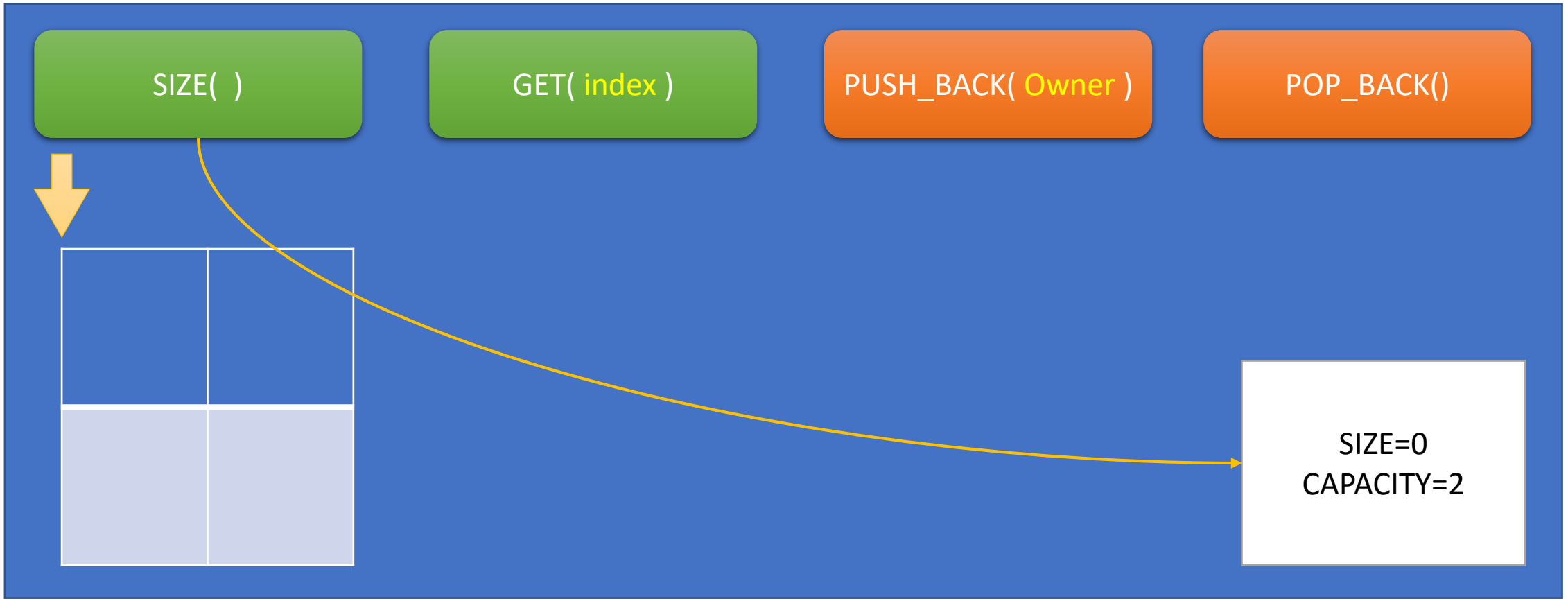
Strong Neighbors

- Pushing back a new owner…



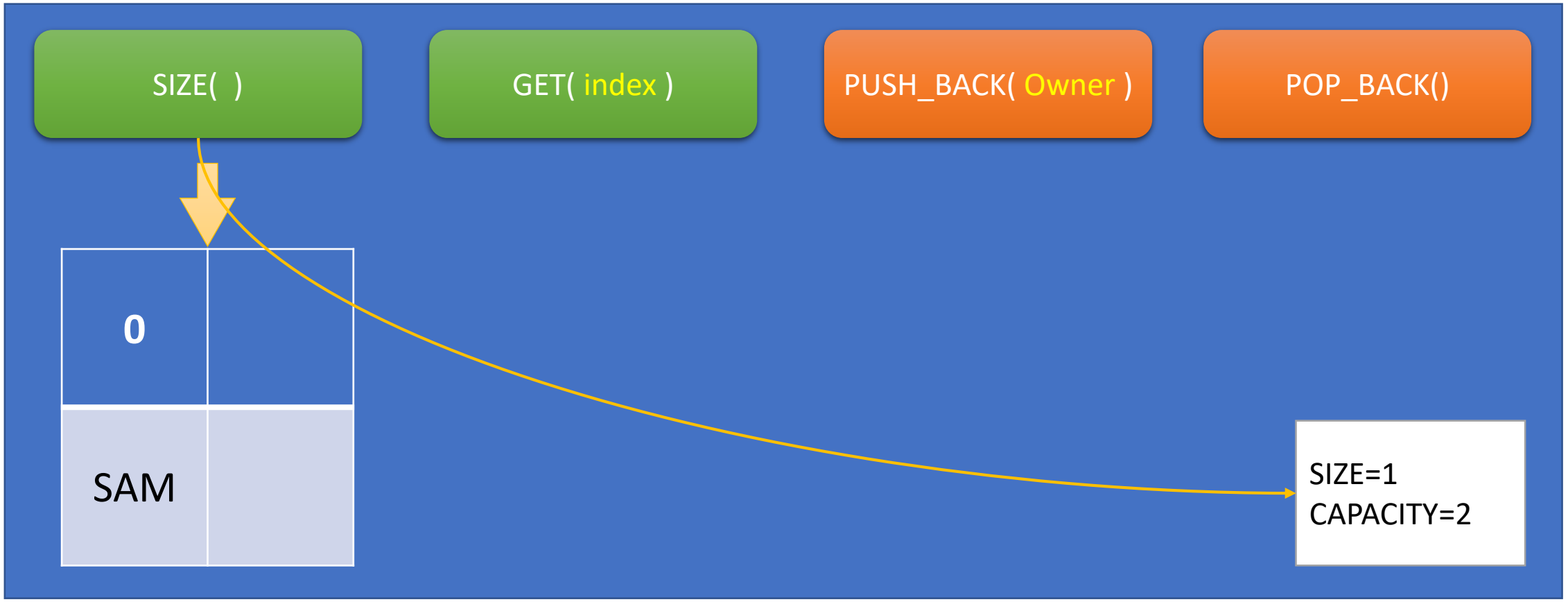
Strong Neighbors

- Pushing back a new owner…



Strong Neighbors

- An Estate of One Person…

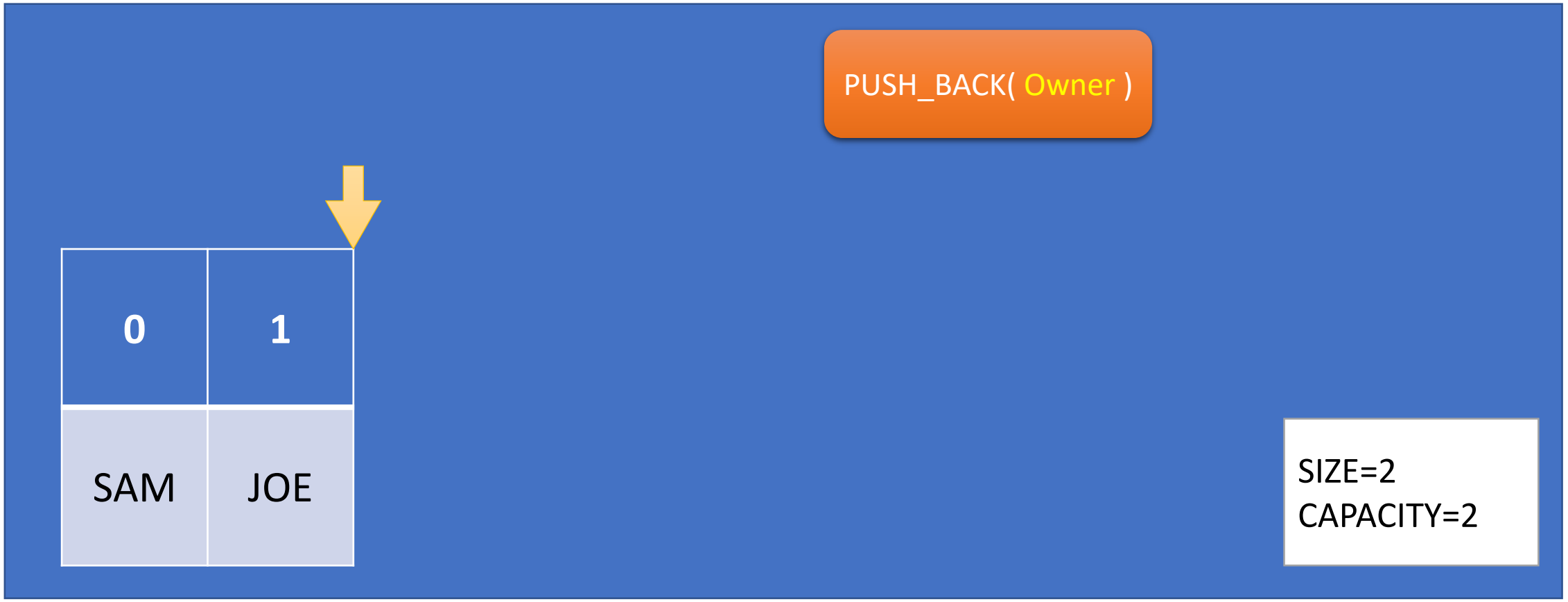


Strong Neighbors

What happens when Estate is full?

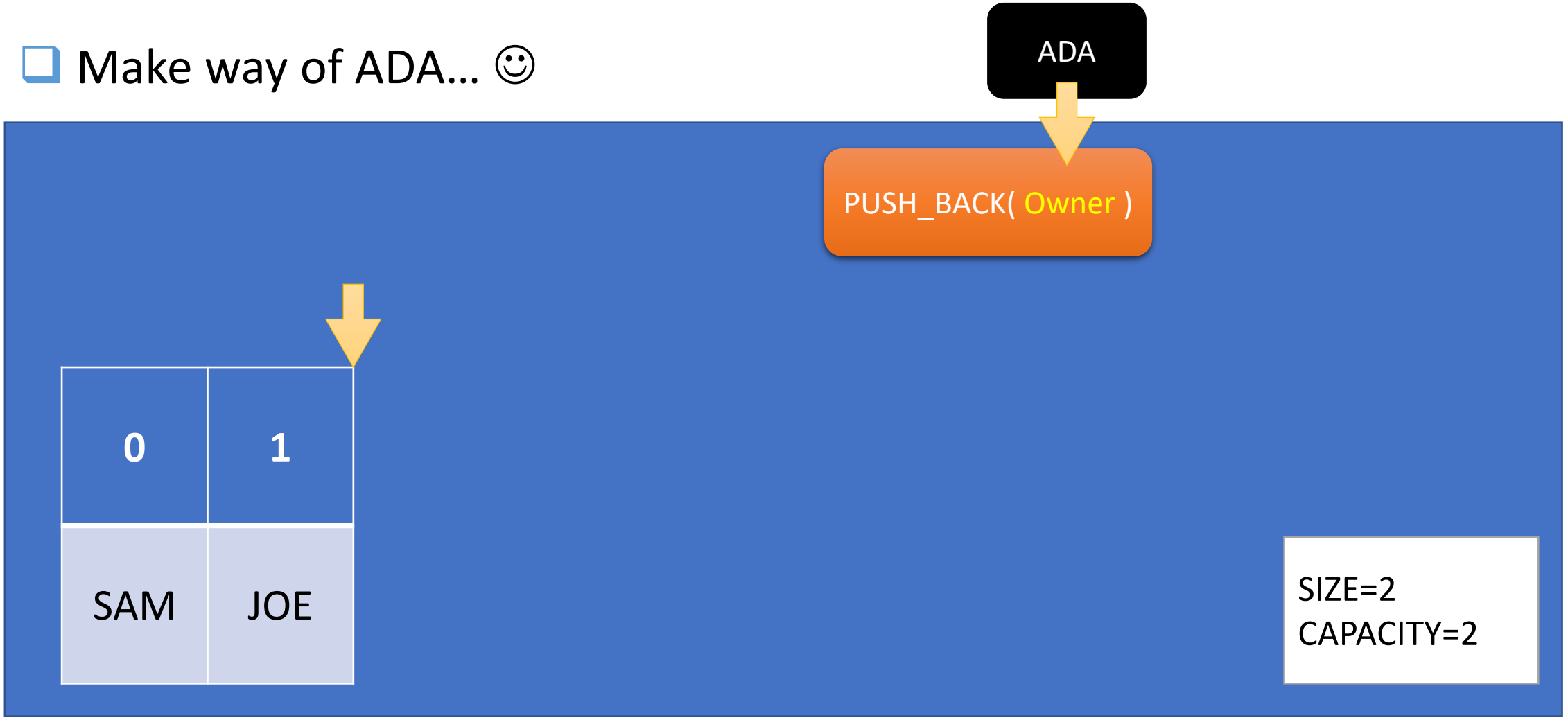
Strong Neighbors

- When the Estate is full and we need new house…



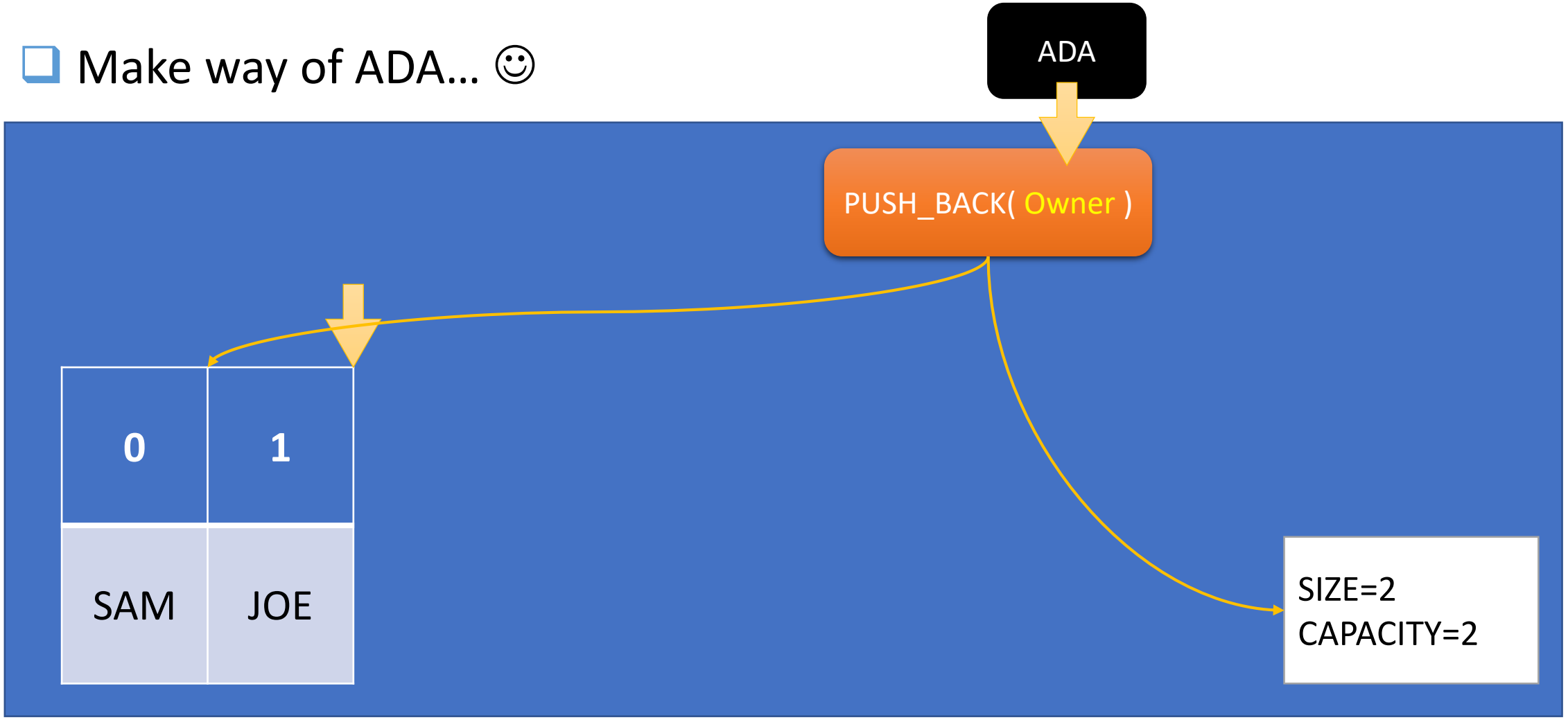
Strong Neighbors

- Make way of ADA… ☺



Strong Neighbors

- Make way of ADA… ☺



Strong Neighbors

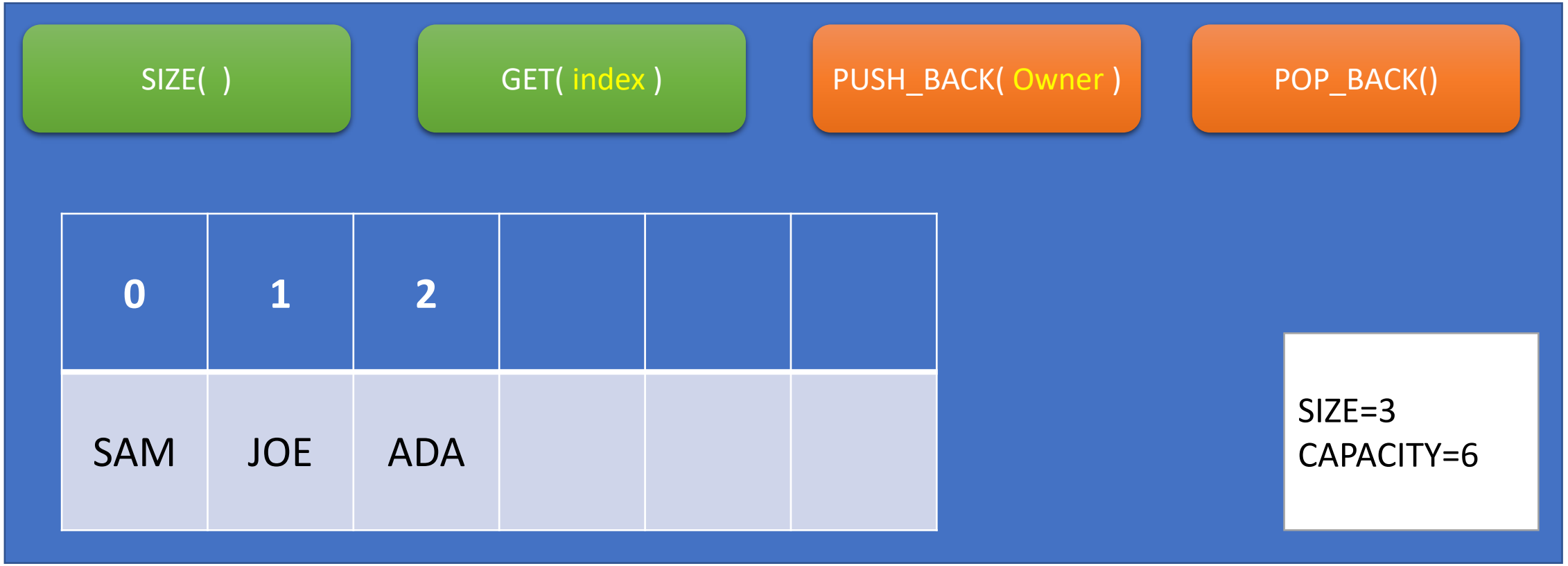
Make way for the fine lady!!

Strong Neighbors

- Buy a new plot of land double the size we have
- Copy or Move Construct Existing members
- Demolish old plot of land
- Construct new houses! ☺

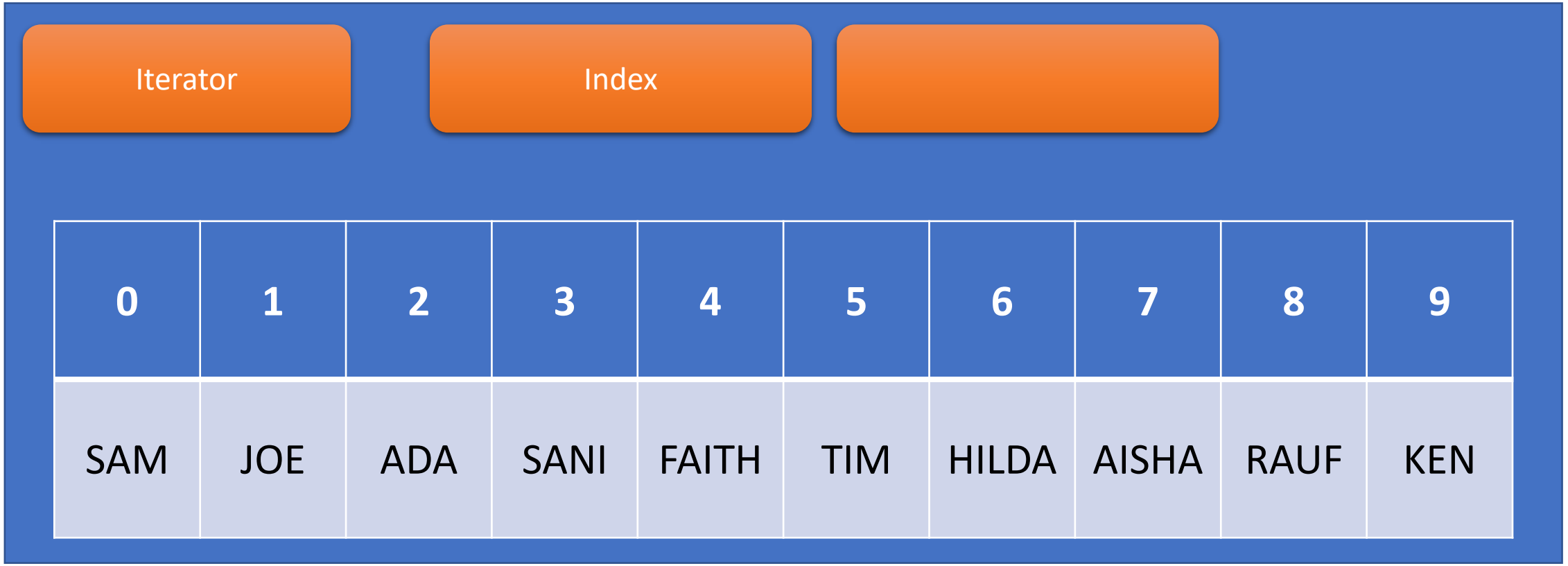
Strong Neighbors

- More space…..



Strong Neighbors – Array!

- This my friend, is an Array! ☺



0	1	2	3	4	5	6	7	8	9
SAM	JOE	ADA	SANI	FAITH	TIM	HILDA	AISHA	RAUF	KEN

Strong Neighbors – Array!

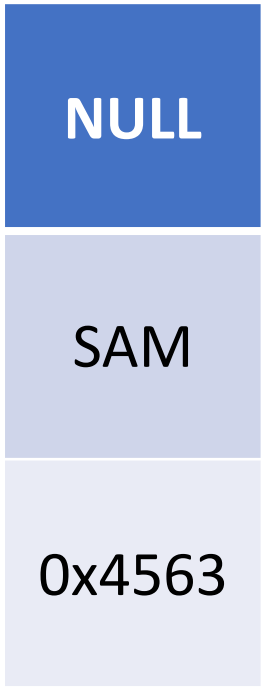
Questions?

Only Best of Friends

No mutuality!

Only Best of Friends!

- We have houses identified by owner's name
- Houses have no numbering
- Owner can refer you to friend far way…
- Houses have a global unique address

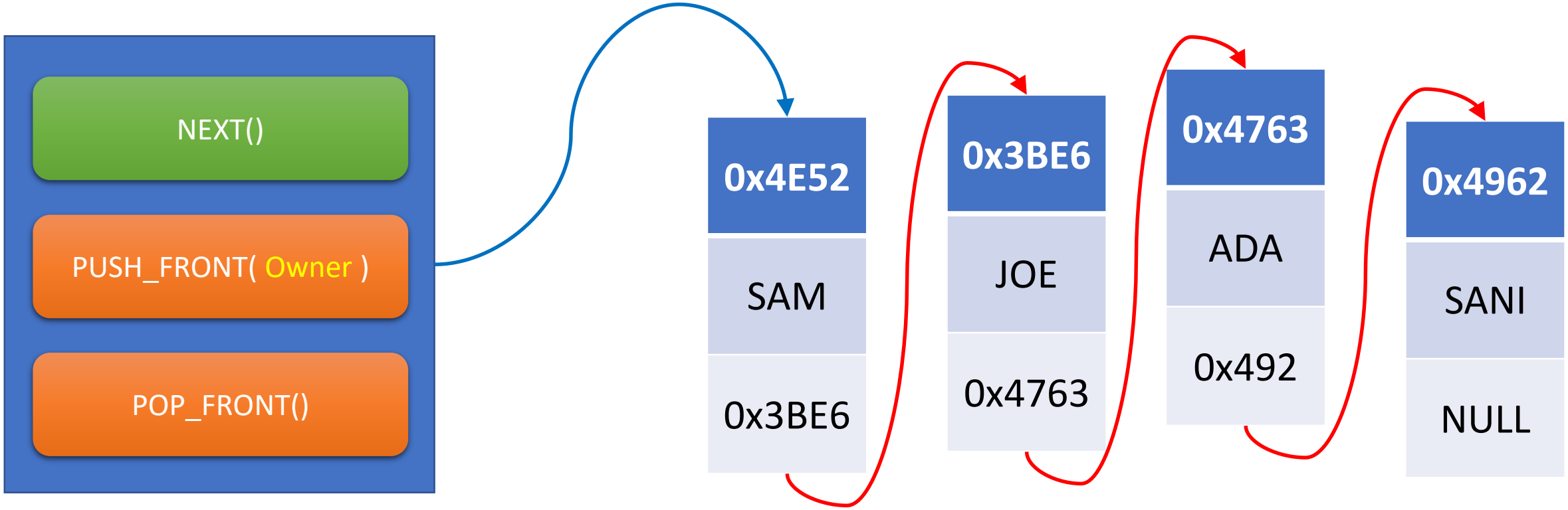


Only Best of Friends!

- We have houses identified by owner's name

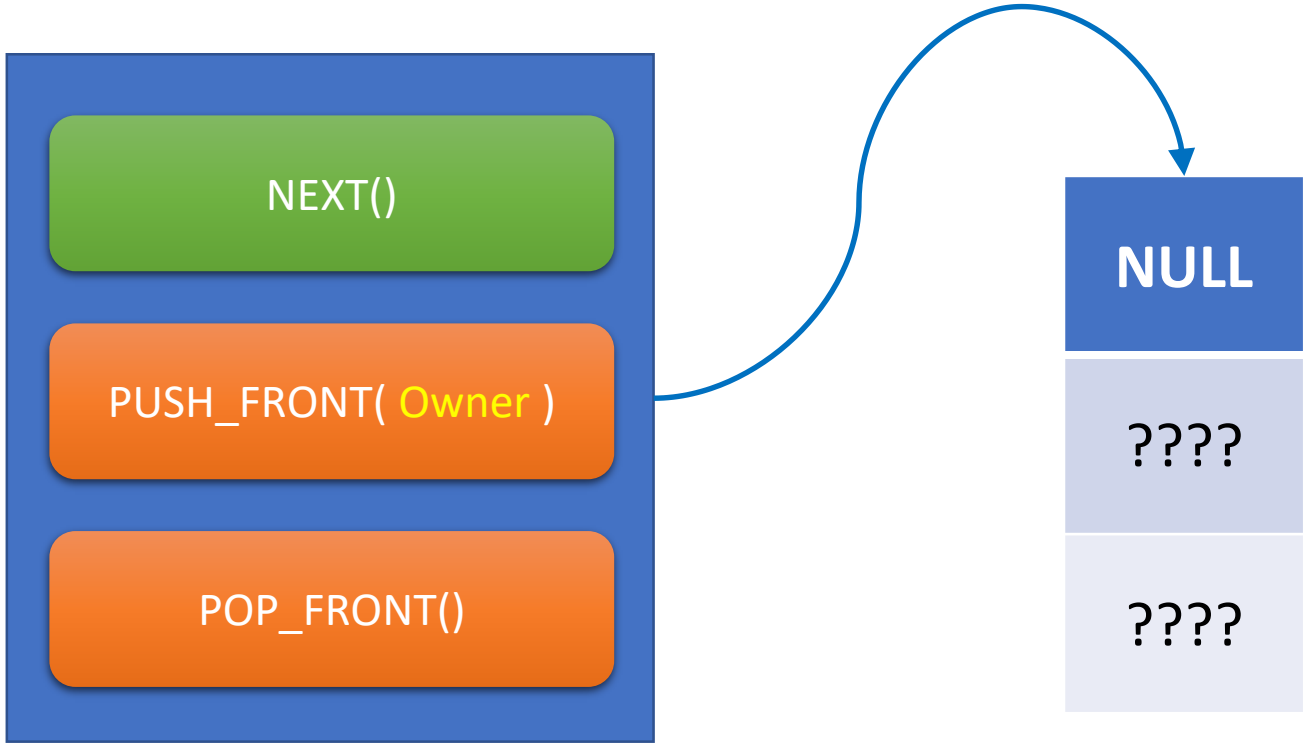
Only Best of Friends!

- The Manager, leaving them out in the wild…



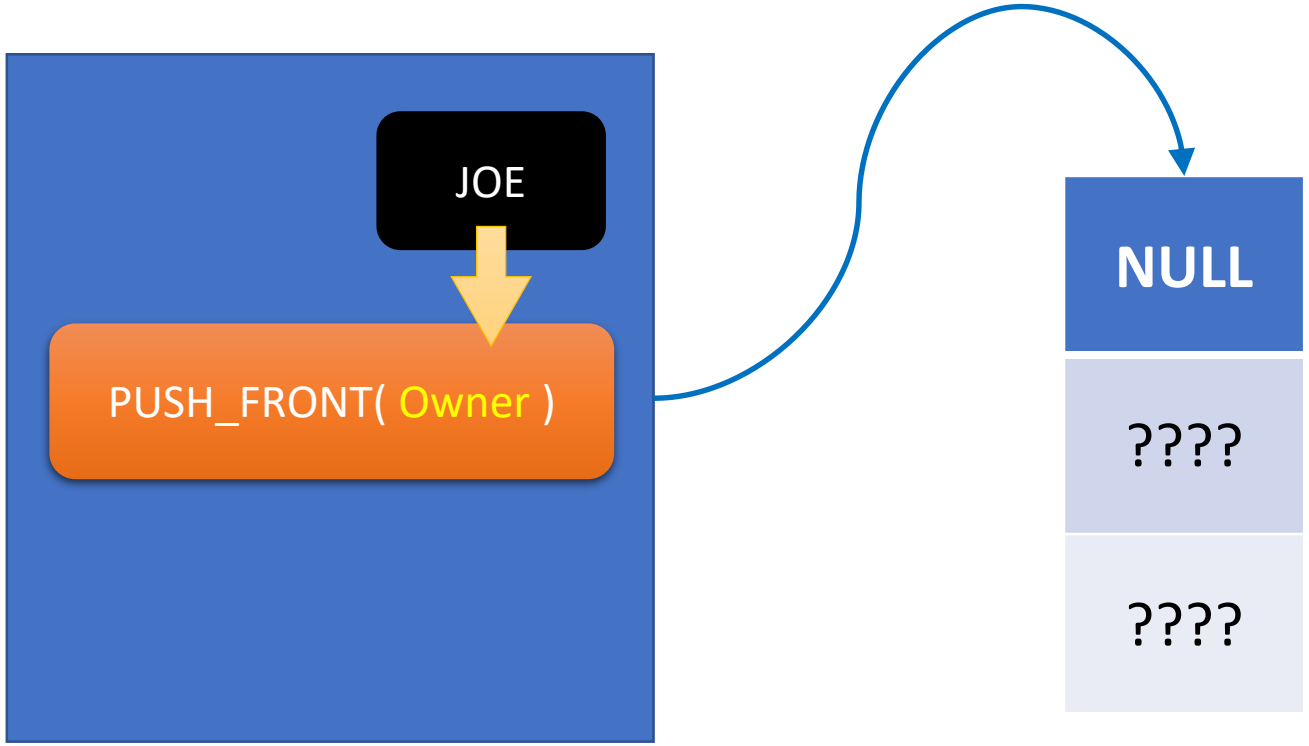
Only Best of Friends!

- An Empty Estate…



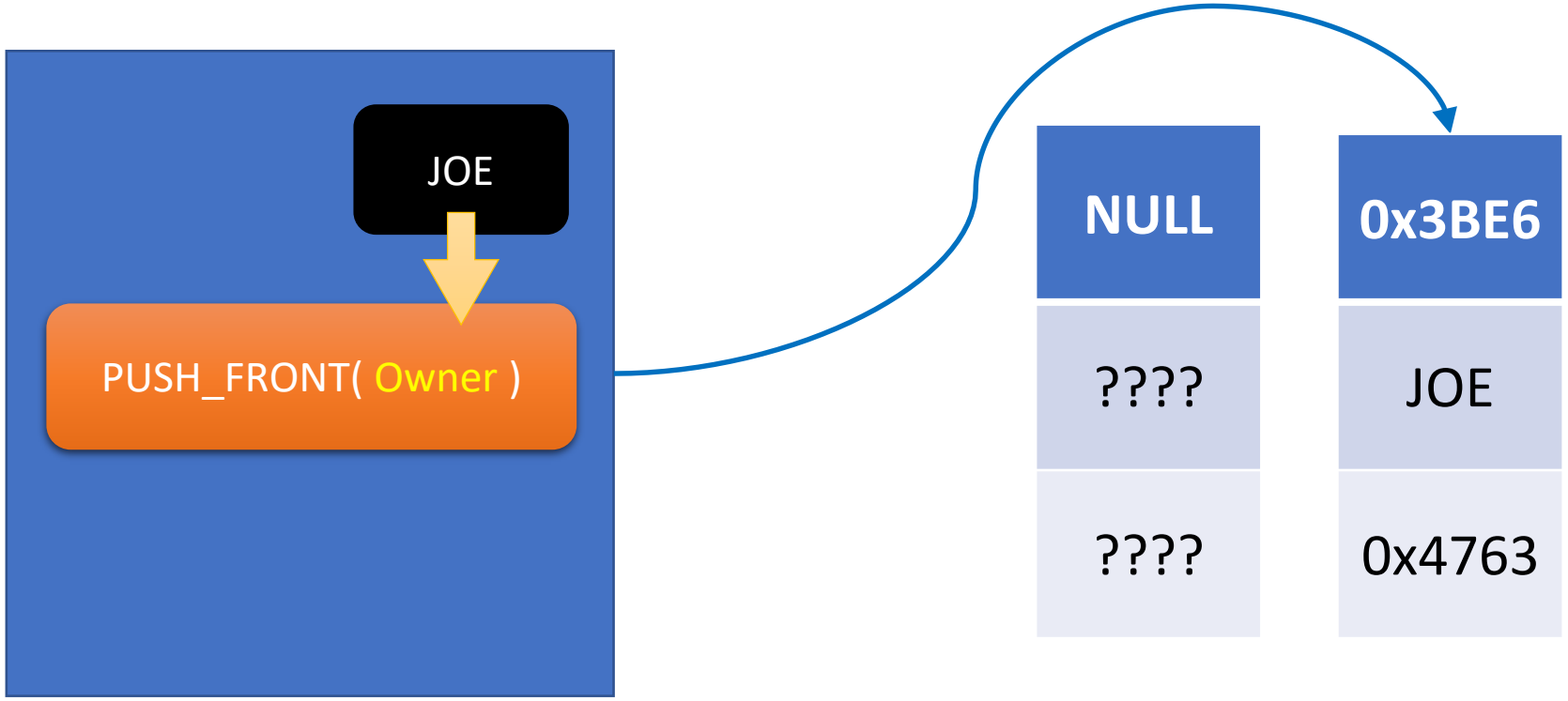
Only Best of Friends!

- Adding Joe…



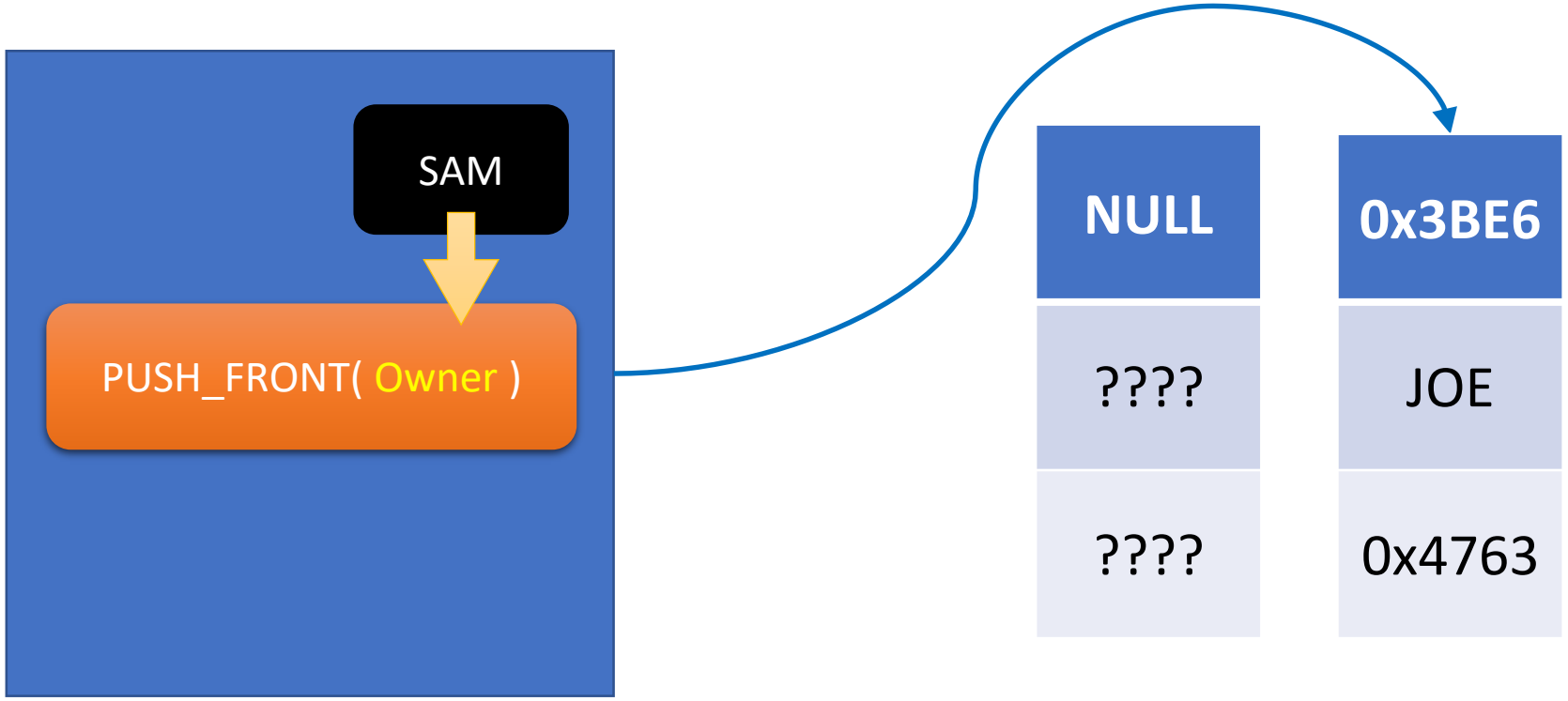
Only Best of Friends!

- Adding Joe…



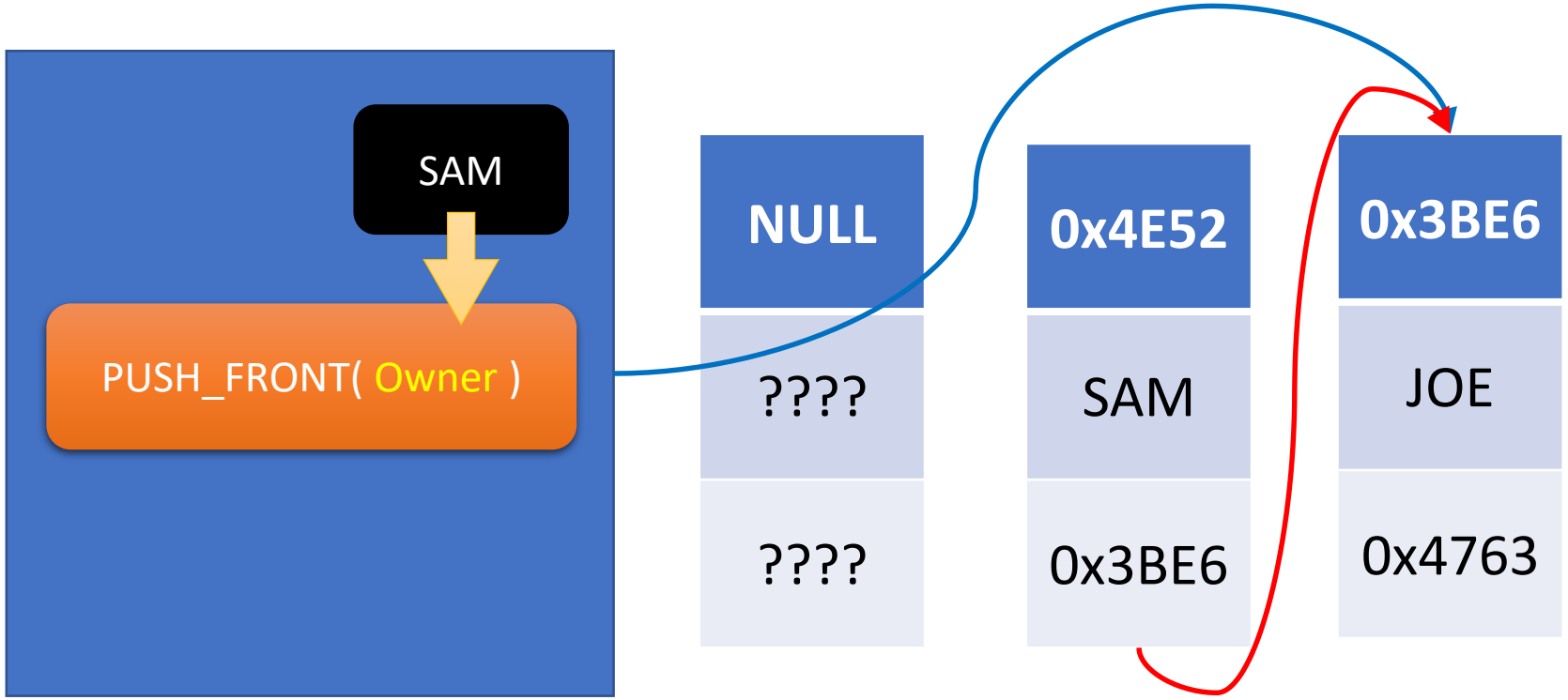
Only Best of Friends!

- Adding SAM…



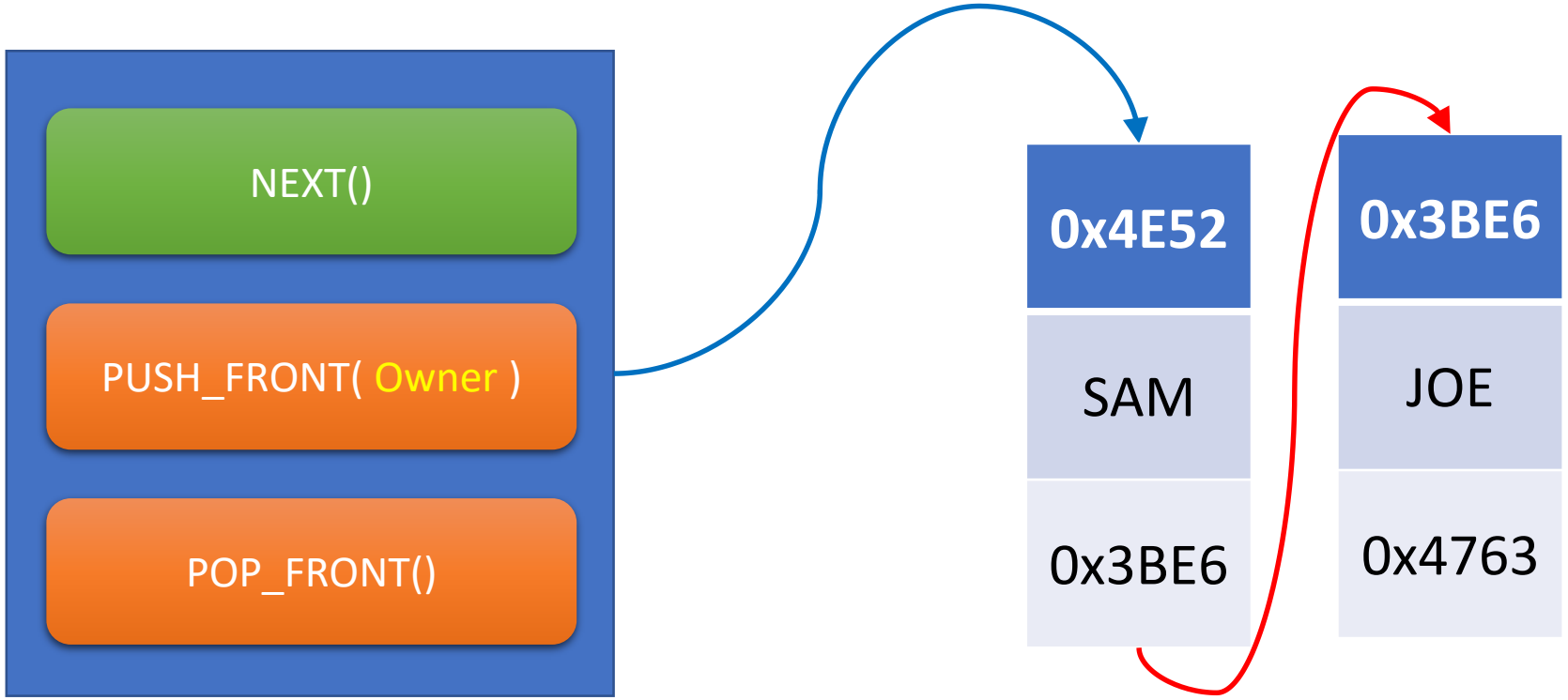
Only Best of Friends!

- Adding SAM…



Only Best of Friends!

- This, is a Singly Linked List! ☺



Best of Friends – List!

Questions?

Not Just Best of Friends

One Level of mutuality!

Not Just Best of Friends!

- We have houses identified by owner's name
- Houses have no numbering
- Owner can refer you to a friend who refers to him far way…
- Houses have a global unique address

CURRENT NODE ADDRESS
POINTER TO PREVIOUS NODE
DATA/PAYLOAD
POINTER TO NEXT NODE

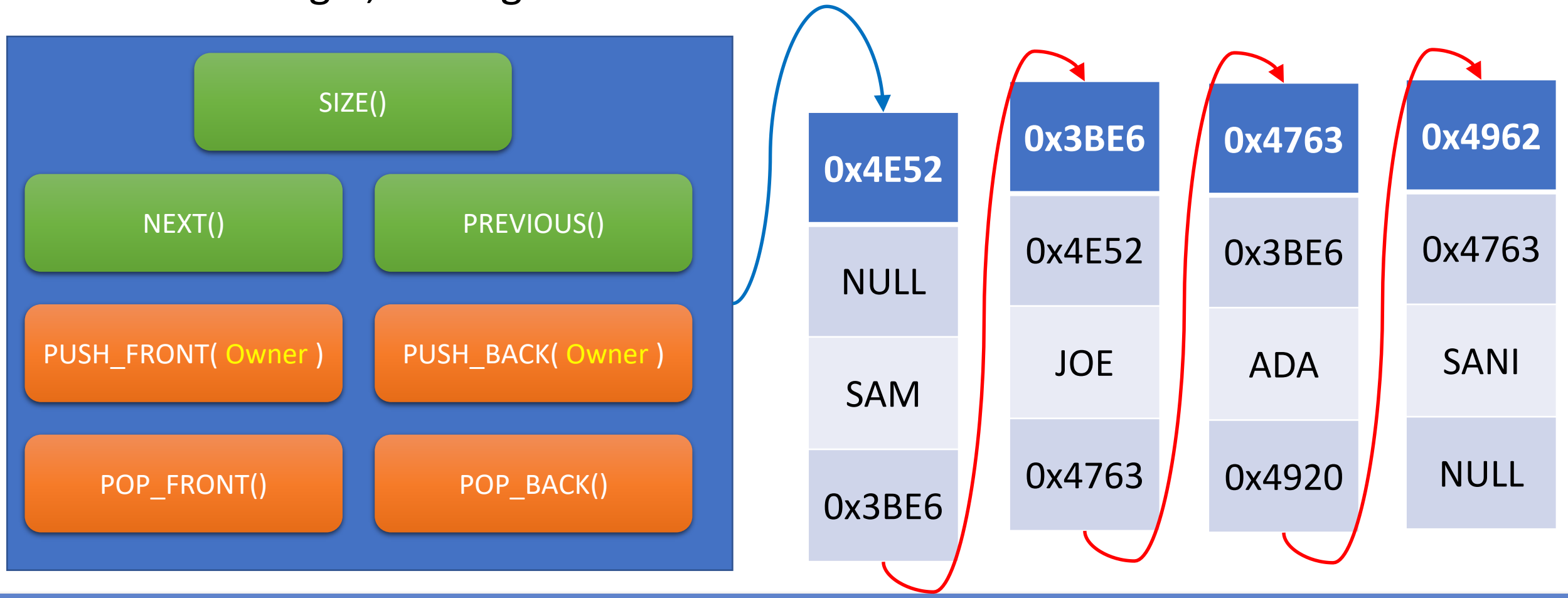
NULL
0x4563
SAM
0x45F3

Not Just Best of Friends!

- We have houses identified by owner's name

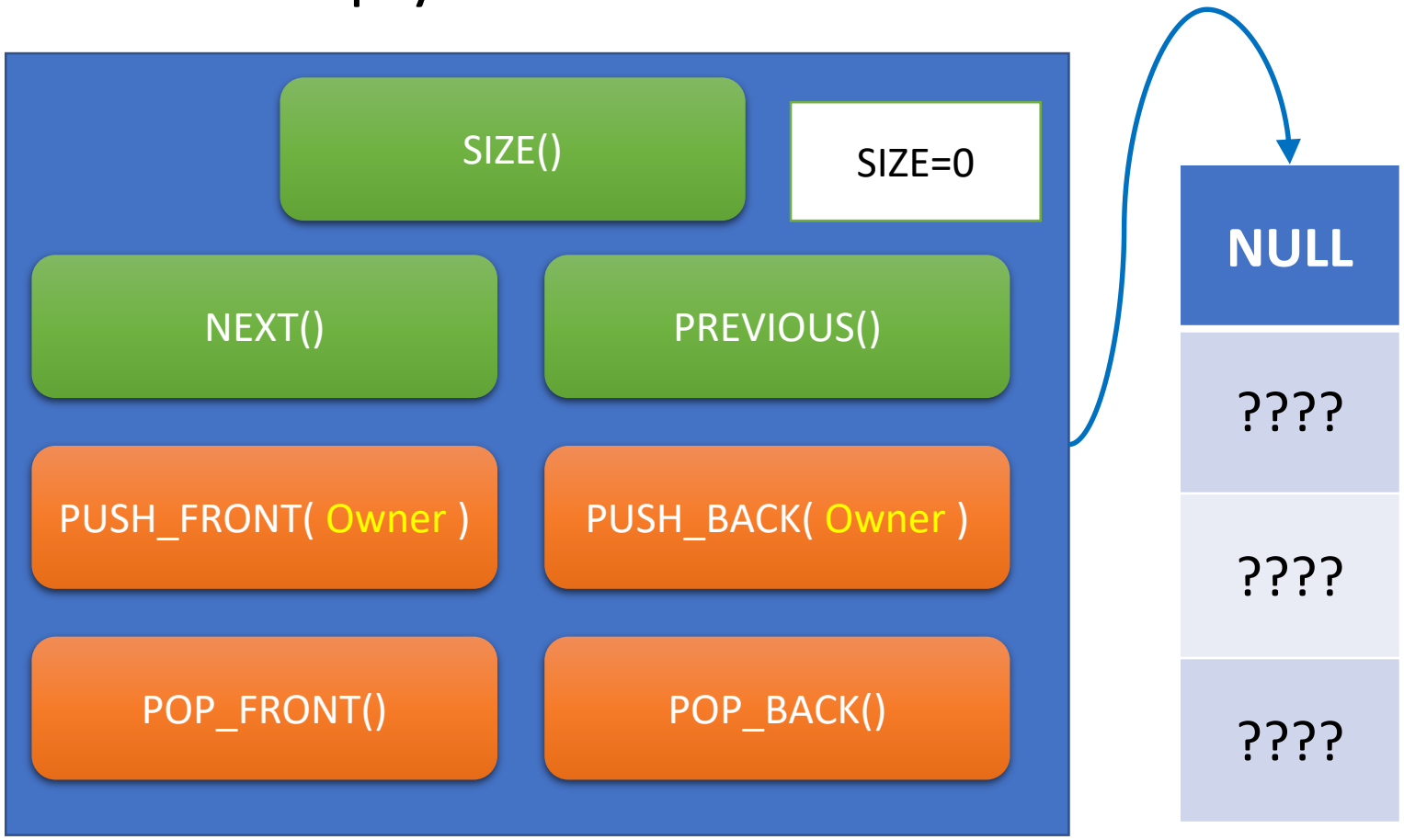
Not Just Best of Friends!

- The Manager, leaving them out in the wild…



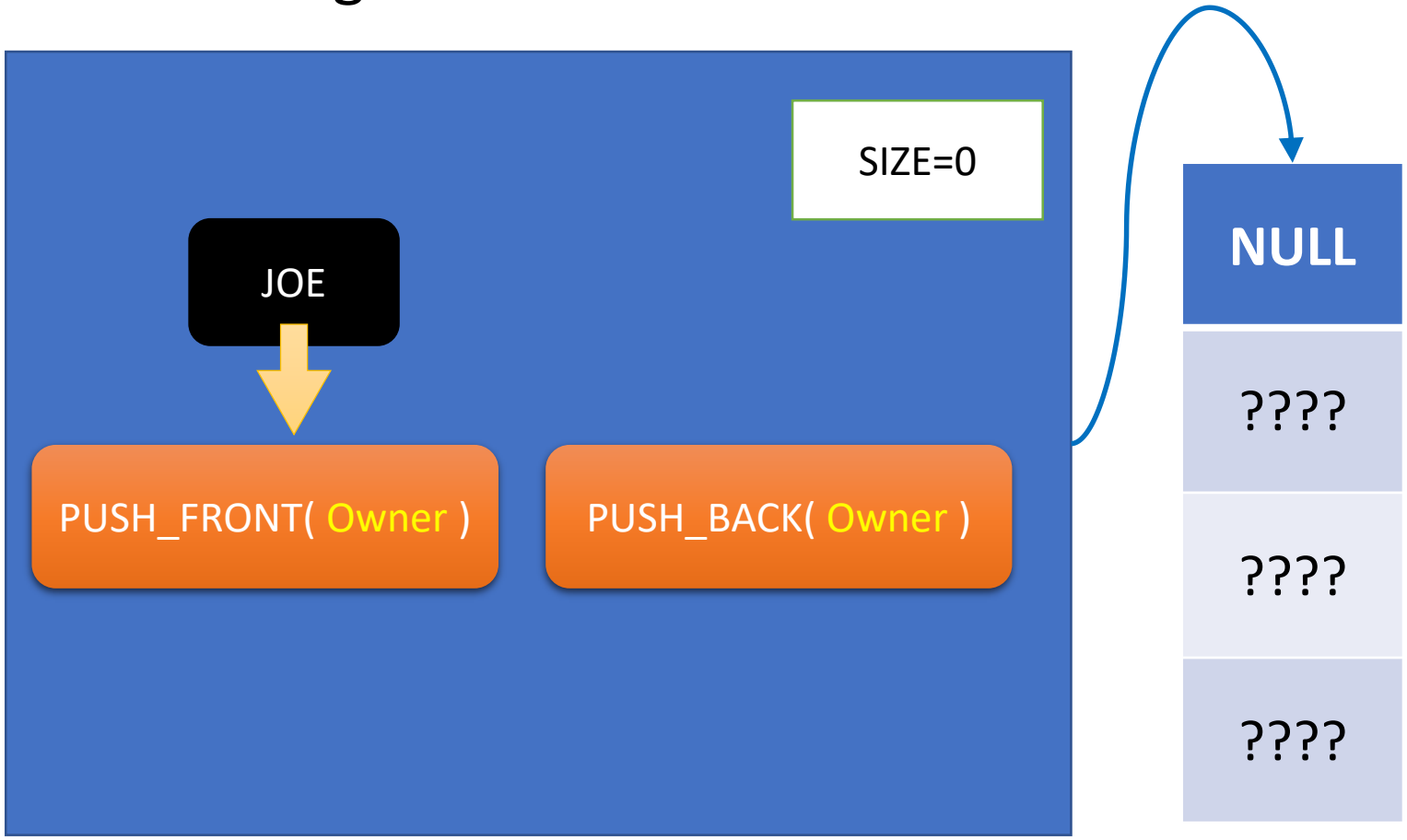
Not Just Best of Friends!

- An Empty Estate



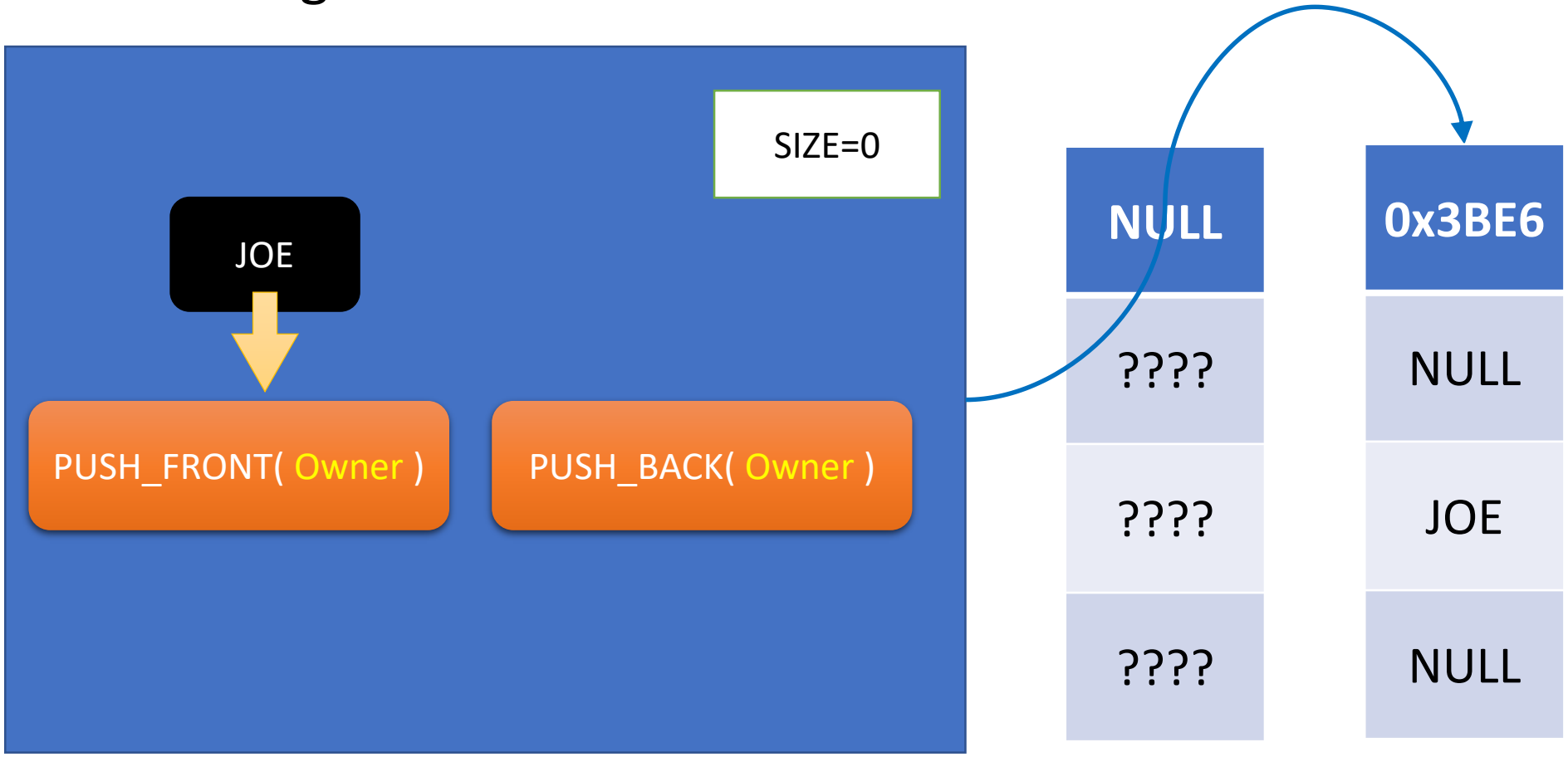
Not Just Best of Friends!

- Adding Joe



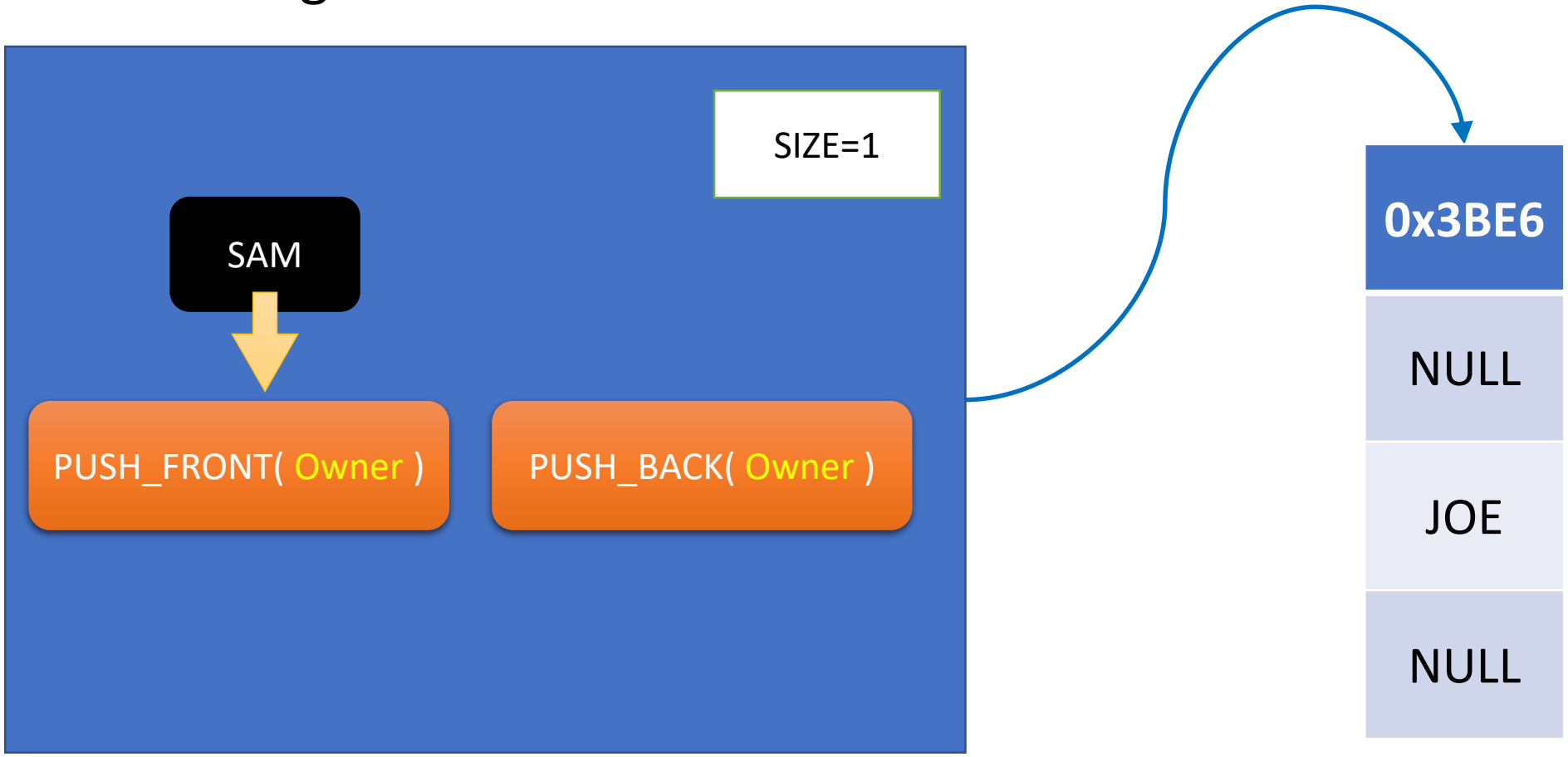
Not Just Best of Friends!

- Adding Joe



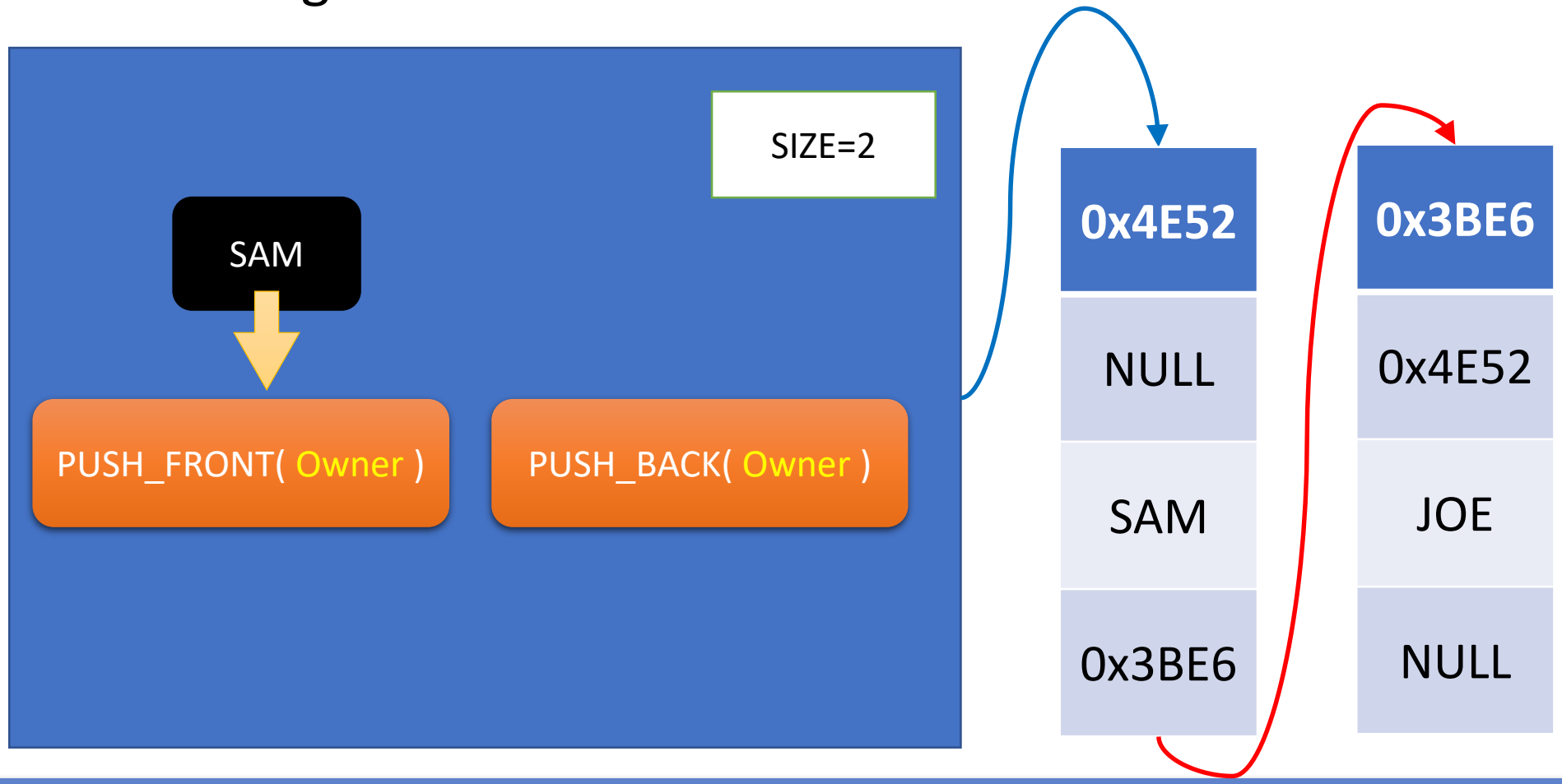
Not Just Best of Friends!

- Adding Sam



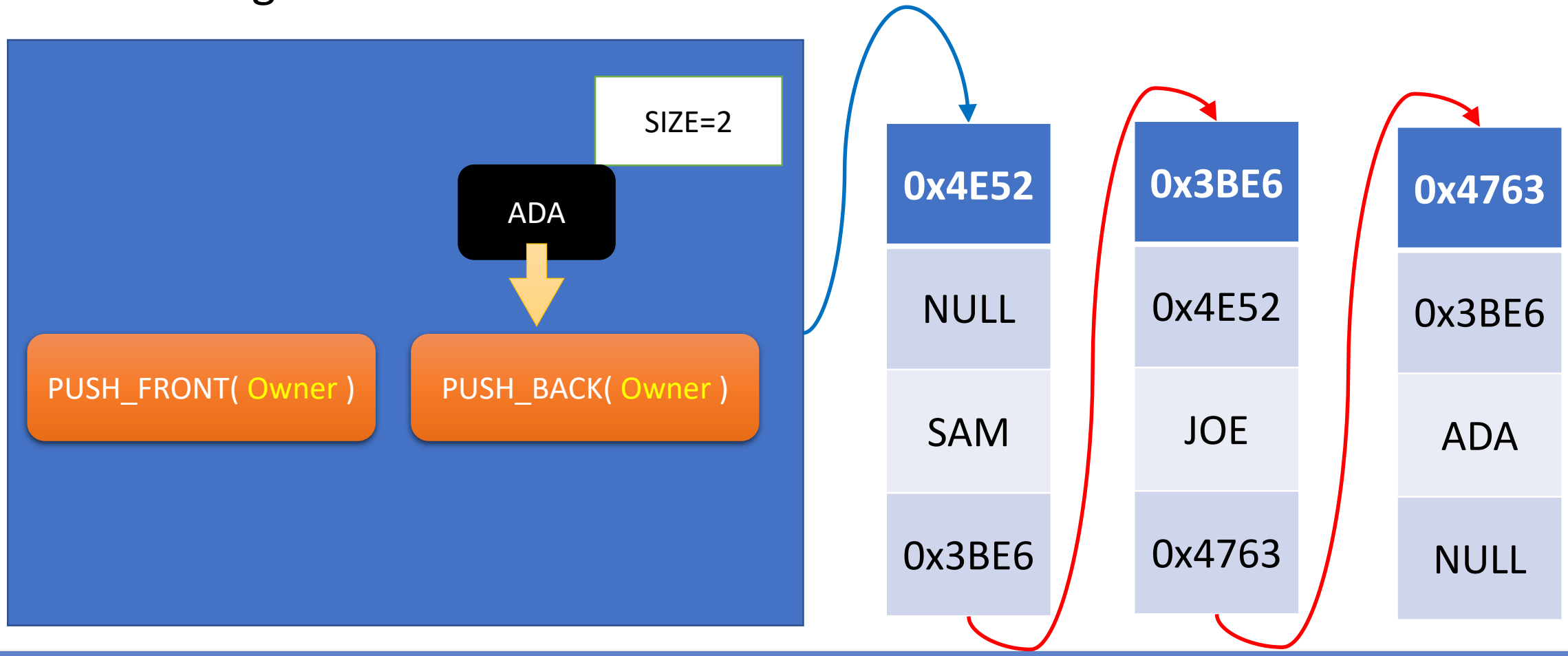
Not Just Best of Friends!

- Adding Sam



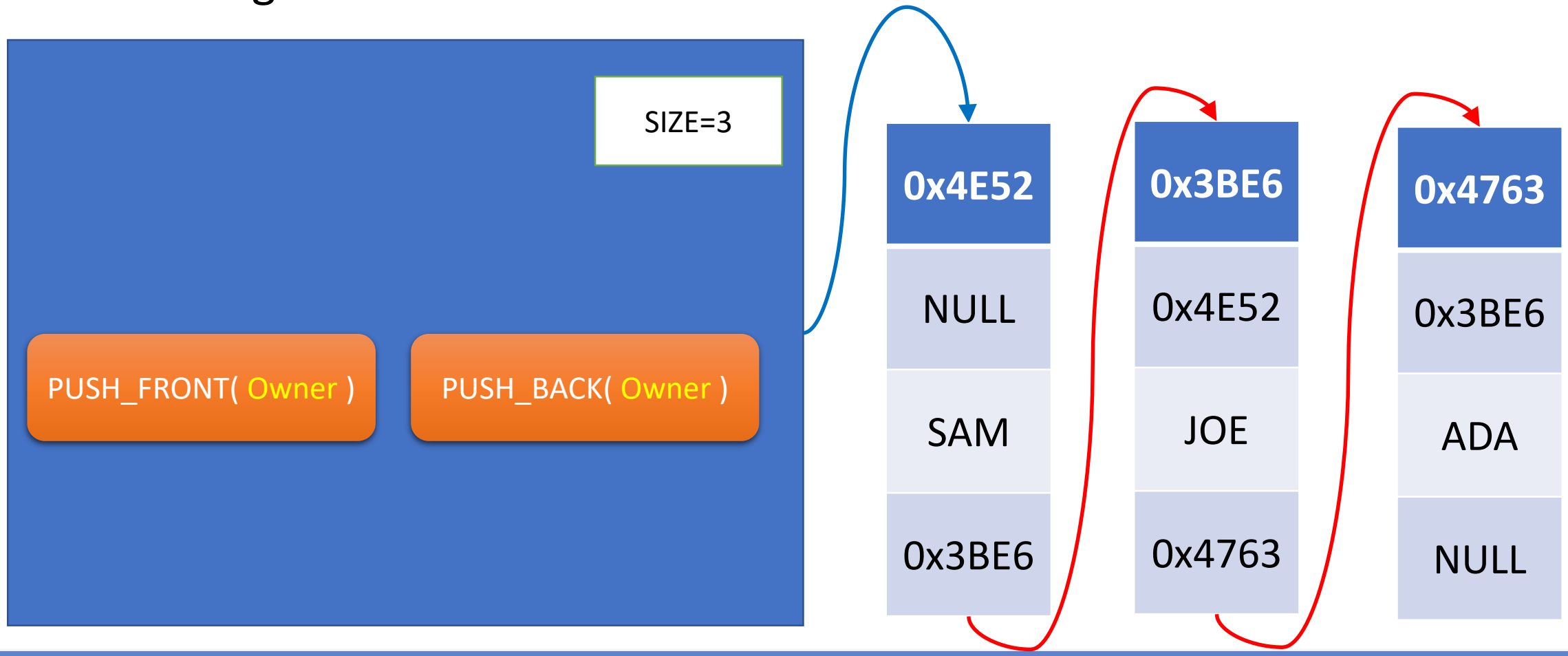
Not Just Best of Friends!

- Adding ADA



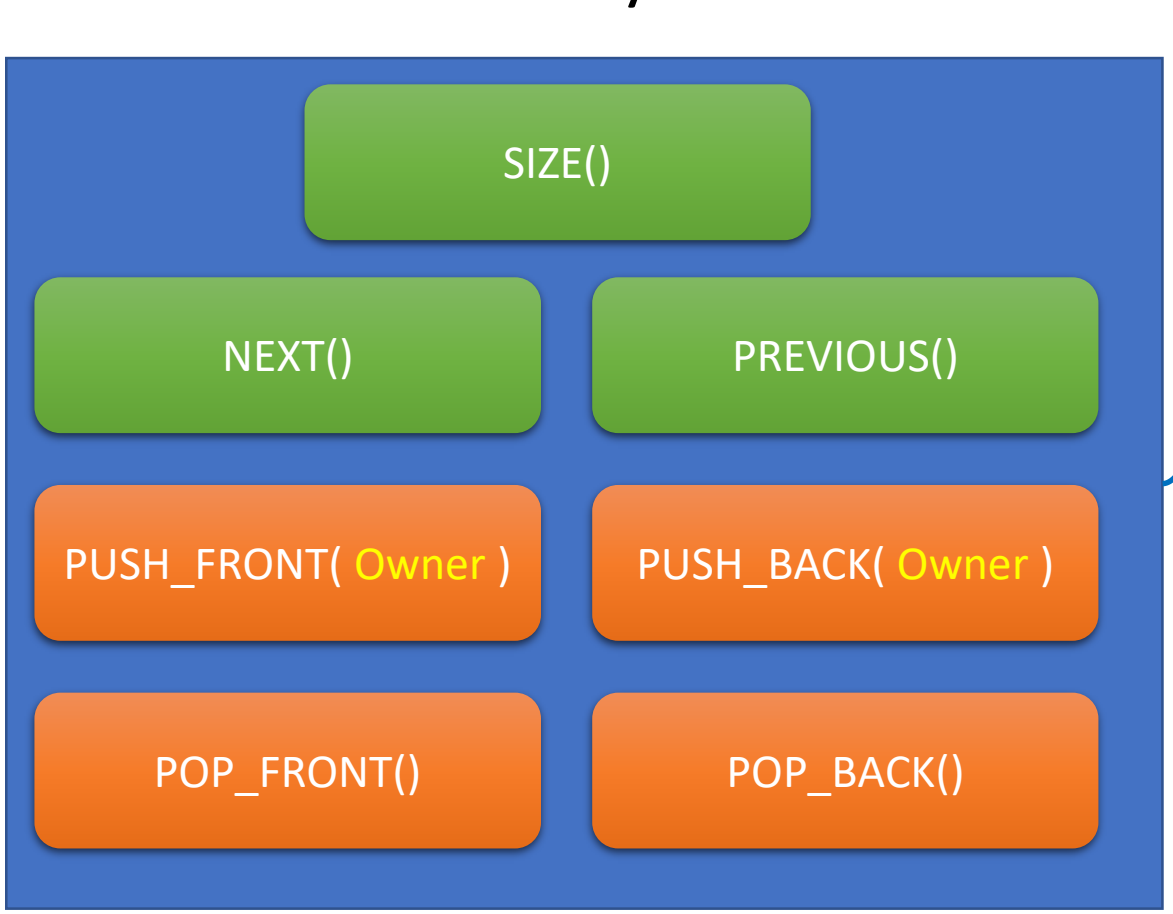
Not Just Best of Friends!

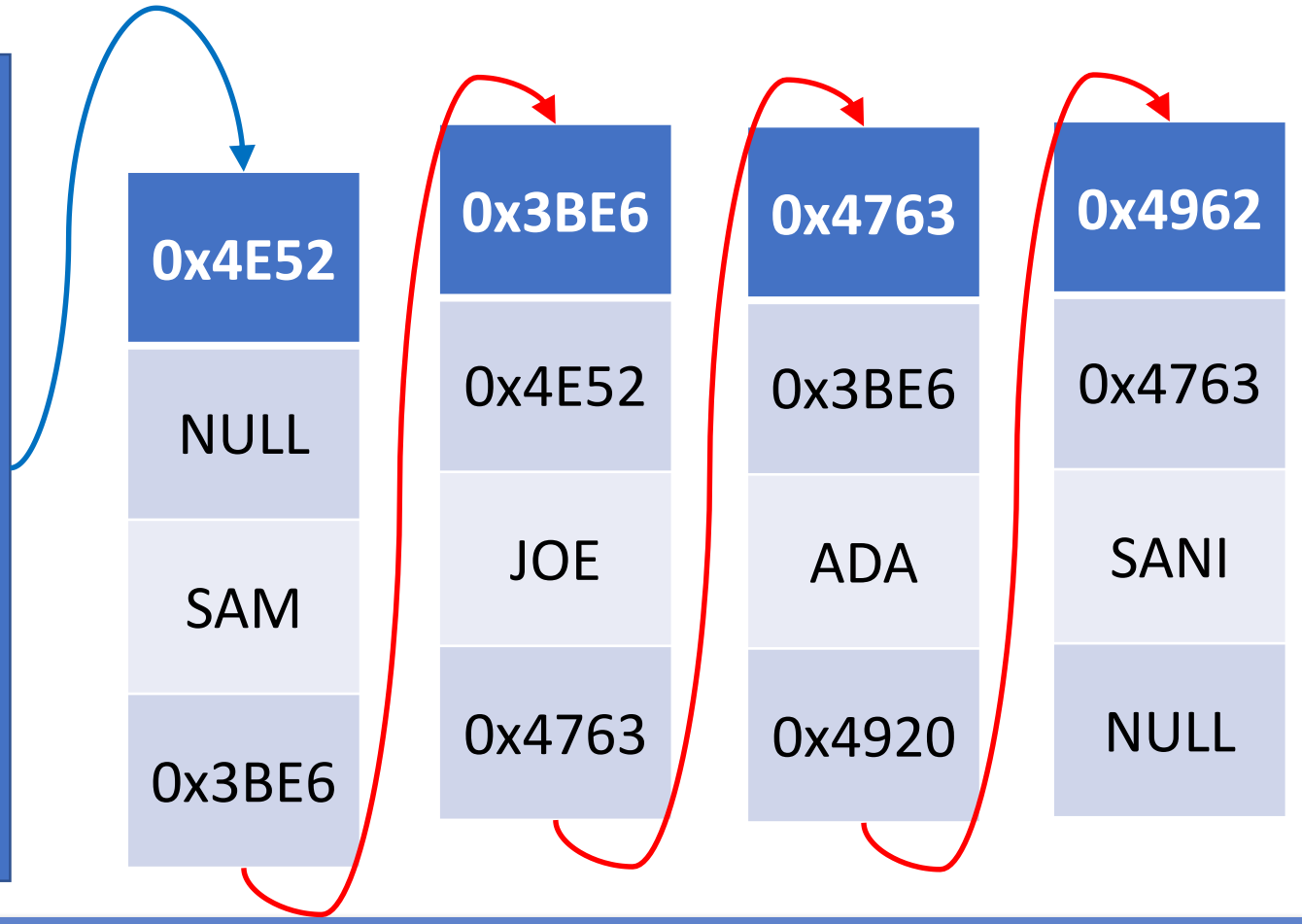
- Adding ADA



Not Just Best of Friends!

- This is a Doubly Linked List…





Not Just Best of Friends!

Questions?

DID YOU KNOW?

In the entire state of Ohio in 1895, there were only two cars on the road, and the drivers of these two cars crashed into each other.

DID YOU KNOW?

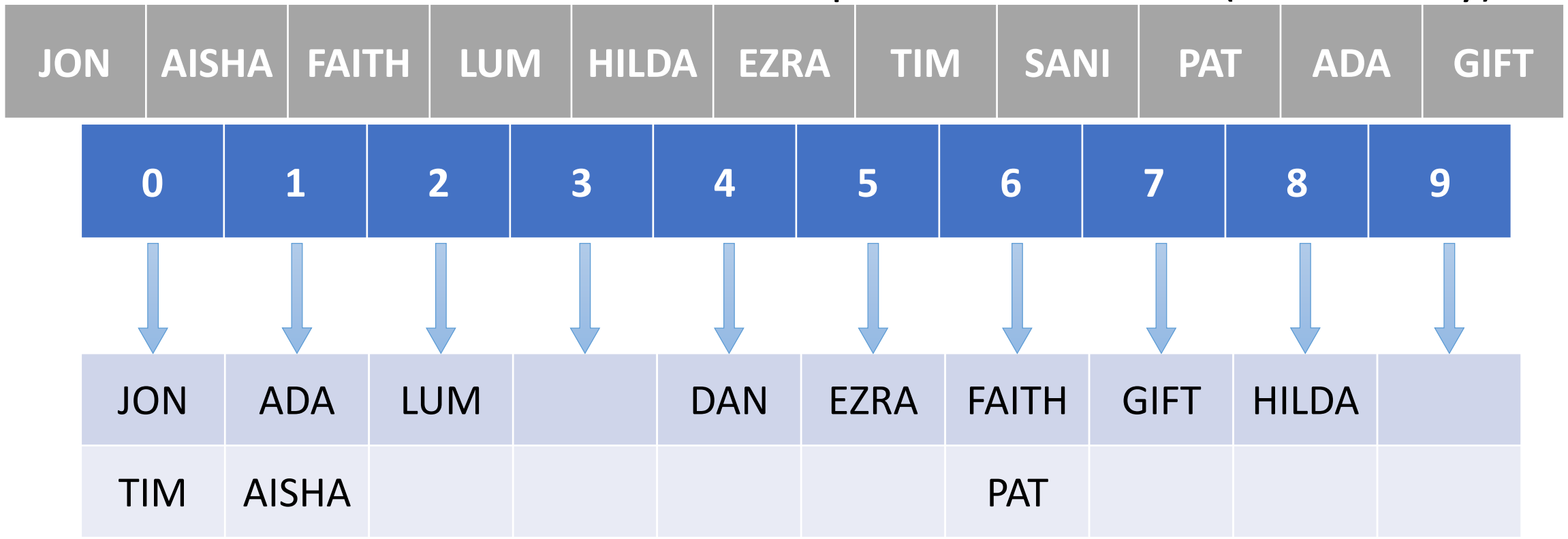
In the entire state of Ohio in 1895, there were only two cars on the road, and the drivers of these two cars crashed into each other.

Directorate of Amebo

Knows exact address of everyone!

Directorate of Amebo

- Such a Function known as “HASH”
- “HASH” does some Mathematics and produces an Index (consistently)



Directorate of Amebo

- Our Hash: First Letter of the Word
- Map-Assign it to numbers from A~Z. If the number is greater than our bucket size, find the remainder of bucket size into the mapped number
-

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	E	F	G	H	I	J	K	L	M

14	15	16	17	18	19	20	21	22	23	24	25	26
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

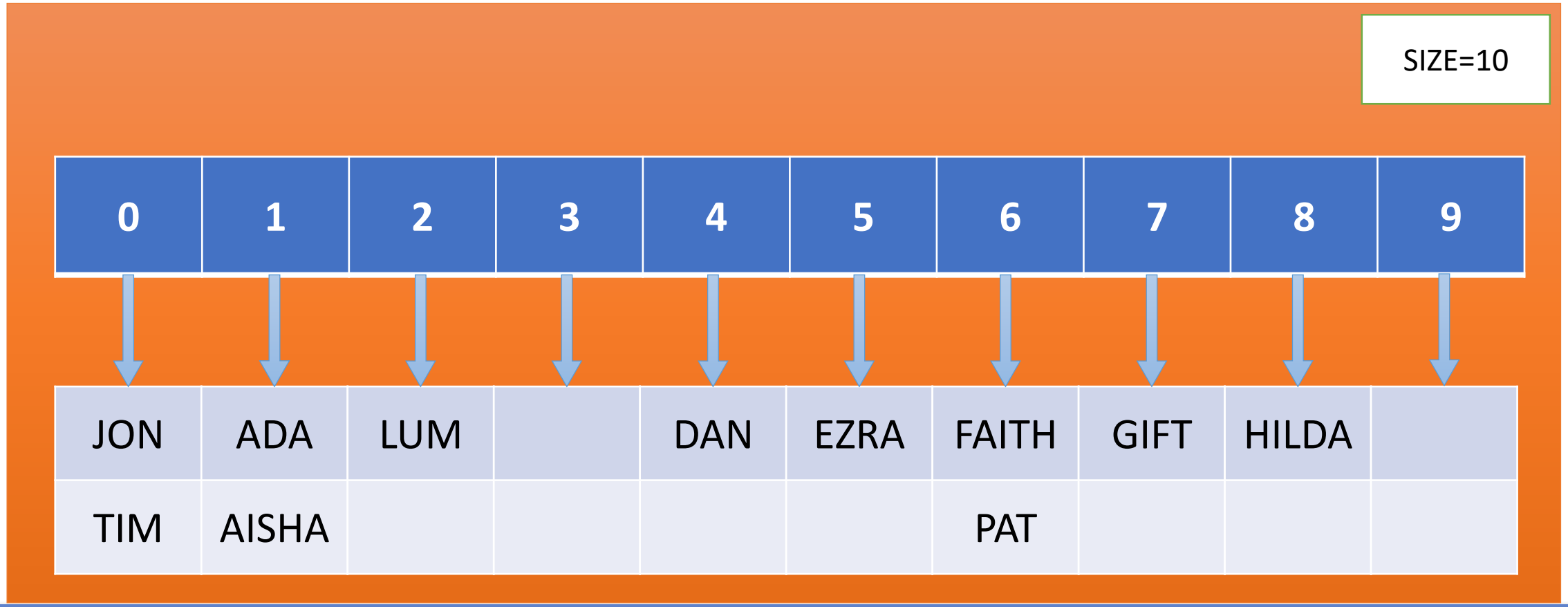
Directorate of Amebo

- Using the hashing scheme described previously, how do we hash the data below into a HashTable of 10 buckets?

JON	AISHA	FAITH	LUM	HILDA	EZRA	TIM	SANI	PAT	ADA	GIFT

Directorate of Amebo

- Manager: knows the hashing method



Directorate of Amebo

- Such a Function known as “HASH”
- “HASH” does some Mathematics and produces an Index (consistently)

0	1	2	3	4	5	6	7	8	9
SAM	JOE	ADA	KEFAS	FAITH	TIM	HILDA	AISHA	DAN	LUM
ADA	JOE	KEFAS	LUM	DAN		FAITH		HILDA	
AISHA		TIM							

Take Home Points

- Data Structures should be used appropriately
- There is always a trade-off
- Minimize the trade-offs

Arrays

- Linear
- Contiguous
- Cache friendly
- Fast traversal
- Fast insertion at growth end
- Slow insertion at arbitrary locations
- Slow deletion at arbitrary locations
- Random Access

Singly Linked Lists

- Sparse, Data is scattered
- Fast Insertion at any known Point
- Fast Deletion at any known Point
- A bit slower to traverse
- No Random Access
- One way traversal
- Requires traversal to know size

Doubly Linked Lists

- Sparse, Data is scattered
- Fast Insertion at any known Point
- Fast Deletion at any known Point
- A bit slower to traverse
- No Random Access
- One way traversal
- Size is known

Hash Set/Map

- Sparse
- Fastest Lookup
- Cannot Be Ordered, Unkown Order
- A bit slow to traverse
- Fast Random Access
- Size is known

The End

Questions?

Homework

1. Develop a better hashing algorithm for any ***string*** in our Futuristic *HashTable*
2. Develop a better hashing algorithm for any ***integer*** in our Futuristic *HashTable*
3. Design an *array-like* data structure that does not require reallocation when adding elements like *standard arrays* do