Data Structures

Array Lists

CS 225
Brad Solomon & G Carl Evans

September 1, 2023

No class
on Monday
Sept 4

UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

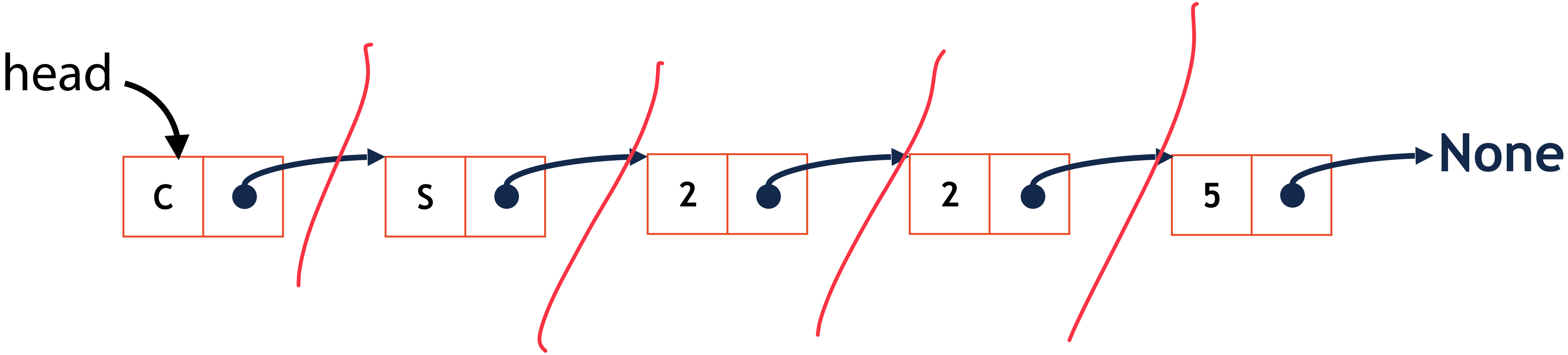
Review fundamentals of array list

Introduce array list implementations

Consider extensions to lists

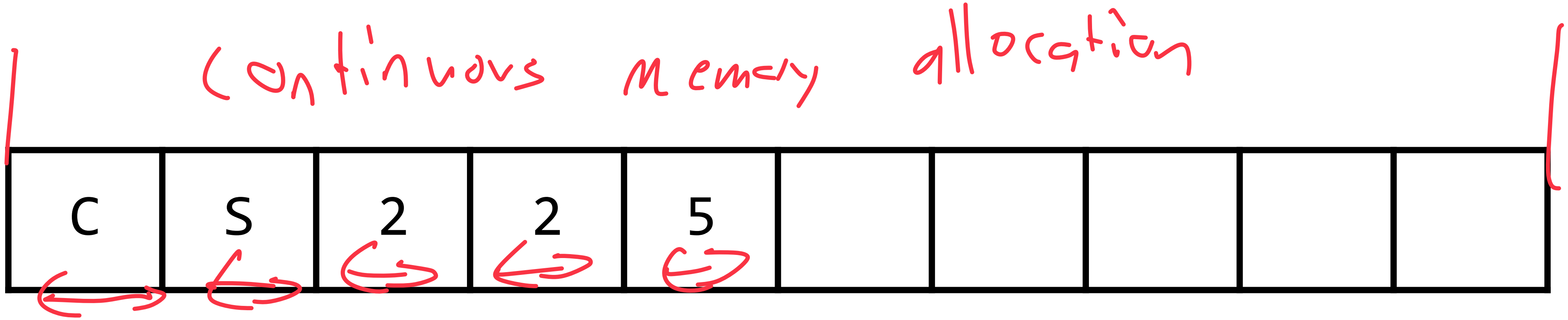
List Implementations

1. Linked List — singly linked list



2. Array List

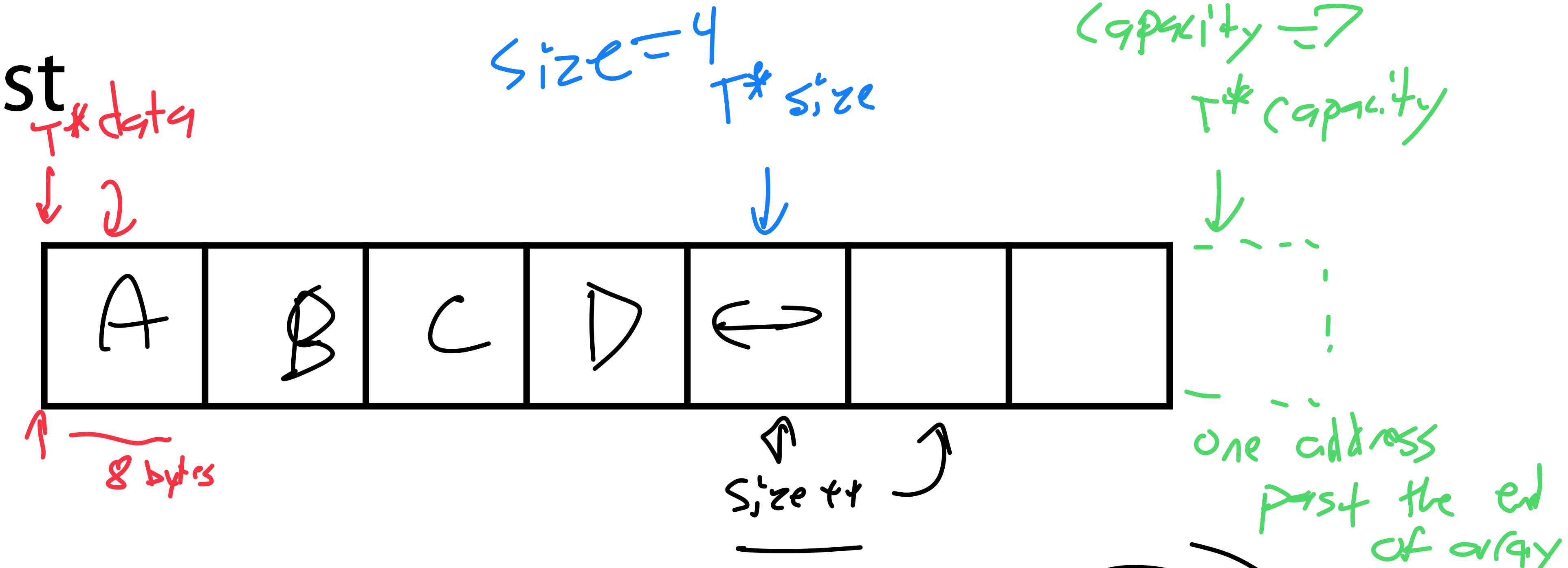
continuous memory allocation



string
int
char

homogenous

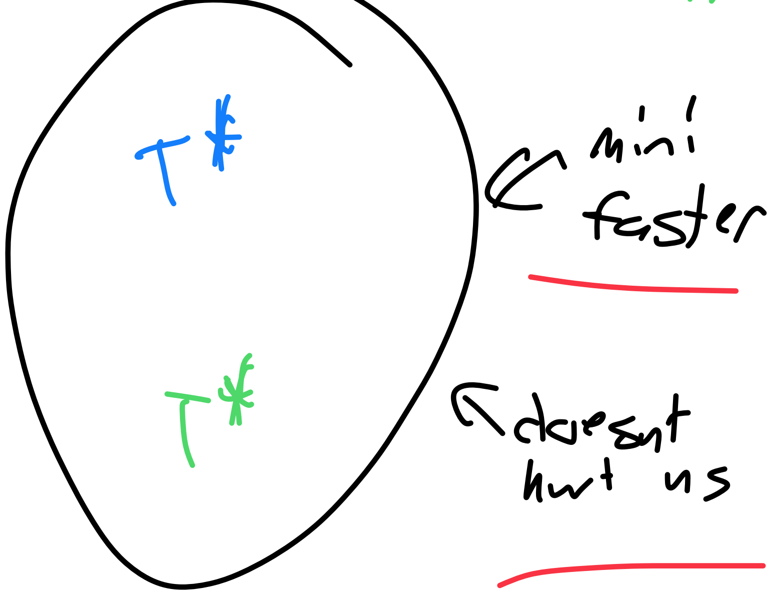
Array List



1) Pointer to array (location)

2) Size - current # of items (unsigned int)

3) Capacity - Max # of items (unsigned int)



List.h

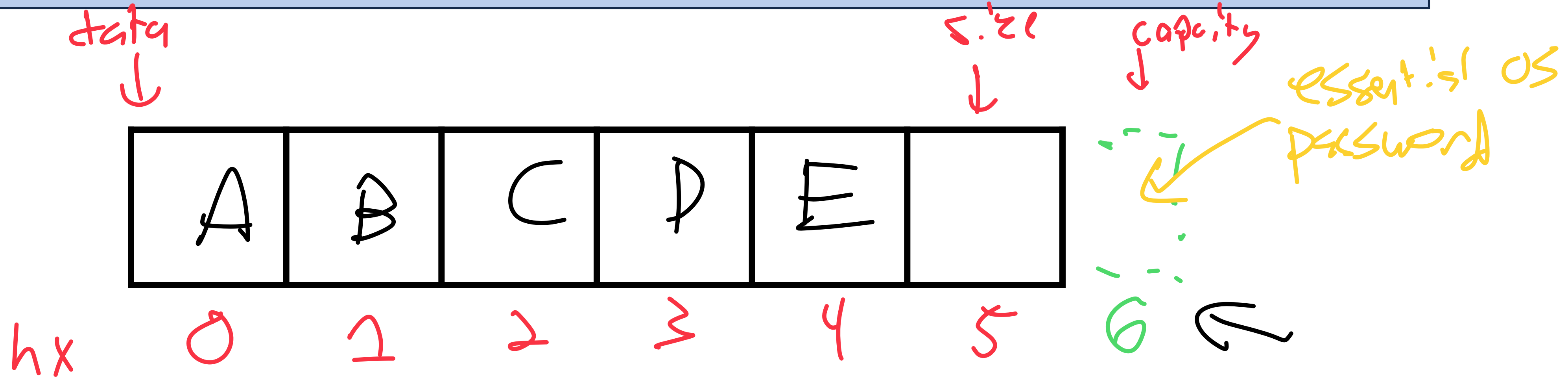
```
#pragma once

template <typename T>
class List {
public:
    /* --- */
private:
  T *data_; =0

  T *size; = 5

  T *capacity; =6

    /* --- */
};
```

$$\# \text{ of objects} = \frac{size - data}{sizeof(T)}$$

$$\# \text{ of possible objects} = \frac{capacity - data}{sizeof(T)}$$

this is a choice ↓ capacity

* size == capacity → array is full



Array List: [ ]

← 3 = i (index)

every object find(data) →

0	1	2	3						
C <T>	S	2	2	5					
↑ 0x0	4	8	~~8~~ 12	16					

Cph:rn T &

← what does C++

$$\underbrace{T^{*}\ data}_{0} + \underbrace{index}_{3} \cdot \underbrace{sizeof(T)}_{4}$$

= 12

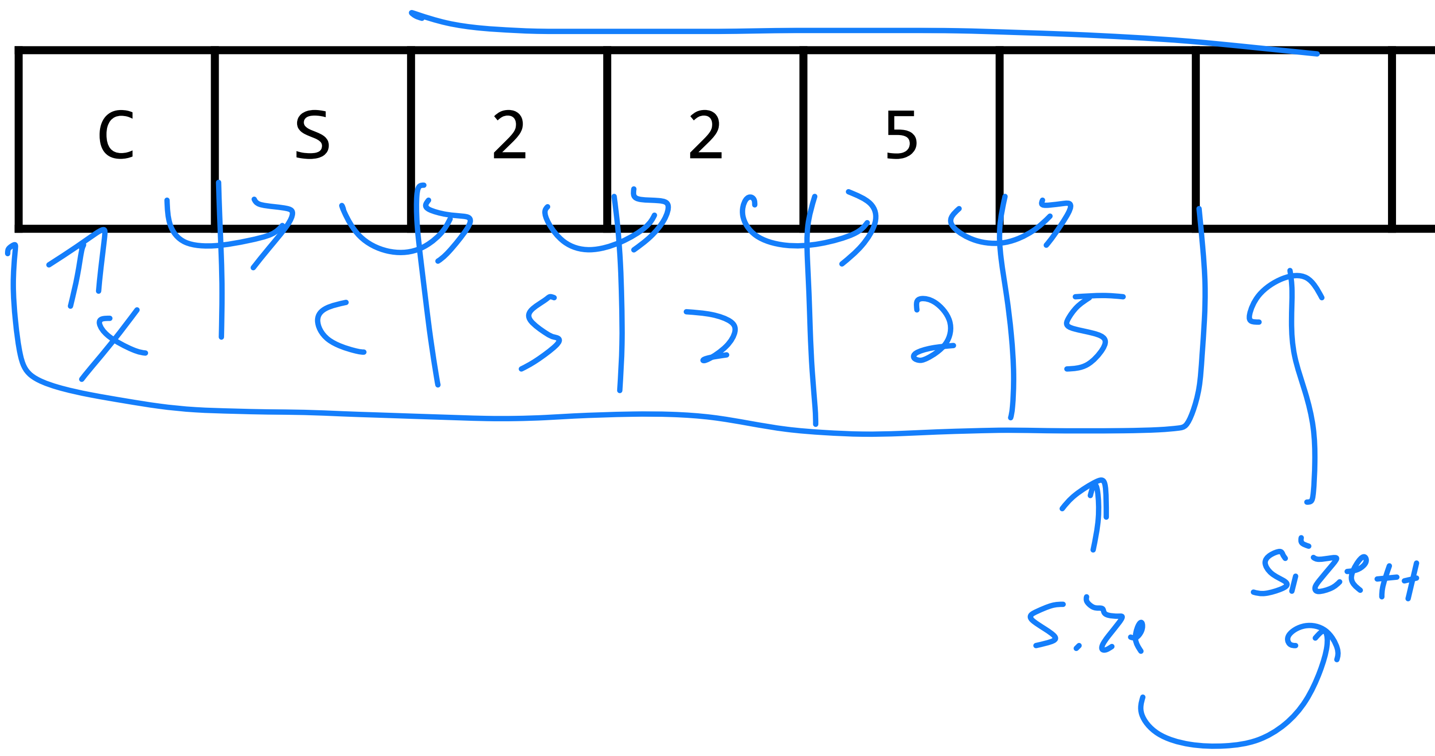
Find address of index @ data

O(1)

random access + index

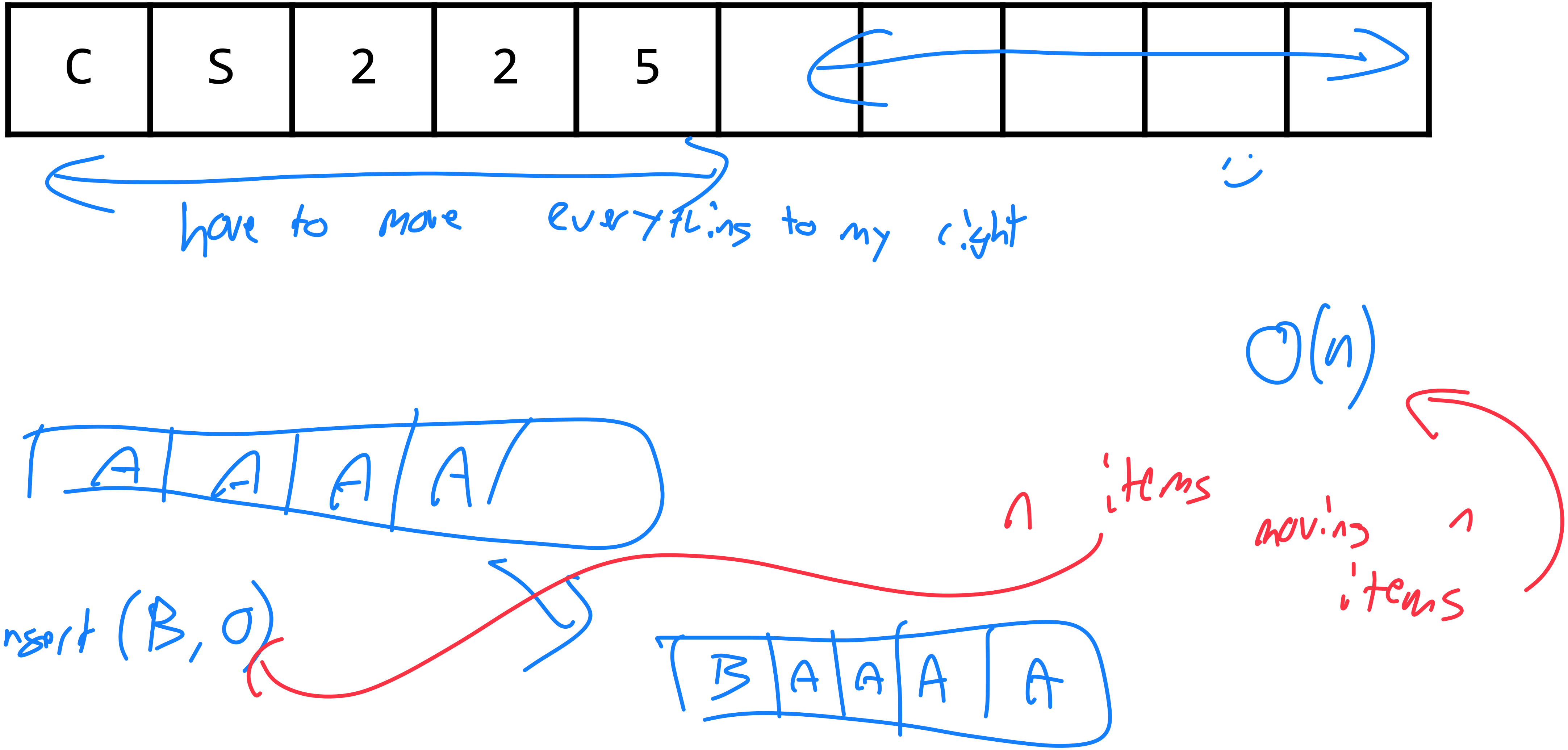
Array List: `insertAtFront(data)`

x



O(n)

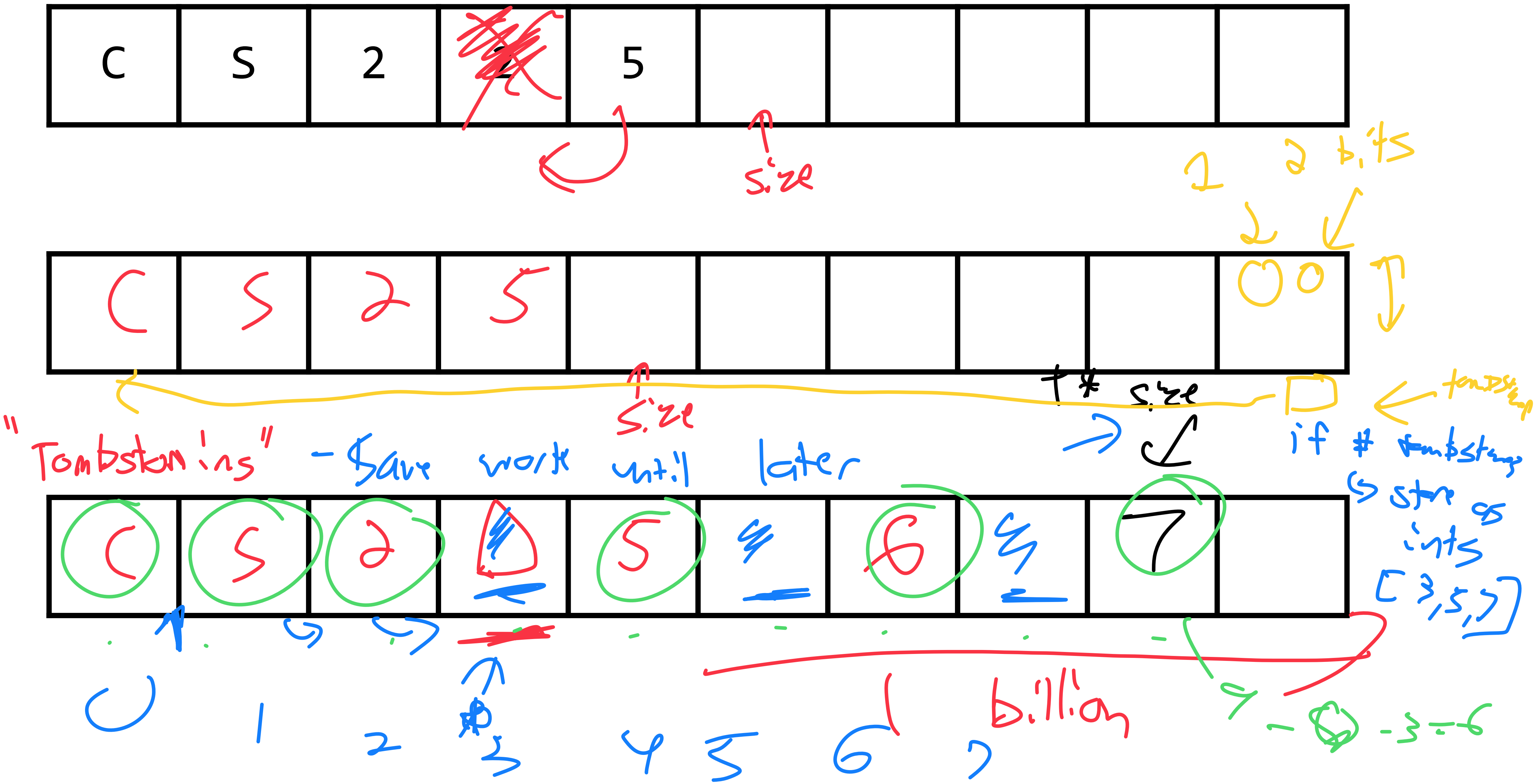
Array List: `insert(data, index)`



C	S	2	2	5					

Array List: remove(index)

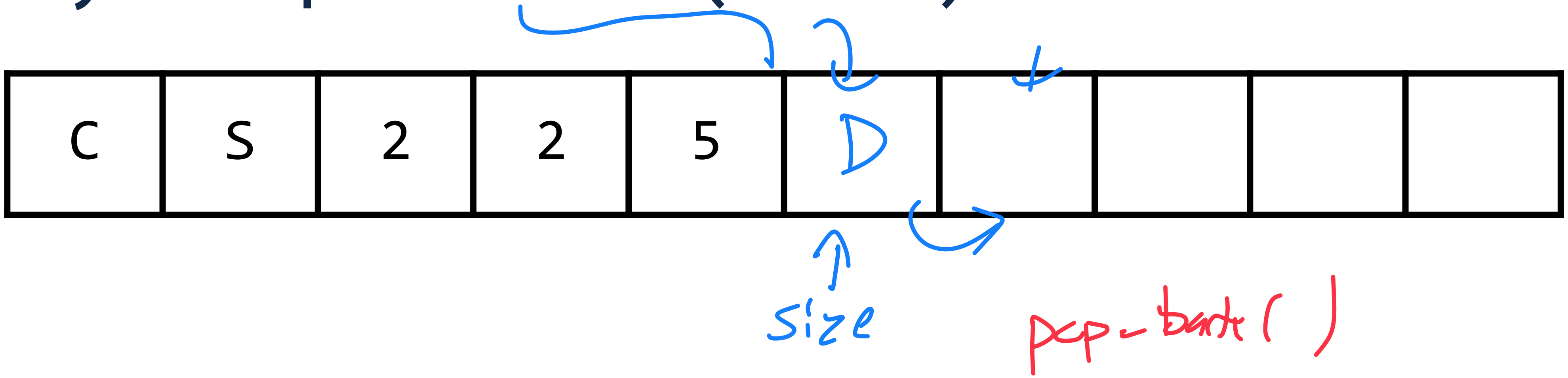
$\frac{size - data}{sizeof(T)}$ - # tombstones



"Tombstoning" - Save work until later

Array List: pushback(data)

insert at back



pop_back()

remove(size)

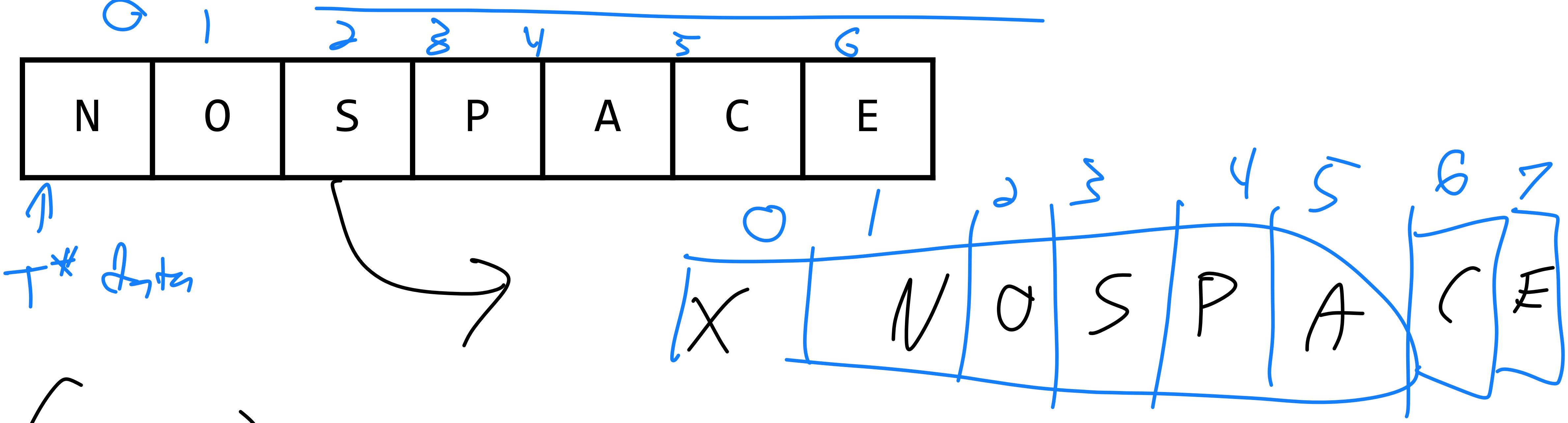
--size;

T *size = data;

size ++;

O(1)

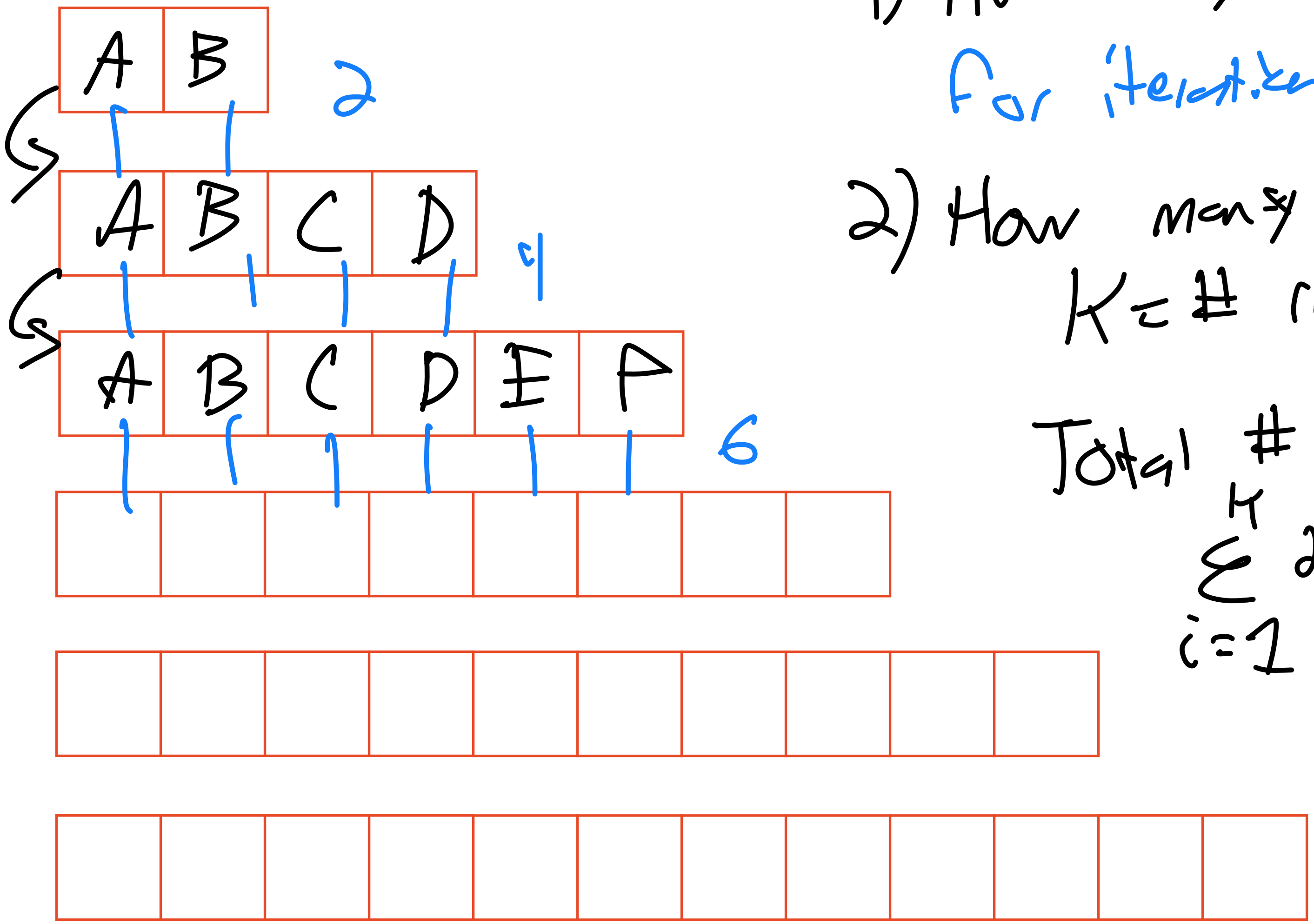
Array List: insert(data, index)



(X, 0)

$O(n)$ to insert

Resize Strategy: +2 elements every time



1) How many copies per realloc?

For iteration i, $2i$ realloc

2) How many reallocs? (N objects total)

$K = \#$ reallocs $= N/2$

Total # of copies

$$\sum_{i=1}^{K} 2i = K(K+1)$$

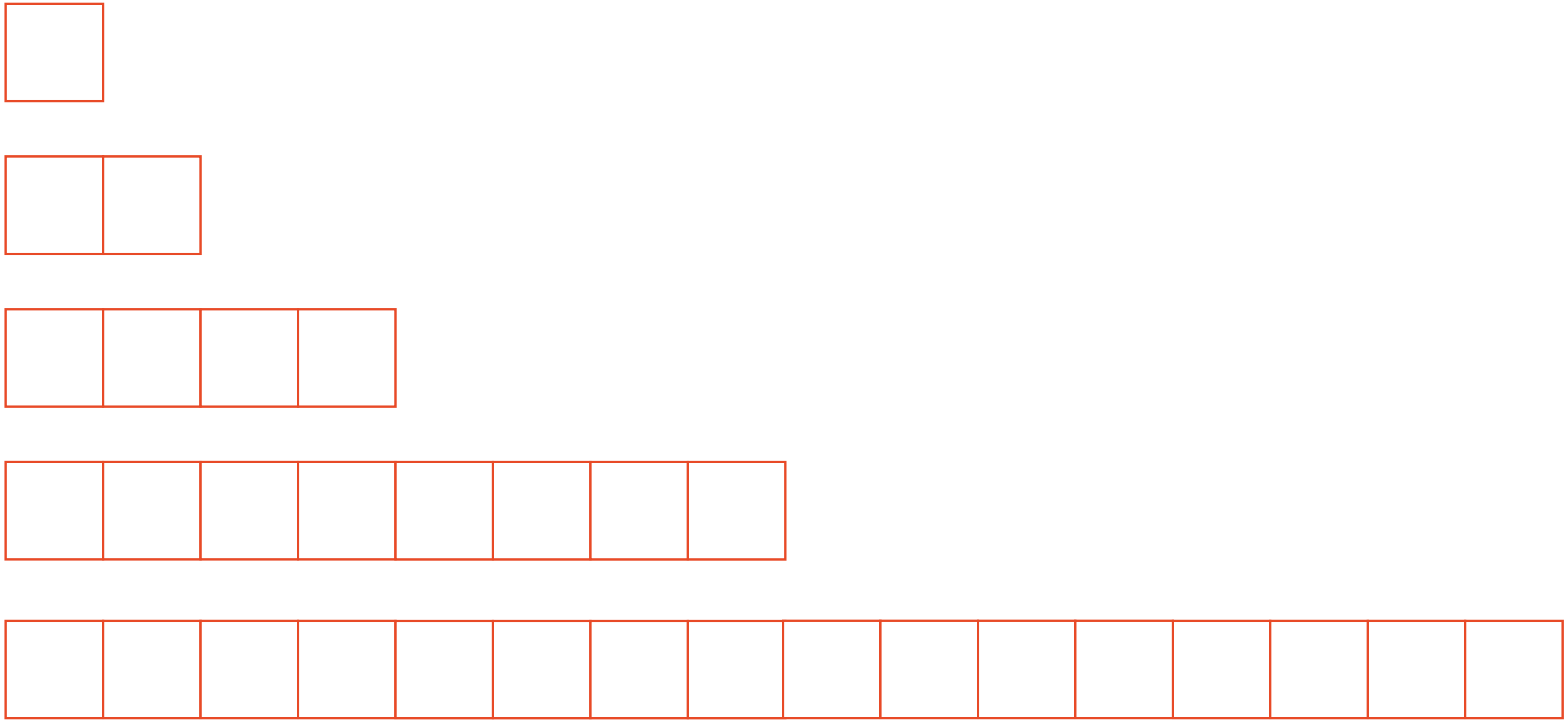
$$K^2 + K$$

$$\frac{N^2}{4} + \frac{N}{2} = \frac{N^2+2N}{4}$$

for N insertions

Resize Strategy: +2 elements every time

Resize Strategy: x2 elements every time



Resize Strategy: x2 elements every time

Array Implementation

	Singly Linked List	Array
Look up **arbitrary** location		
Insert after **given** element		
Remove after **given** element		
Insert at **arbitrary** location		
Remove at **arbitrary** location		
Search for an input **value**		

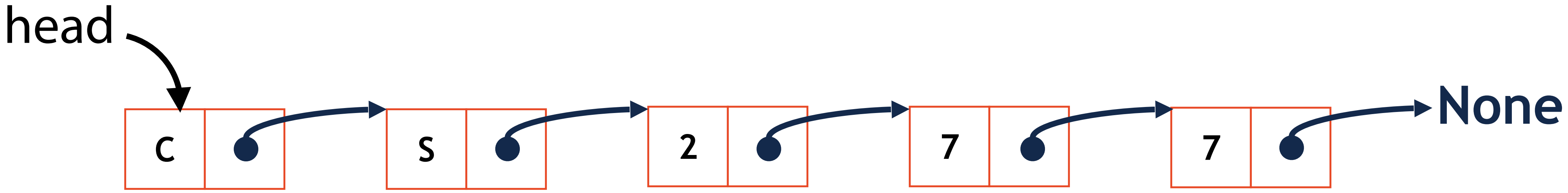
Thinking critically about lists: tradeoffs

The implementations shown are foundational.

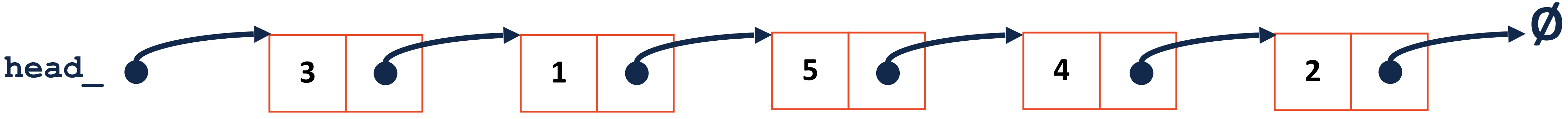
Can we make our lists better at some things? What is the cost?

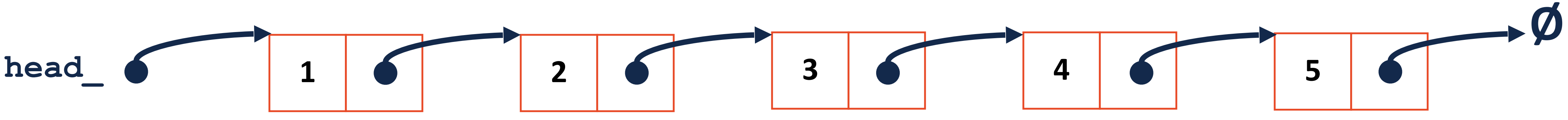
Thinking critically about lists: tradeoffs

Getting the size of a linked list has a Big O of:



Thinking critically about lists: tradeoffs



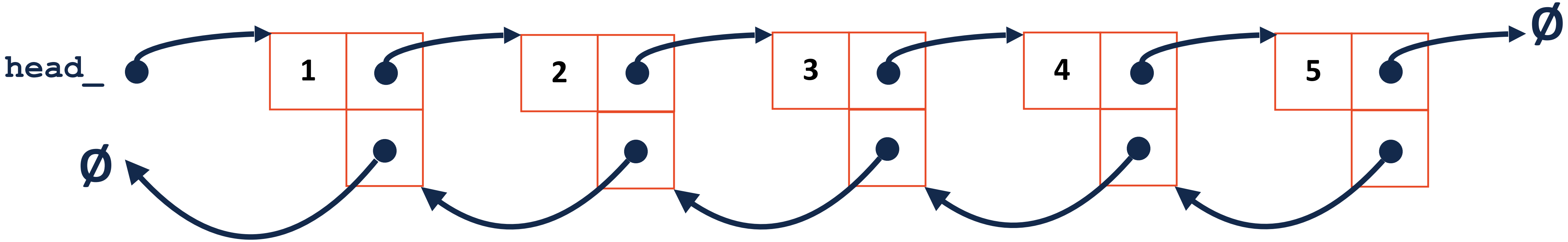


Thinking critically about lists: tradeoffs

2	7	5	9	7	14	1	0	8	3

0	1	2	3	5	7	7	8	9	14

Thinking critically about lists: tradeoffs



Thinking critically about lists: tradeoffs

When we discuss data structures, consider how they can be modified or improved!

Next time: Can we make a ‘list’ that is O(1) to insert and remove? What is our tradeoff in doing so?