Algorithms and Data Structures for Data Science

Lists and Asymptotic Efficiency Review

CS 277
Brad Solomon

February 8, 2023

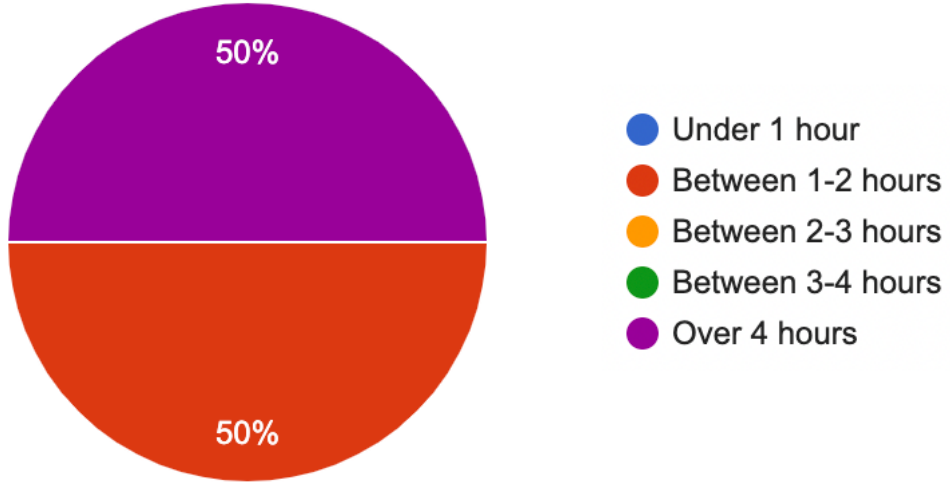
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Lab_parsing Feedback

Average score: 92%

PL average time: 77 minutes



Class helpful to only half of students (as far as completing lab)

Lab generally improved most but not all confidence; some lost confidence

Many issues with the clarity of questions

Assignments are released on the website

Look on the website for examples and the full details of an assignment

I'll remove the descriptions in the notebook — at least for now

I'll put back in working examples in the notebooks

Exam 1

Exams will be proctored by the CBTF: https://cbtf.engr.illinois.edu/

(That link will have a link to **Prairietest**, where you can sign up for exam 1)

Reservations open on February 2nd @ 9 AM

You must take the exam sometime between 2/14 and 2/16!

See website for expected content:

https://courses.grainger.illinois.edu/cs277/sp2023/exams/

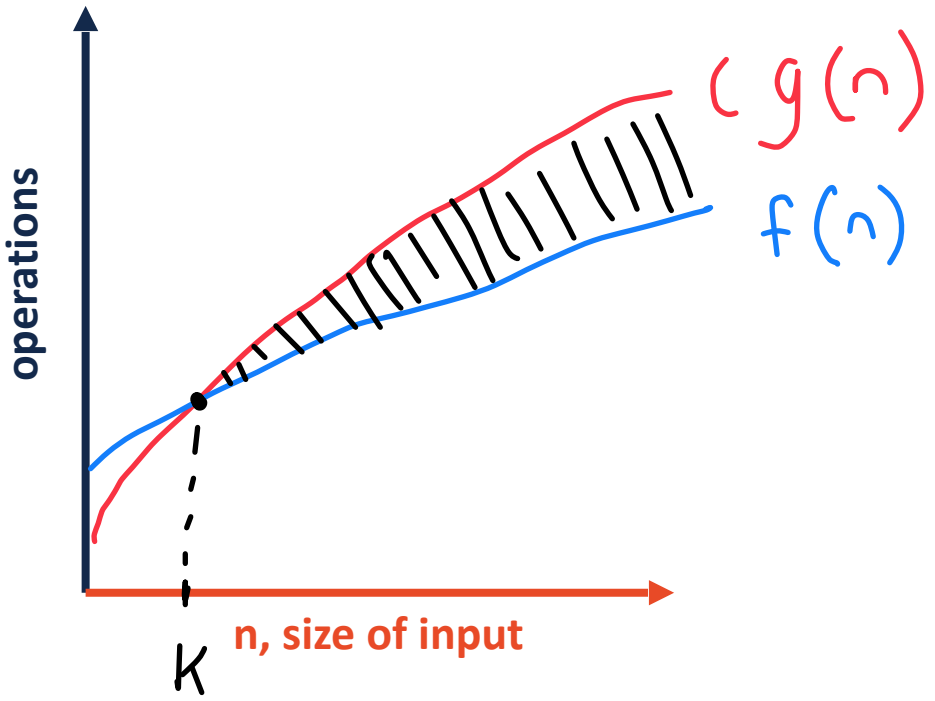
Learning Objectives

Review asymptotic efficiency

Compare list implementations using Big O

Big-O notation

$f(n)$ is $O(g(n))$ iff $\exists c, k$ such that $f(n) \leq cg(n)\ \forall n > k$



1) $cg(n)$ is an upper bound on $f(n)$

2) This is true for all input values larger than some arbitrary k

In-Class Exercise

What is the big O for the following functions?

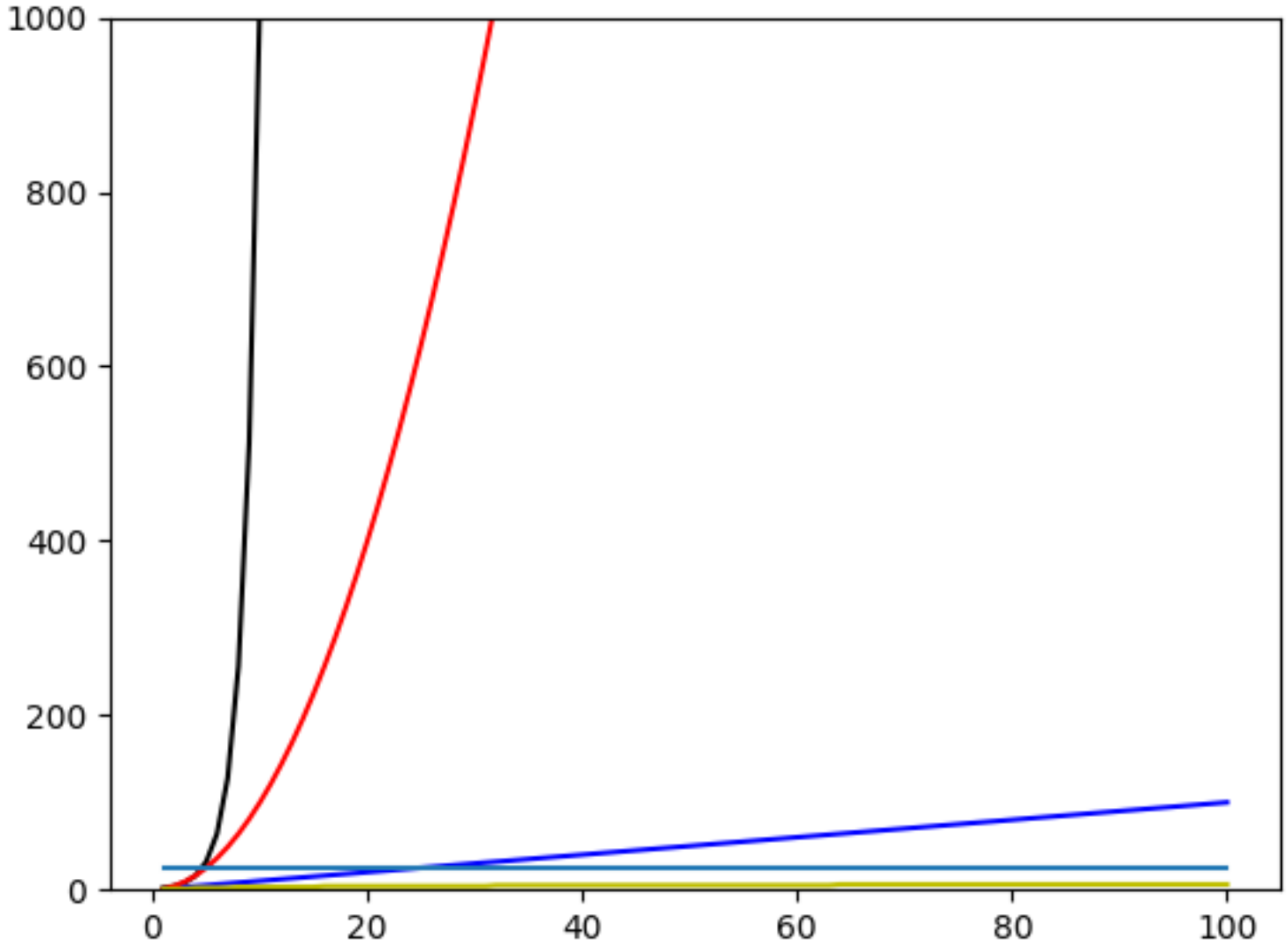
$$a(n) = n^4 + 50n + 10$$

$$b(n) = 500nlog\ n + 50n + log(n)$$

$$c(n) = n^3 + 3n! + 12$$

$$d(n) = n^2 + nlog\ n$$

Big-O Complexity Classes



- $O(2^n)$
- $O(n^2)$
- $O(n)$
- $O(\log n)$
- $O(1)$

Identifying the Big O of an algorithm

1) Label the key factors that drive algorithm performance

2) Write out the worst-case performance for each step

3) Identify (or reduce to) the largest terms for each factor

Big O of arbitrary code

```
1  def doStuff(inList1, inList2):
2
3      c1 = 0
4      for i in inList1:
5          c1+=1
6
7
8
9      c2 = 0
10     for v1 in inList1:
11         for v2 in inList2:
12             c2+=1
13
14
15
16     return c1, c2
17
18
19
20
21
22
23
```

Big O of arbitrary code

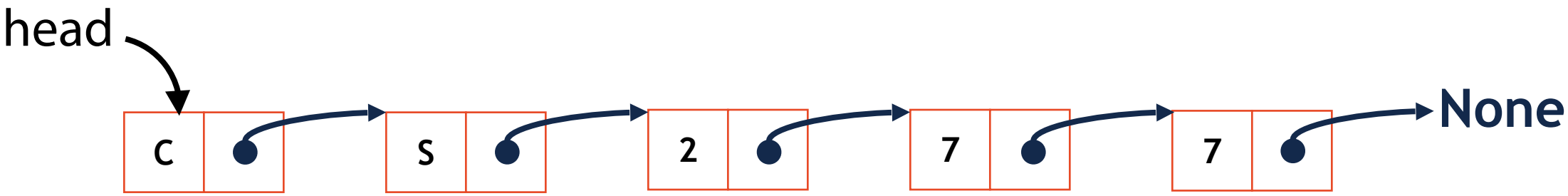
```
 1 def doStuff2(inList):
 2     ops = 0
 3     size = len(inList)
 4     while size > 0:
 5         size = int(size / 2)
 6         ops+=1
 7     return ops
 8
 9 def doStuff3(inList1, inList2):
10     ops = 0
11     for i in inList1:
12         ops+= doStuff2(inList2)
13     return ops
14
15
16
17
18
19
20
21
22
23
```

Big O of arbitrary code

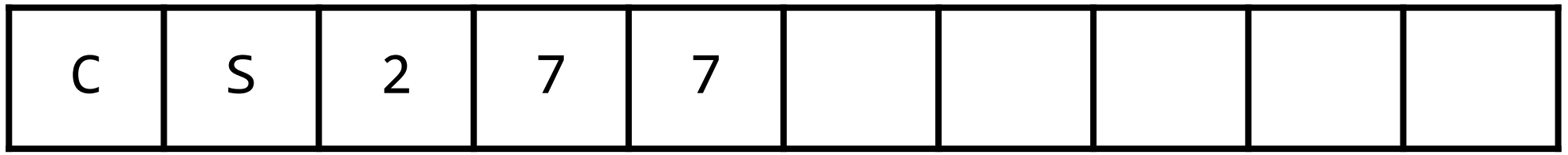
```
 1  def convert_1D_to_2D(inList, rowSize):
 2      listLen = len(inList)
 3      numRows = math.ceil(listLen/rowSize)
 4
 5      outList = []
 6      count = 0
 7
 8      ops = 0
 9      for i in range(numRows):
10          tempList = []
11
12          for j in range(rowSize):
13
14              if count >= listLen:
15                  tempList.append(-1)
16              else:
17                  tempList.append(inList[count])
18
19              ops+=1
20              count+=1
21
22          outList.append(tempList)
23
24      print(ops)
25      return outList
```

List Implementations

1. Linked List



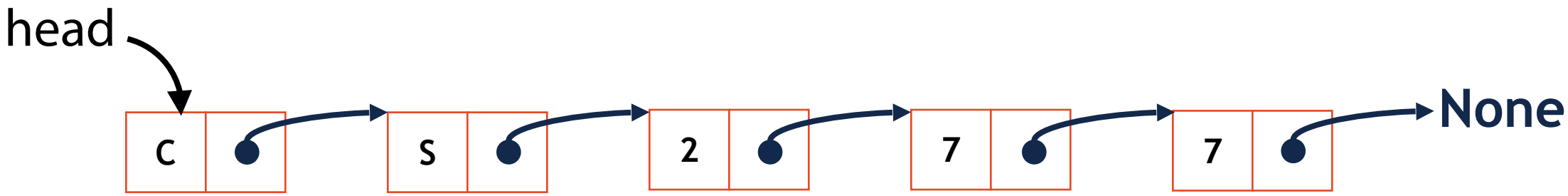
2. Array List



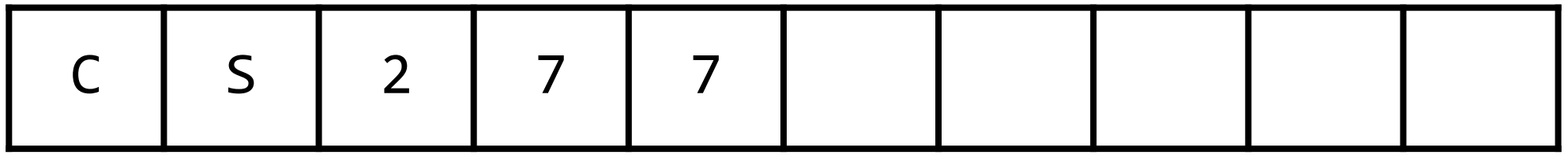
List Implementation Big O

`__getItem__(  )`

1. Linked List



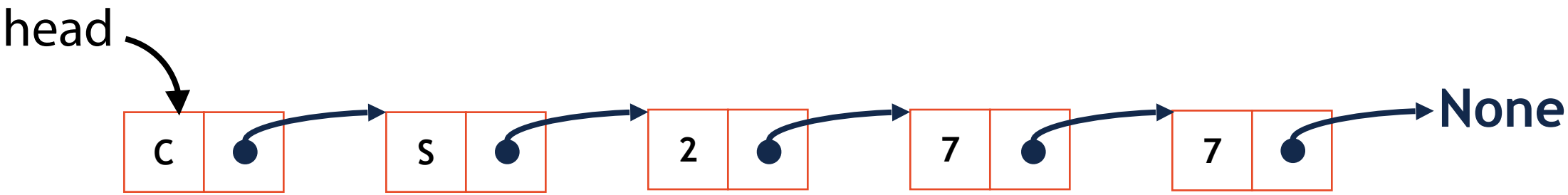
2. Array List



List Implementation Big O

`__getItem__(  )`

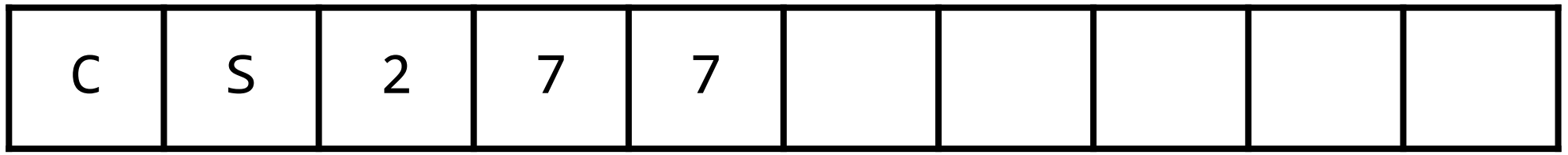
1. Linked List



List Implementation Big O

`__getItem__(  )`

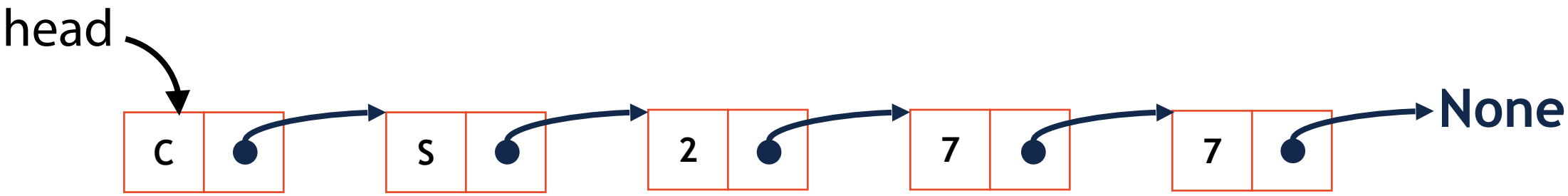
2. Array List



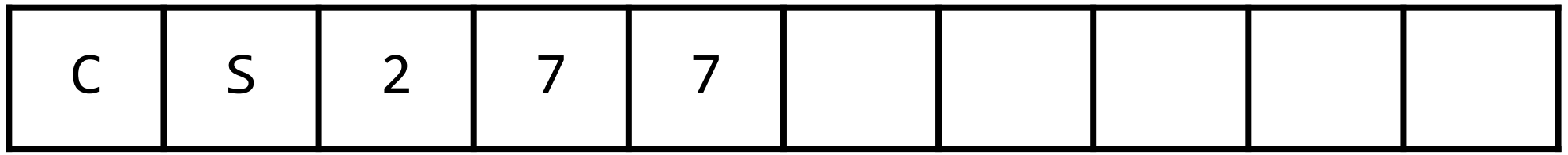
List Implementation Big O

`__find__(  )`

1. Linked List



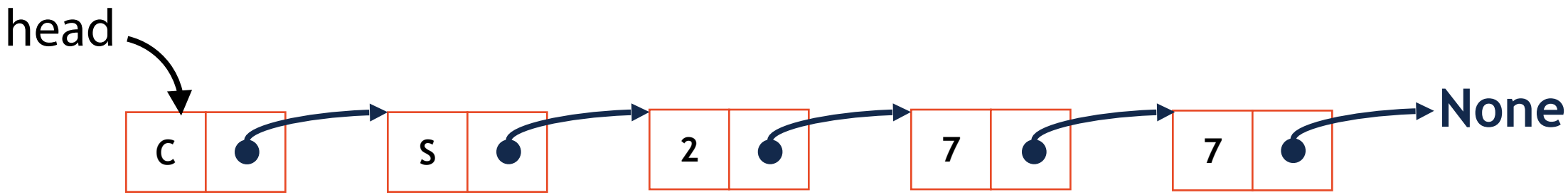
2. Array List



List Implementation Big O

`__find__(  )`

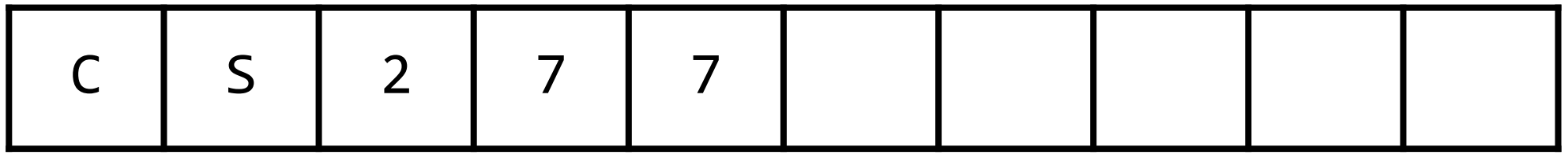
1. Linked List



List Implementation Big O

`__find__(  )`

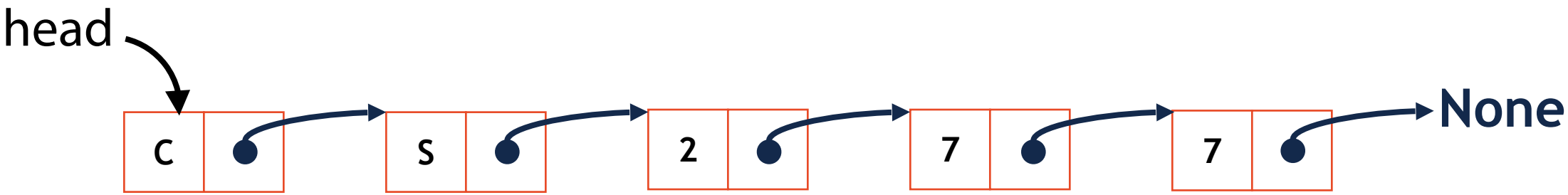
2. Array List



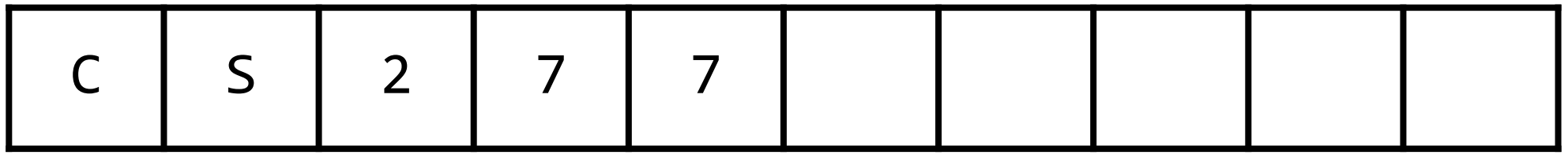
List Implementation Big O

`insertFront(  )`

1. Linked List



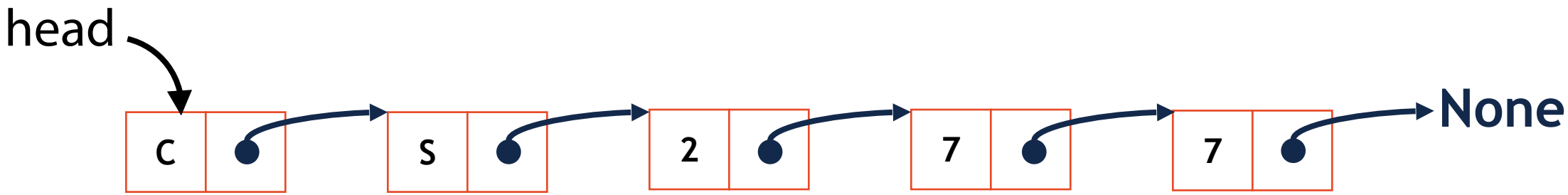
2. Array List



List Implementation Big O

`insertFront(  )`

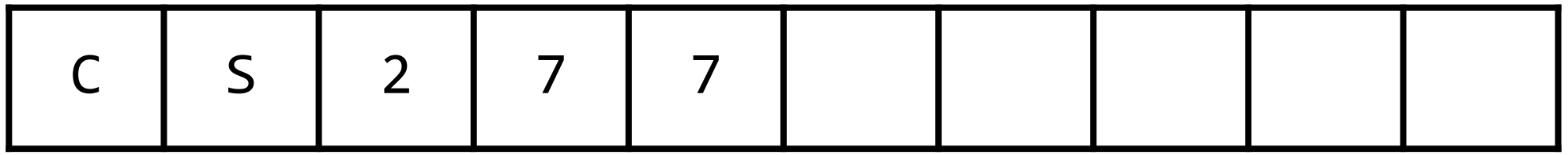
1. Linked List



List Implementation Big O

`insertFront(  )`

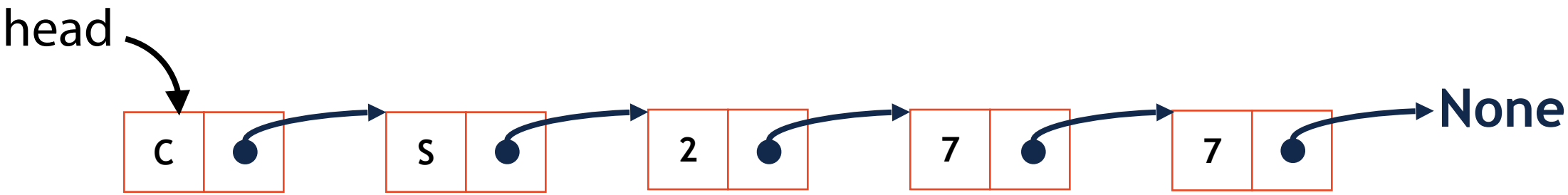
2. Array List



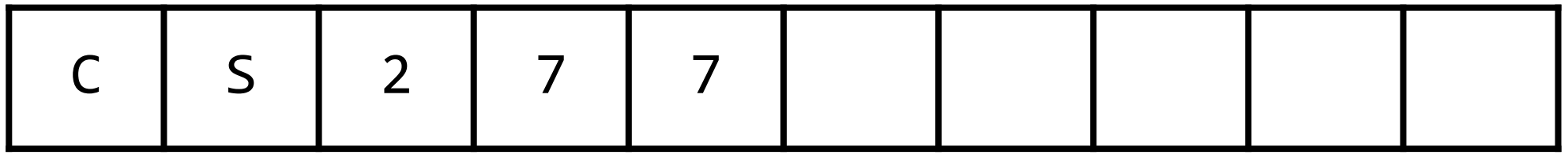
List Implementation Big O

`insertEnd(  )`

1. Linked List



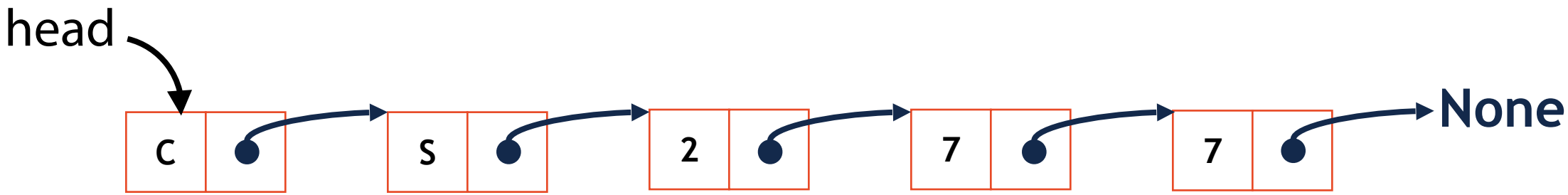
2. Array List



List Implementation Big O

`insertEnd(  )`

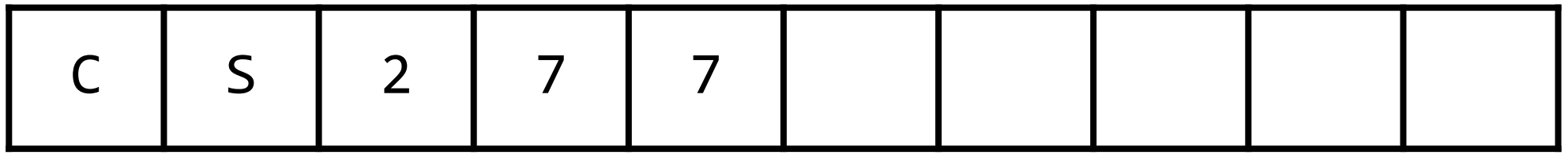
1. Linked List



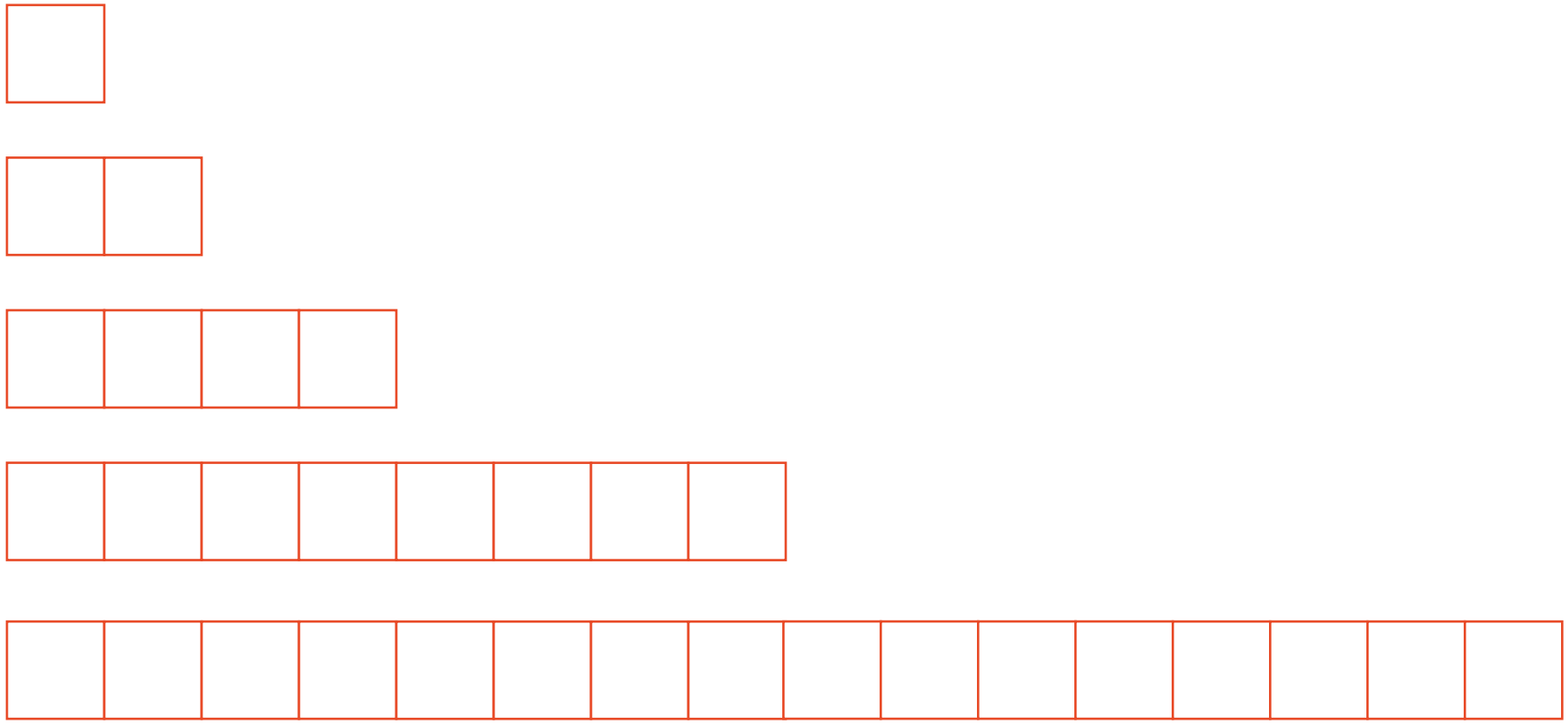
List Implementation Big O

`insertEnd(  )`

2. Array List



Resize Strategy: x2 elements every resize

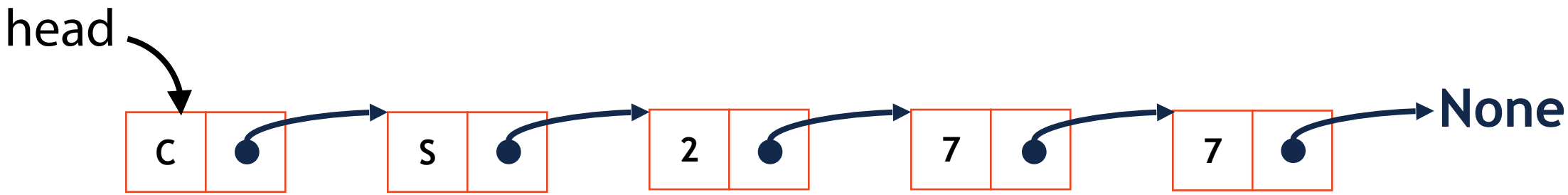


Resize Strategy: x2 elements every resize

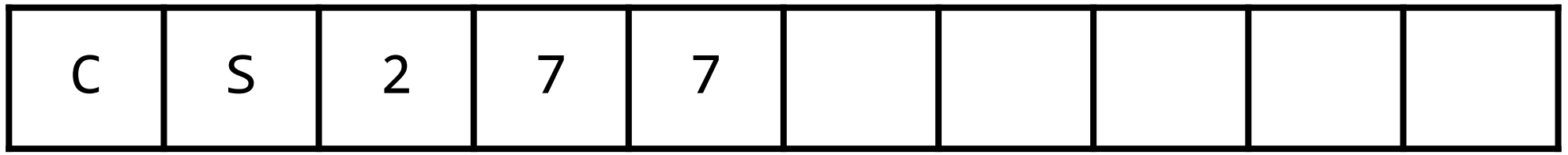
List Implementation Big O

`insert(  ,  )`

1. Linked List



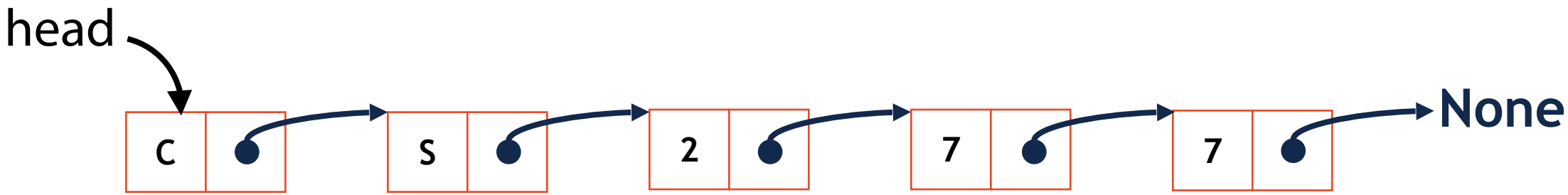
2. Array List



List Implementation Big O

`insert(  ,  )`

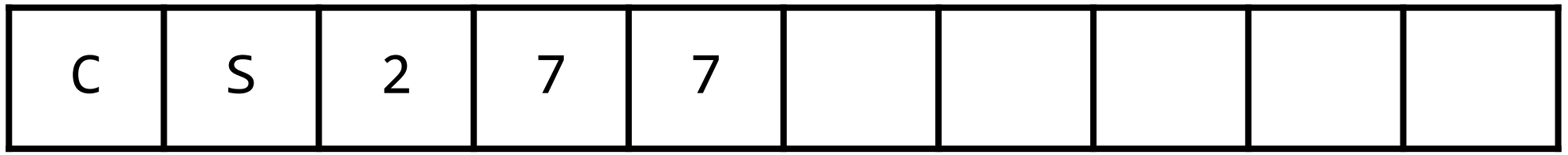
1. Linked List



List Implementation Big O

`insert(  ,  )`

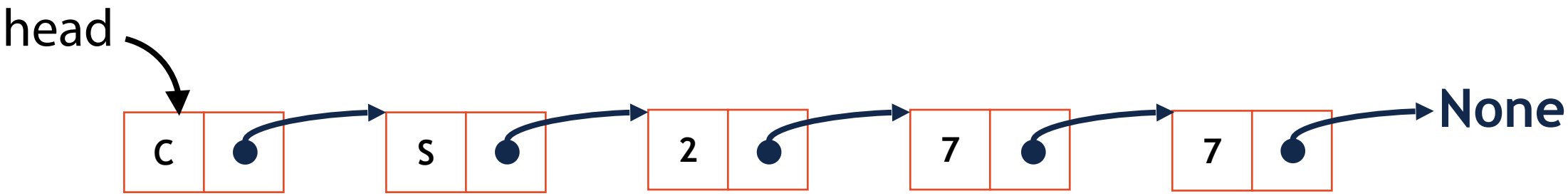
2. Array List



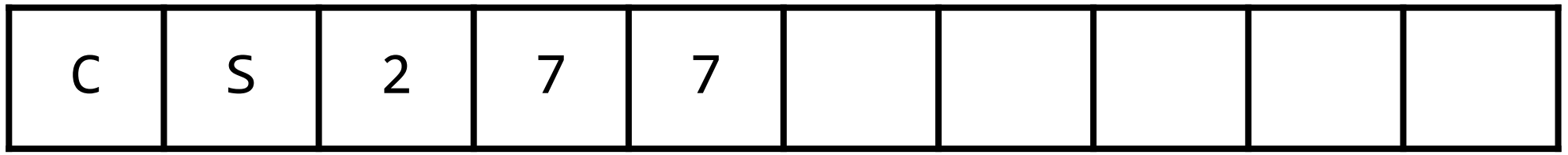
List Implementation Big O

`remove( , )`

1. Linked List



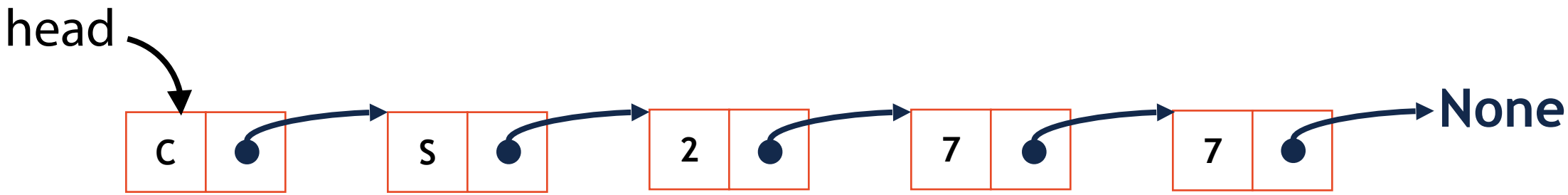
2. Array List



List Implementation Big O

`remove(  ,  )`

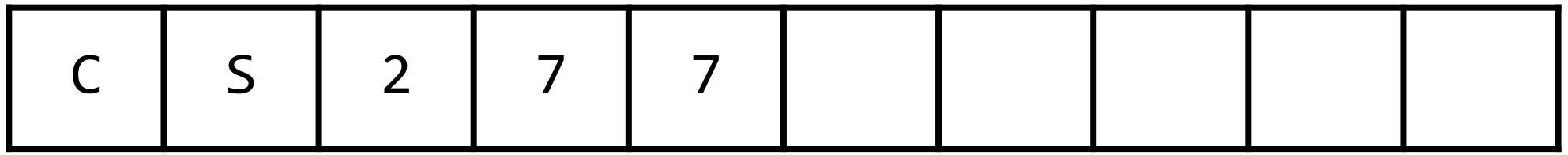
1. Linked List



List Implementation Big O

`remove(  ,  )`

2. Array List



Array Implementation

	Singly Linked List	Array
Look up given an input **position**		
Search given an input **value**		
Insert/Remove at **front**		
Insert/Remove at **arbitrary** location		

When would you use LinkedList vs Array?

Questions?