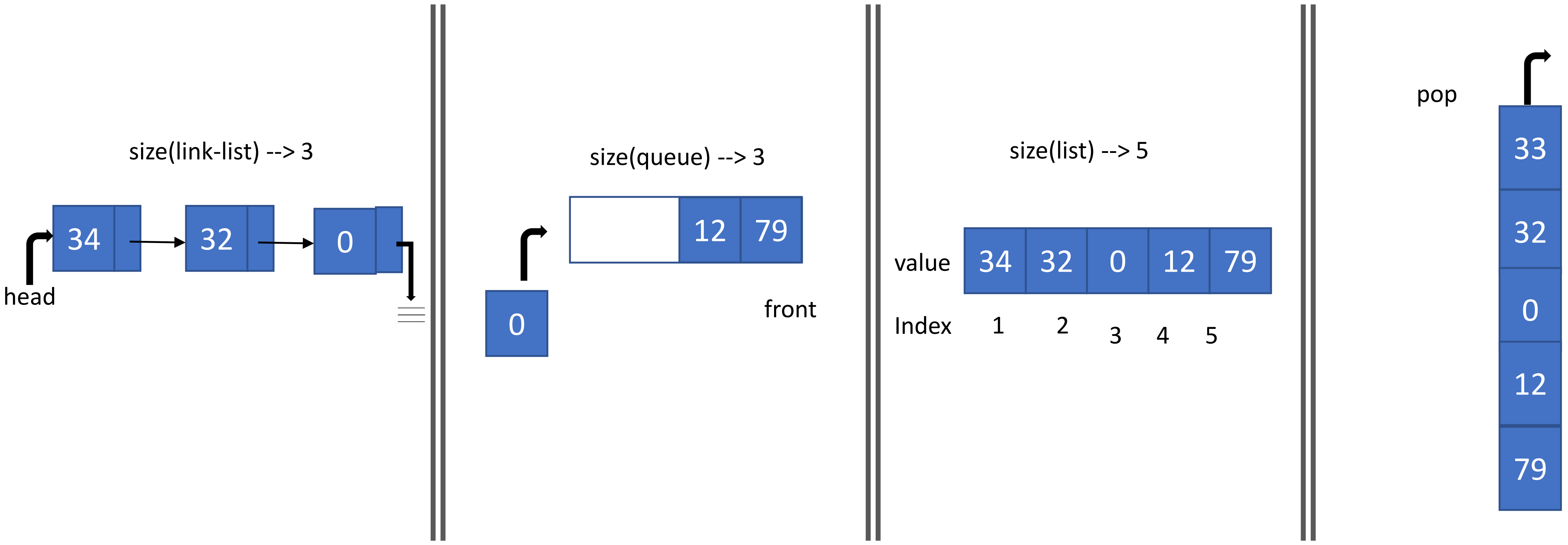
Fundamental Data Structures I

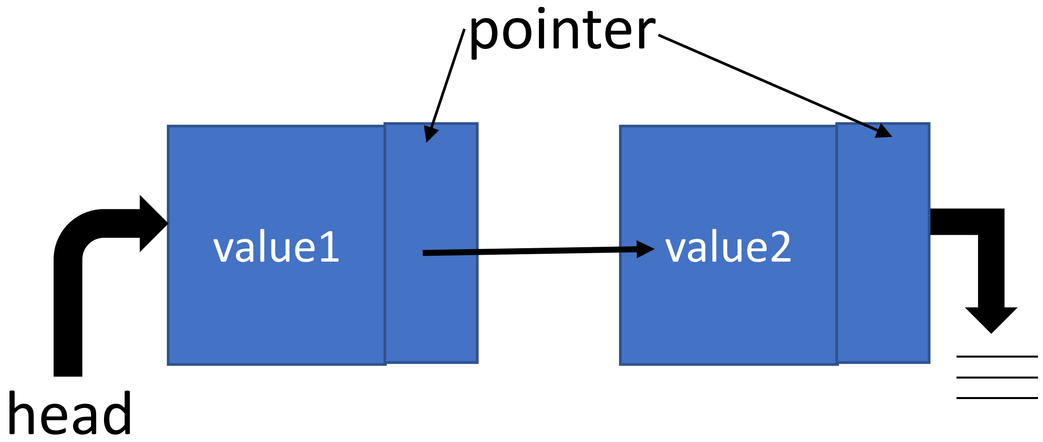
Dr Roberto Murcio



Linear Data Structures

As opposed to Lists/Arrays, linked-lists do not have their order defined by their physical placement in the memory.

Linked-list



Implementation

- Creator

```
// Creating a linked-list
// A linked list node
struct Node
{
  int data;
  struct Node *next;
}; typedef struct list st;
```

Implementation

- Creator
- Observer – delete()

```
void delete(struct Node **head_ref, int position)
{
   struct Node* temp = *head_ref;
    if (position == 0)
    {
        *head_ref = temp->next;
        free(temp);              /
        return;
    }
    for (int i=0; temp!=NULL && i<position-1; i++)
         temp = temp->next;
    if (temp == NULL || temp->next == NULL)
         return;
    struct Node *next = temp->next->next;
    free(temp->next);  // Free memory
    temp->next = next;  // Unlink the deleted node from list
}
```

Implementation

- Creator
- Observer – delete()
- Mutator – add()

```
void add(struct Node** head_ref, int new_data){
  struct Node* new_node=(struct Node*)malloc(sizeof(struct Node));
   new_node->data  = new_data;
   new_node->next = (*head_ref);
   (*head_ref) = new_node;
}
```

UPs and DOWNs

- Efficiently splice new elements into the list or remove existing elements anywhere in the list
- Useful to implement other data structures

- No random access is not allowed.
- Extra memory space for a pointer with each element of the list.

Your turn

Give an algorithm to order a set of numbers stored in a linked-list

Complexity

1

add — O(1)

2

delete — O (n)

n=1 if deleting head or tail

3

search — **O (n)**

LIFO – Last In First Out

Stacks

- Trays on a canteen
- Back/Forward
- Undo/Redo
- CPU architecture

Image Credits: Evgeny Atamanenko / Shutterstock

Implementation

- Creator

```
// Creating a stack
struct stack {
    int items[MAX];
    int top;
}; typedef struct stack st;
```

Implementation

- Creator

- Observer – pop ()

```
// Remove from stack
int pop(st *s) {
  count--;
  return s->top--;
}
```

Implementation

- Creator
- Observer – pop ()
- Mutator – push ()

```
// Add element to a stack
void push(st *s, int newitem) {
   s->top++;
   s->items[s->top] = newitem;
   count++;
};
```

UPs and DOWNs

- Useful for lots of problems
- Very easy to build one from an array.

- No random access. You get the top, or nothing.
- No walking through the stack at all.
- No searching through a stack.

Your turn

Give an algorithm to reverse the words in a sentence (only letters and spaces). You must use a stack data structure.

Complexity

1

pop / push— **O**(1)

2

search — O (n)

(Worst case scenario - Avoid!)

FIFO (First In First Out)

QUEUE

- Ticket counters, supermarkets, banks, etc.
- Call centers.
- CPU scheduling

Implementation

- Creator

```
// Creating a queue – two stacks version

struct sNode {
    int data;
    struct sNode* next;
};

struct queue {
    struct sNode* stack1;
    struct sNode* stack2;
};
```

Implementation

- Creator

- Observer – enQueue()

```
// add element to the queue
enQueue(struct queue* q, int x)
{
    push(&q->stack1, x);
}
```

Implementation

- Creator
- Observer – enQueue()
- Mutator – deQueue()

```c
// Remove elements queue
int deQueue(struct queue* q) {
  int x;
  if (q->stack2 == NULL) {
    while (q->stack1 != NULL) {
        x = pop(&q->stack1);
        push(&q->stack2, x);
    }
  }
  x = pop(&q->stack2);
  return x;
}
```

UPs and DOWNs

- Useful for lots of problems.
- Very easy to build one from an array.

- Built from an array makes a queue quite inneficient.
- In practice, not trivial to implement and maintain.

Your turn

Give an algorithm to convert a positive integer to its binary representation using a queue.

Complexity

1

enqueue / dequeue— **O**(1)

2

search — O (n)

(Worst case scenario - Avoid!)

Summary

- We implemented the basic operations for the ADTs linked-list, stack and queue.

- We reviewed the advantages of each one as their disadvantages.

- We found that these are optimal structures in terms of complexity as it takes only one step to insert or delete elements.

- Not particularly good structures for searching values.

THANKS!