Arrays

- Can’t make insertions efficiently at both ends
  - insertions in the middle of the array are also expensive
  - what is the cost?

- Can’t make deletions efficiently
  - what is the cost?

```plaintext
ptr
3 1 2 4 10 20 22 15 31 35
```

Linked Lists

- Collections of sequential elements stored at non-contiguous locations
- Nodes are connected by links
  - every node keeps a pointer to the next node
- Can grow and shrink dynamically
Singly Linked List

Head

<table>
<thead>
<tr>
<th>1</th>
<th>7</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>data</td>
<td>data</td>
<td>data</td>
</tr>
</tbody>
</table>

Tail

NULL pointer

Operations on Linked Lists

- Linked lists are just **collections** of sequential data
  - can **insert** 1 or more elements
    - front, end, by index, by value (sorted lists)
  - can **delete** 1 or more elements
    - front, end, by index, by value
  - can **search** for a specific element
  - can **get** an element at a given index
  - can **traverse** the list
    - visit all nodes and perform an operation (e.g. print or destroy)

Implementing a Singly Linked List

```cpp
class SLList {

private:
    SLNode *head;
    SLNode *tail;
    // all private data/methods
    // ...

public:
    SLList();
    ~SLList();
    // all public methods
    // ...
};
```
```cpp
class SLNode {
private:
    int data;
    SLNode *next;
    // all private data/methods
    // ...

public:
    SLNode(int d);
    ~SLNode();
    // all public methods
    // ...
    friend class SLLList;
};
```
Delete at front

Delete at end

Delete by value

Delete by index
Get

Destroy

Search

Circular Singly Linked List

1 7 3 5

Head

Tail
Doubly Linked List

Head

Tail

3 ↘️ 7 ↘️ 8 ↘️ 1 ↘️ NULL pointer

Circular Doubly Linked List

Head

Tail

3 ↘️ 7 ↘️ 8 ↘️ 1 ↘️ 3 ↘️ 7 ↘️ 8 ↘️ 1 ↘️