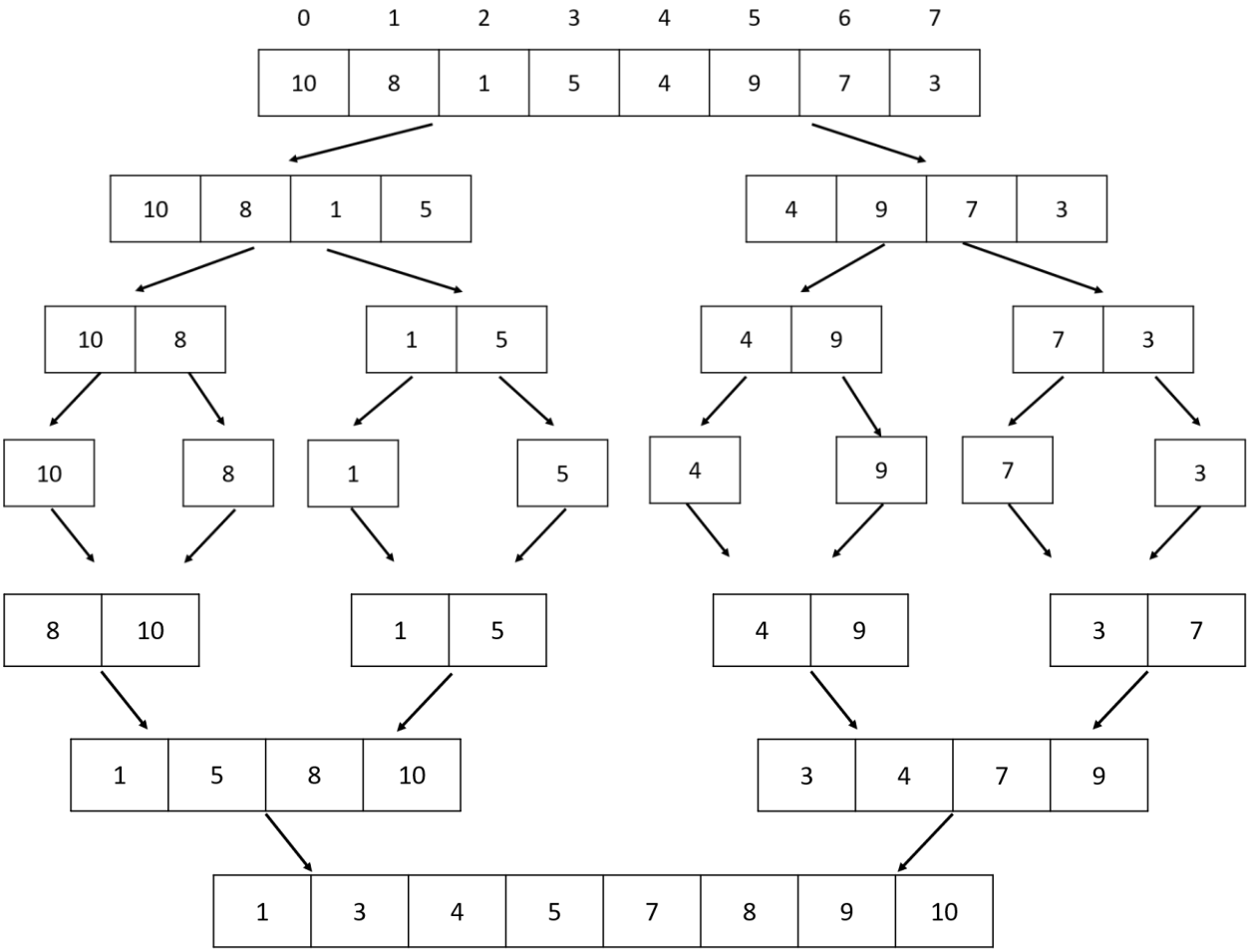
```
template <typename T>
class LinkedList {
public:
	void mergeSort();
private:
	void mergeSort(Node<T>*& head);
	void divideList(Node<T>* first1, Node<T>*& first2);
	Node<T>* mergeLists(Node<T>* a, Node<T>* b);
	Node<T>* first;
};
```

```
template <typename T>
void LinkedList<T>::divideList(Node<T>* first1,
                               Node<T>*& first2)
{
    Node<T>* middle;
    Node<T>* current;
    if (first1 == nullptr) {
        first2 = nullptr;
    }
    else if (first1->link == nullptr) {
        first2 = nullptr;
    } else {
        middle = first1;
        current = first1->link;
        if (current != nullptr) {
            current = current->link;
        }
        while (current != nullptr) {
            middle = middle->link;
            current = current->link;
            if (current != nullptr) {
                current = current->link;
            }
        }
        first2 = middle->link;
        middle->link = nullptr;
    }
}
```

```
template <typename T>
Node<T>* LinkedList<T>::mergeLists(Node<T>* head,
                                   Node<T>* otherHead)
{
    Node<T>* newHead = nullptr;
    Node<T>* current = nullptr;
    if (head->value < otherHead->value) {
        newHead = head;
        head = head->link;
    }
    else {
        newHead = otherHead;
        otherHead = otherHead->link;
    }
    current = newHead;
```

```
    while ((head != nullptr) && (otherHead != nullptr)) {
        if (head->value < otherHead->value) {
            current->link = head;
            current = head;
            head = head->link;
            if (head == nullptr) {
                current->link = otherHead;
            }
        }
        else {
            current->link = otherHead;
            current = otherHead;
            otherHead = otherHead->link;
            if (otherHead == nullptr) {
                current->link = head;
            }
        }
    }
    return newHead;
}
```

```
template <typename T>
void LinkedList<T>::mergeSort(Node<T>*& head)
{
	Node<T>* otherHead;
	if (head != nullptr) {
		if (head->link != nullptr) {
			divideList(head, otherHead);
			mergeSort(head);
			mergeSort(otherHead);
			head = mergeLists(head, otherHead);
		}
	}
}
```


0 1 2 3 4 5 6 7
10 8 1 5 4 9 7 3
10 8 1 5
4 9 7 3
10 8
1 5
4 9
7 3
10
8
1
5
4
9
7
3
8 10
1 5
4 9
3 7
1 5 8 10
3 4 7 9
1 3 4 5 7 8 9 10