```
template <class T>
Node<T>* LinkedList<T>::MergeSort(Node<T>*hea)
{
  mergeSorting(&hea);
  return hea;
}
```

Costo	# veces o $\partial(?)$
c_1	$\partial(n \log_2 n)$
c_2	1

$T(n) = c_1(n \log_2 n) + c_2(1)$

$T(n) = \partial(n \log_2 n)$

```
template <class T>
Node<T>* LinkedList<T>::MergeSort(Node<T>*hea)
{
  mergeSorting(&hea);
  return hea;
}
```

Costo	# veces o $\partial(?)$
c_1	$\partial(n \log_2 n)$
c_2	1

$$T(n) = c_1(n \log_2 n) + c_2(1)$$

$$T(n) = \partial(n \log_2 n)$$

```
template <class T>
void LinkedList<T>::mergeSorting(Node<T>** hea){
    Node<T>* cur = *hea;
    Node<T>* first;
    Node<T>* second;
    if(!cur or !cur->getNext()) return;
    findMiddle(cur, &first, &second); → O(n)
    mergeSorting(&first);
    mergeSorting(&second);
    *hea = mergeBoth(first, second);
```

Costo	# veces o O(?)
c_1	1
	1
c_2	
c_3	1
c_4	1
c_5	n
c_6	$T(n/2)$
c_7	$T(n/2)$
c_8	n

$$T(n) = \begin{cases} 1, & n=1 \\ 2T(n/2) + n, & n>1 \end{cases}$$

$= 2T(n/2) + n$

$2[2T(n/2^2) + n/2] + n$

$= 2^2 T(n/2^2) + 2n$

$= 2^3 T(n/2^3) + 3n$

$= 2^k T(n/2^k) + kn$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \quad k = \log_2 n$

$2^{\log_2 n} T(n/2^{\log_2 n}) + \log_2 n \cdot n$

$n T(1) + n \log_2 n$

$T(n) = n + n\log_2 n$

$T(n) = O(n \log_2 n)$

```
template <class T>
void LinkedList<T>::findMiddle(Node<T>* cur, Node<T>** first, Node<T>** second)
{
  Node<T>* fast;
  Node<T>* slow;
  slow = cur;
  fast = cur->getNext();

  while(fast != nullptr) -> fast getNext +2 until end
  {
    fast = fast->getNext();
    if(fast != nullptr)
    {
      slow = slow->getNext();
      fast = fast->getNext();
    }
  }
  *first = cur;
  *second = slow->getNext();
  slow->setNext(nullptr);
}
```

Best case = worst case

$O(n)$

costo	# veces $O(?)$
c_1	1
c_2	1
c_3	1
c_4	1
c_5	n
c_6	n
c_7	n
c_8	n
c_9	n
c_{10}	1
c_{11}	1
c_{12}	1

$O(n)$

$T(n) = c_1(1) + c_2(1) + c_3(1) + c_4(1) + c_5(n) + c_6(n) + c_7(n) + c_8(n) + c_9(n) + c_{10}(1) + c_{11}(1) + c_{12}(1)$

$= c_1 + c_2 + c_3 + c_4 + c_5 n + c_6 n + c_7 n + c_8 n + c_9 n + c_{10} + c_{11} + c_{12}$

$T(n) = n(c_5 + c_6 + c_7 + c_8 + c_9) + 1(c_1 + c_2 + c_3 + c_4 + c_{10} + c_{11} + c_{12})$

$T(n) = O(n)$

```
template <class T>
Node<T>* LinkedList :: mergeBoth (Node<T>* first, Node<T>* second)
{
  Node<T>* answer = nullptr;
  if(! first)
  {
    return second;
  }
  else if(! second)
  {
    return first;
  }

  if(first->getIpVal() <= second->getIpVal())
  {
    answer = first;
    answer->setNext(mergeBoth(first, second->getNext()));
  }
  else
  {
    answer = second;
    answer->setNext(mergeBoth(first->getNext(), second->getNext()));
  }

  return answer;
}
```

costo	#veces o $\delta(?)$
	n
C_1	1
C_2	n
C_3	1
C_4	1
C_5	1
C_6	1
C_7	1
C_8	$T(n-1)$
C_9	1
C_{10}	$T(n-1)$
C_{11}	1

$$T(n) = C_1(1) + C_2(n) + C_3(n) + C_4(n) + C_5(1) + C_6(1) + C_7(1) + C_8(T(n-1)) + C_9(T(n-1)) + C_{10}(T(n)) + C_{11}(1)$$

$$T(n) = \begin{cases} 1, & n = 1 \\ 2T(n-1) + 1, & n > 1 \end{cases}$$

$$T(n) = 2T(n-1)$$
$$= 2[2T(n-1)]$$
$$= 2^2 T(n-2)$$
$$= 2^3 T(n-3)$$
$$= 2^k T(n-k)$$

$$2^k T(n-k)$$
$$2^k T(n+k)$$
$$n+k=1$$
$$1+k=1$$
$$k=0$$

$$T(n) = \mathcal{O}(n)$$