Sorting (V): Merge Sort

CSCD 300 – Data Structures

Eastern Washington University

EASTERN WASHINGTON UNIVERSITY
start something big

Goal

We will learn the mechanism of the Merge Sort algorithm and then analyze its time complexity in the best as well as in the worst case.

Outline

EASTERN WASHINGTON UNIVERSITY
start something big

Merge sort

Basic idea

- Divide the n-number sequence into two halves. Each subsequence has $n/2$ numbers.
- Recursively sort each n/2-number subsequence.
- Merge two sorted subsequences into one sequence.

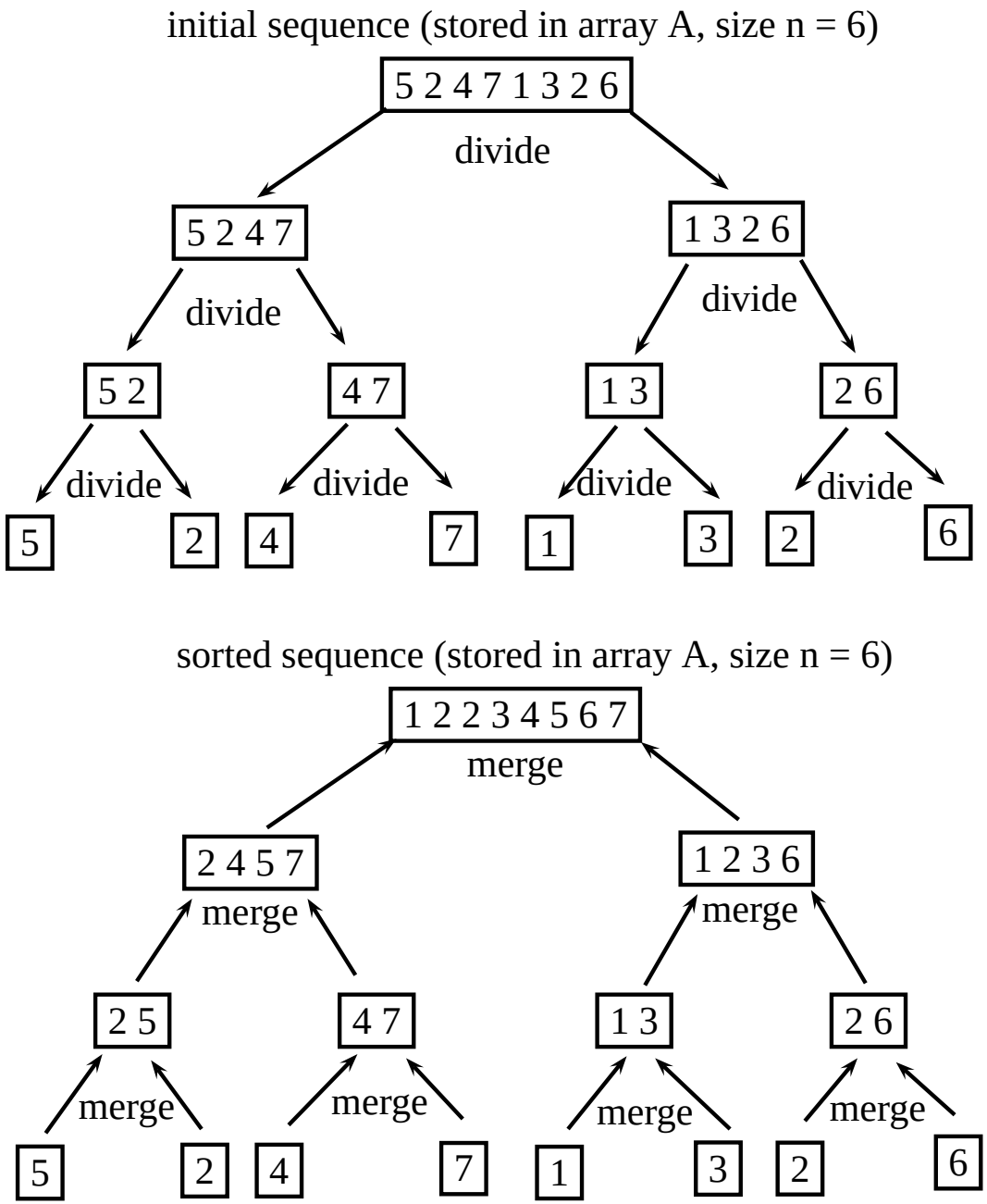
Psuedocode

```
MERGE_SORT(A, p, r)
{
   if(p<r)
      q = floor((p+r)/2)

      MERGE_SORT(A, p, q)
      MERGE_SORT(A, q+1, r)

      MERGE(A, p, q, r)
}
```

```
MERGE(A, p, q, r)
{
   copy A[p ... q] into L[0 ... q-p]
   copy A[q+1 ... r] into R[0 ... r-q-1]

   i = 0; j = 0;
   for k = p to r
      if(i == q-p+1)
         A[k] = R[j]; j ++;
      else if(j == r-q)
         A[k] = L[i]; i ++;
      else if L[i] <= R[j]
         A[k] = L[i]; i ++;
      else
         A[k] = R[j]; j ++;
}
```

Call "MERGE_SORT(A, 0, n-1)" will sort the sequence of n numbers that are stored in array A. Let's look at an example ...

An example


initial sequence (stored in array A, size n = 6)
5 2 4 7 1 3 2 6
divide
5 2 4 7
1 3 2 6
divide
divide
5 2
4 7
1 3
2 6
divide
divide
divide
divide
5
2
4
7
1
3
2
6
sorted sequence (stored in array A, size n = 6)
1 2 2 3 4 5 6 7
merge
2 4 5 7
1 2 3 6
merge
merge
2 5
4 7
1 3
2 6
merge
merge
merge
merge
5
2
4
7
1
3
2
6

Time complexity of Merge Sort: recurrence equation

```
MERGE_SORT(A, p, r)
{
   if(p<r)
      q = floor((p+r)/2)

      MERGE_SORT(A, p, q)
      MERGE_SORT(A, q+1, r)

      MERGE(A, p, q, r)
}
```

Initial call: `Merge_Sort(A,0,n-1)`

Let $T(n)$ =time cost for sorting the n numbers in $A[0...n-1]$, then we have:

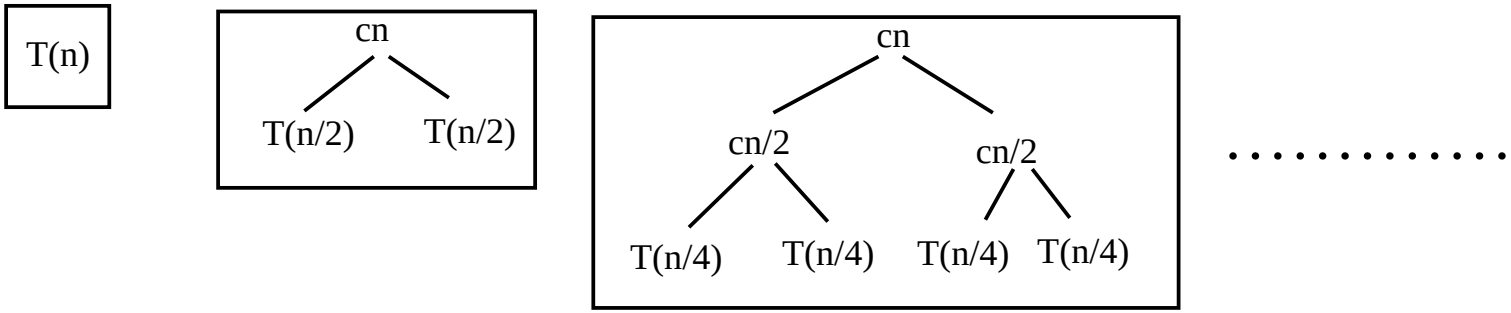
$$T(n) = 2T(n/2) + O(n)$$

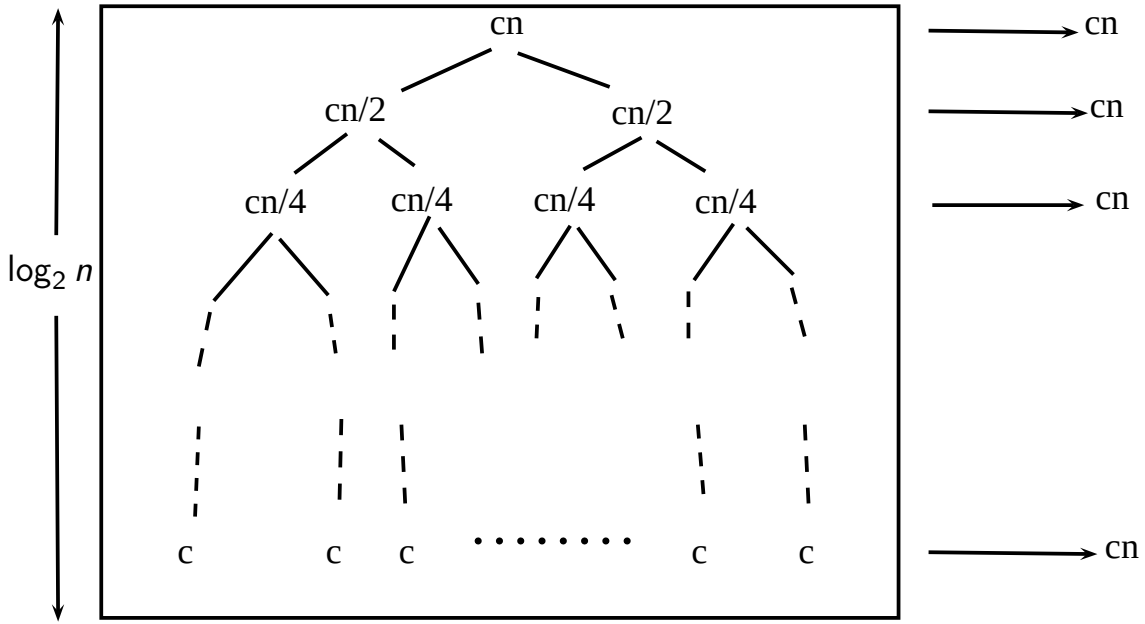
because

- sorting A[0...n-1] is divided into sorting of two subsequences. Sorting each subsequence takes time $T(n/2)$ as each subsequence has size $n/2$.
- merging the two sorted subsequences of each sized $n/2$ takes n constant-time operations.

Solve the recurrence

Note: c is some positive constant.





Total: $T(n) = cn \log_2 n = O(n \log n)$

Question

How do you use the Merge sort if the data sequence is saved in a singly linked list ?