

Quick Sort and Merge Sort

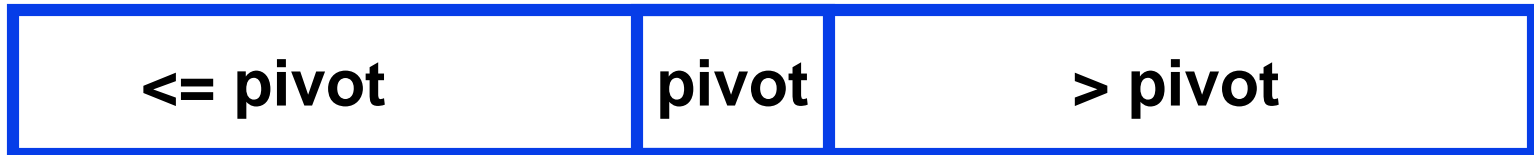
Chan Hou Pong, Ken
CSCI2100A Data Structures

Quick Sort

- Efficient sorting algorithm
- Example of **Divide and Conquer** algorithm
- Two phases
 - Partition phase
 - **Divides** the work into half
 - Sort phase
 - **Conquers** the halves!

Quicksort

- Partition
 - Choose a **pivot**
 - Find the position for the pivot so that
 - all elements to the left are less/equal
 - all elements to the right are greater



Quicksort

- Conquer
 - Apply the same algorithm to each half





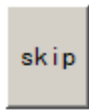
stop



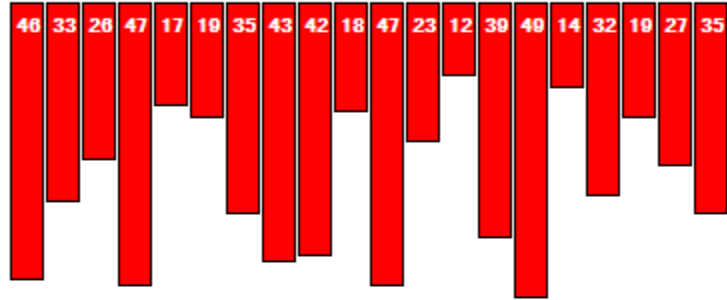
run



step



skip



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition



stop



run



step



skip



Pivot set to 46...

LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口



stop



run



step



skip



Smaller than pivot -> add to left partition...

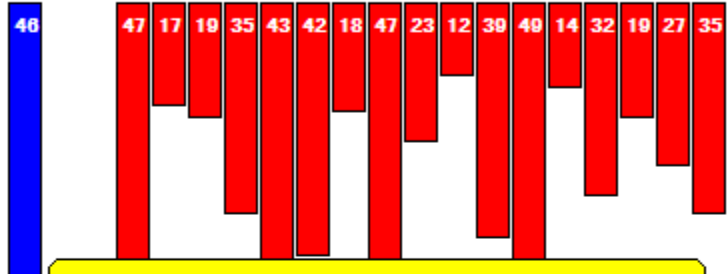
Comparing each entry in the partition with the pivot...

LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition



Click NEXT STEP...

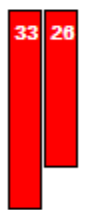


LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Smaller than pivot -> add to left partition...

Comparing each entry in the partition with the pivot...

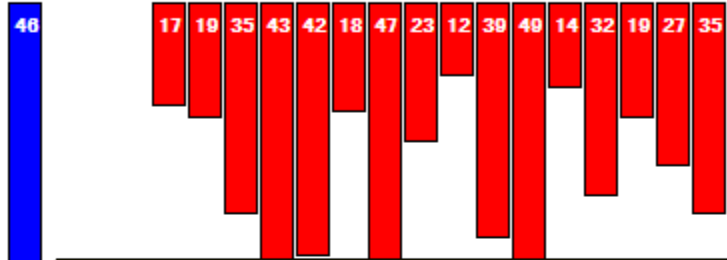


Click NEXT STEP...



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

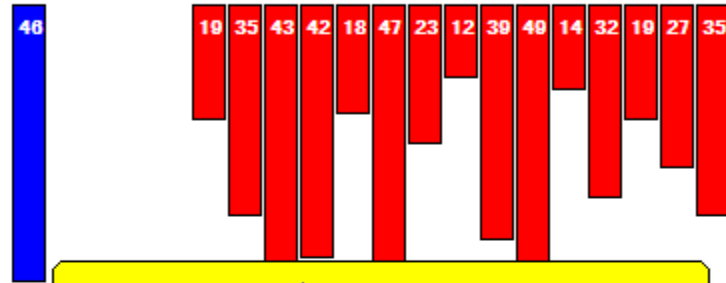
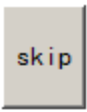


Greater than (or equal to) pivot -> add to right partition...

Comparing each entry in the partition with the pivot...



Click NEXT STEP...



LEGEND

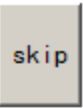
- Pivot
- Sorted
- Inactive Partition
- Active Partition

Smaller than pivot -> add to left partition...

Comparing each entry in the partition with the pivot...



Click NEXT STEP...



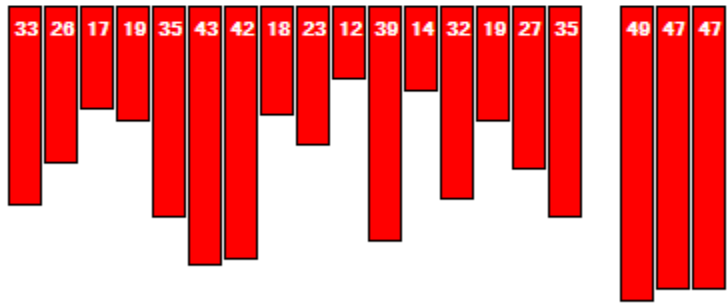
LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition



Smaller than pivot -> add to left partition...

Comparing each entry in the partition with the pivot...



Click NEXT STEP...



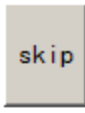
stop



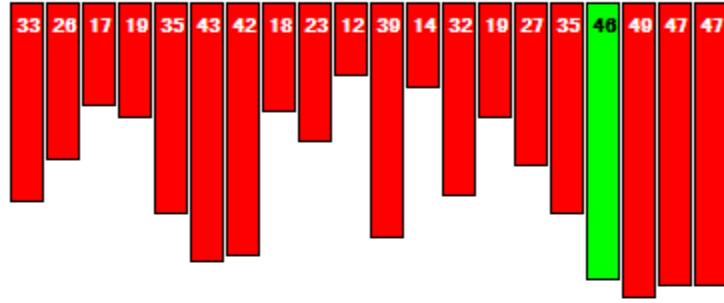
run



step



skip

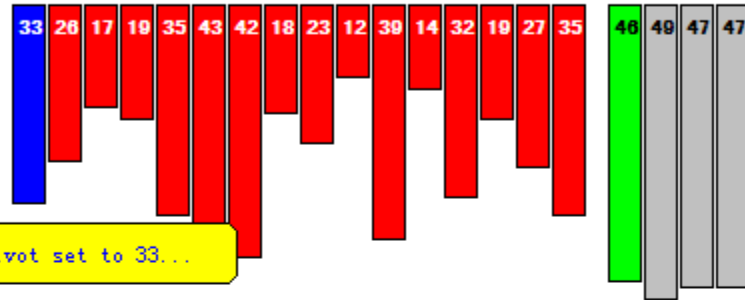


LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...



stop



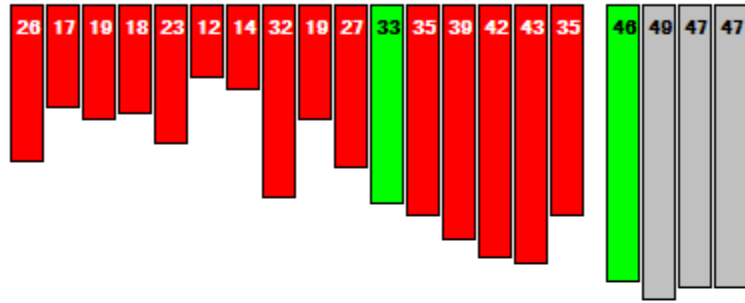
run



step



skip



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...



stop



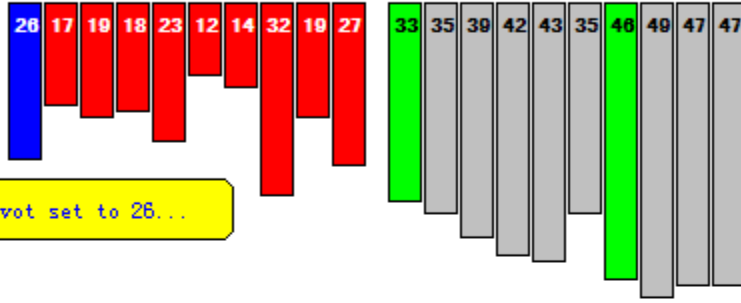
run



step



skip



Pivot set to 26...

LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口



stop



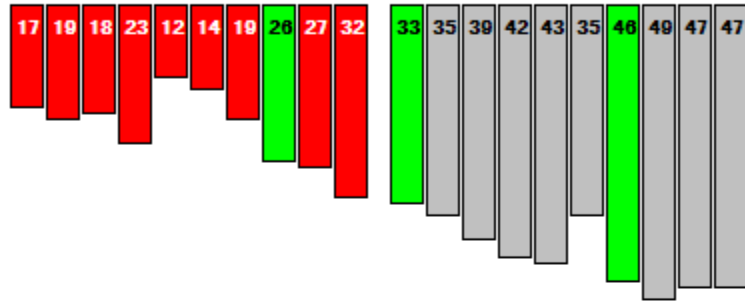
run



step



skip



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口



stop



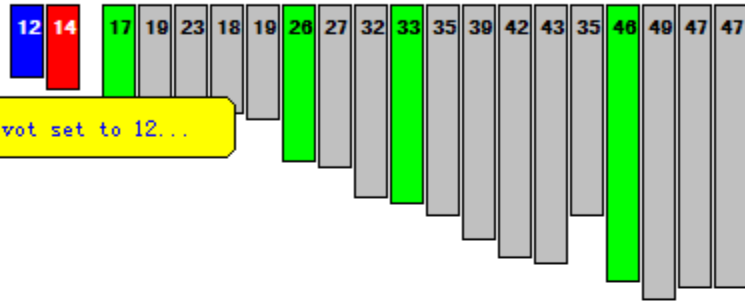
run



step



skip

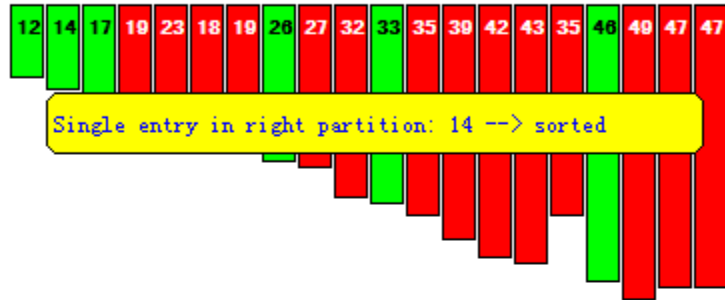


LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口

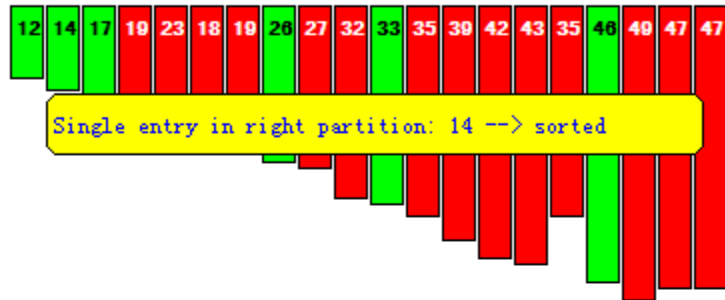


LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...

警告: Applet 窗口



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...



stop



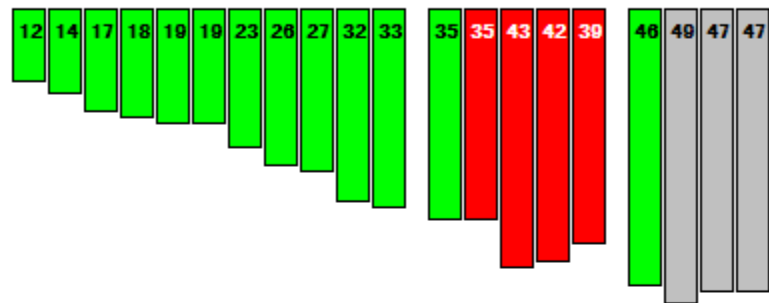
run



step



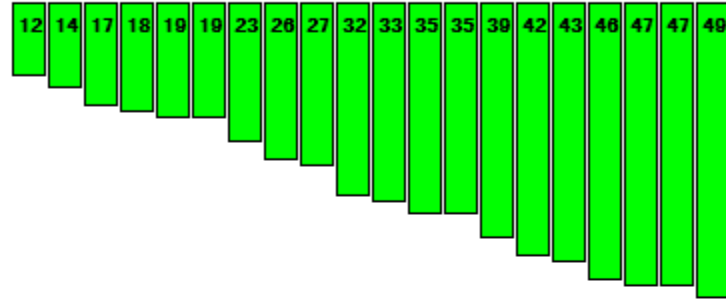
skip



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Click NEXT STEP...



LEGEND

- Pivot
- Sorted
- Inactive Partition
- Active Partition

Animation completed successfully. Press 'run' to restart.

Quicksort

- Implementation

```
quicksort( void *a, int low, int high )
{
  int pivot;
  /* Termination condition! */
  if ( high > low )
  {
    pivot = partition( a, low, high );
    quicksort( a, low, pivot-1 );
    quicksort( a, pivot+1, high );
  }
}
```

Divide

Conquer

Quicksort - Partition

```
int partition( int *a, int low, int high ) {
    int left, right;
    int pivot_item;
    pivot_item = a[low];
    pivot = left = low;
    right = high;
    while ( left < right ) {
        /* Move left while item < pivot */
        while( a[left] <= pivot_item && left < right ) left++;
        /* Move right while item > pivot */
        while( a[right] > pivot_item) right--;
        if ( left < right ) {
            SWAP(a, left, right);
        }
    }
    /* right is final position for the pivot */
    a[low] = a[right];
    a[right] = pivot_item;
    return right;
}
```

Quicksort - Analysis

- Non stable sorting algorithm
- Average case time complexity: $O(n \log n)$
- Worst case time complexity: $O(n^2)$
 - Better pivot selection reduces probability
- Use when you need good performance on average

Questions in previous exam

- 7, 6, 1, 4, 8, 2, 3, 5
- Demonstrate ONLY the process of partition the sequence of the QuickSort, with a cutoff of 3
- Choose pivot as the **median** of the **first**, **middle**, and **last** element of the array
- Please swap the selected pivot with the first element
- Suppose: Size of array is n , middle element index is $n/2$. Index starts from 0 to $n-1$

Merge Sort - Definition

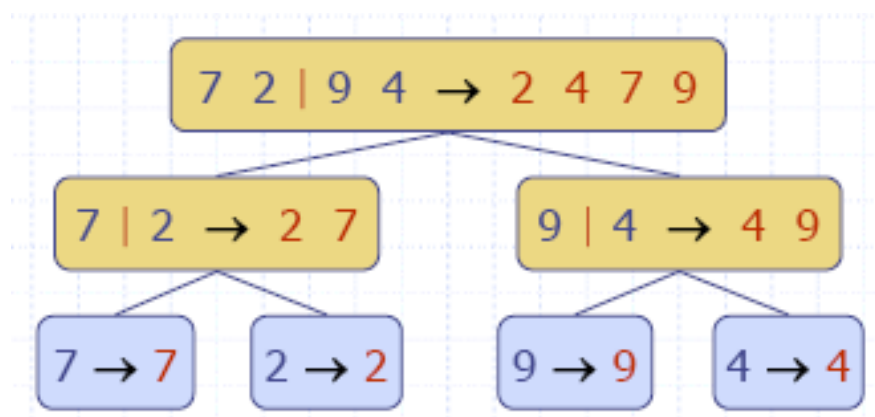
- **Definition:**
- A *sort* algorithm that
- Splits the items to be sorted into **two** groups
- *Recursively* sorts each group
- *Merge* them into a final sorted sequence.

Merge Sort – Divide-and-Conquer

- **Divide-and-conquer** is a general algorithm design paradigm:
 - **Divide**: divide the input data S in two disjoint subsets S_1 and S_2
 - **Conquer**: solve the sub-problems associated with S_1 and S_2 recursively
 - **Combine**: combine the solutions for S_1 and S_2 into a solution for S
- The base case for the recursion are sub-problems of size 0 or 1

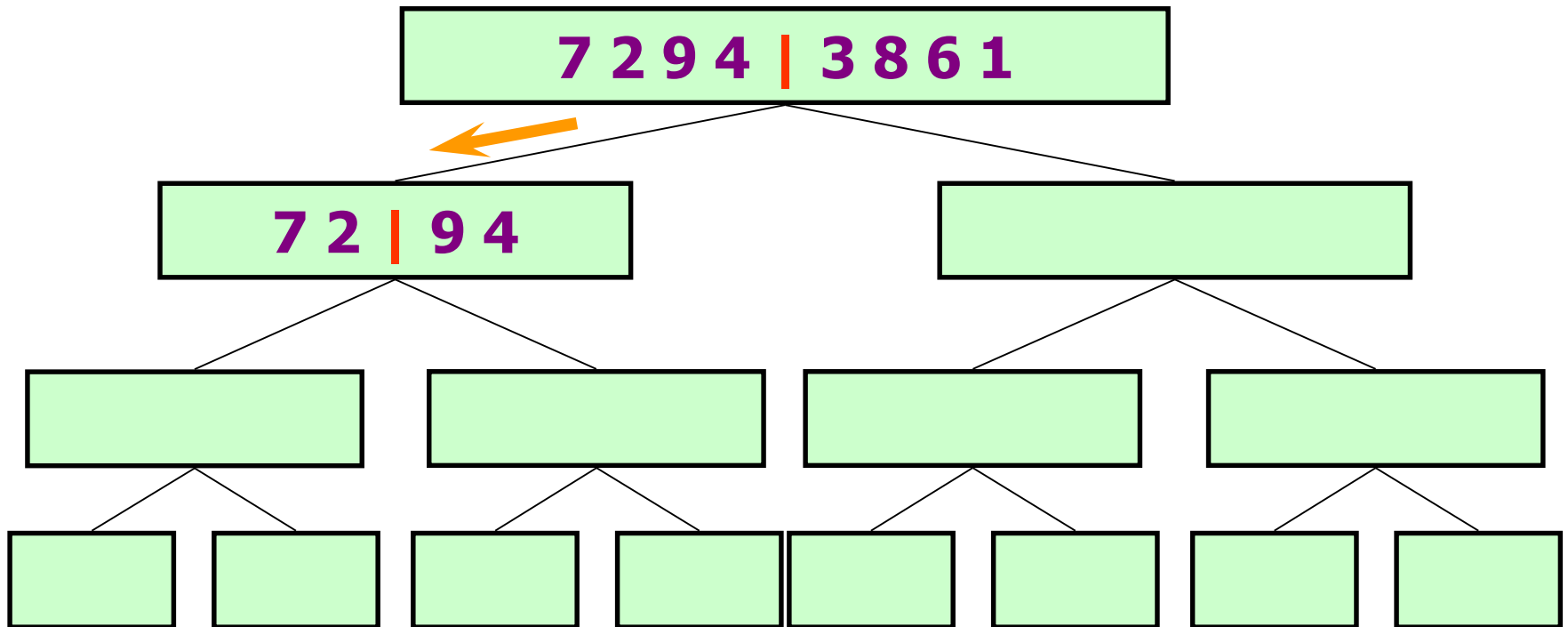
Merge Sort Tree

- An execution of merge-sort is depicted by a binary tree
 - each node represents a recursive call of merge-sort, and it stores
 - unsorted sequence before the execution
 - sorted sequence at the end of the execution
 - the root is the **initial** call
 - the leaves are calls on subsequences of size 0 or 1



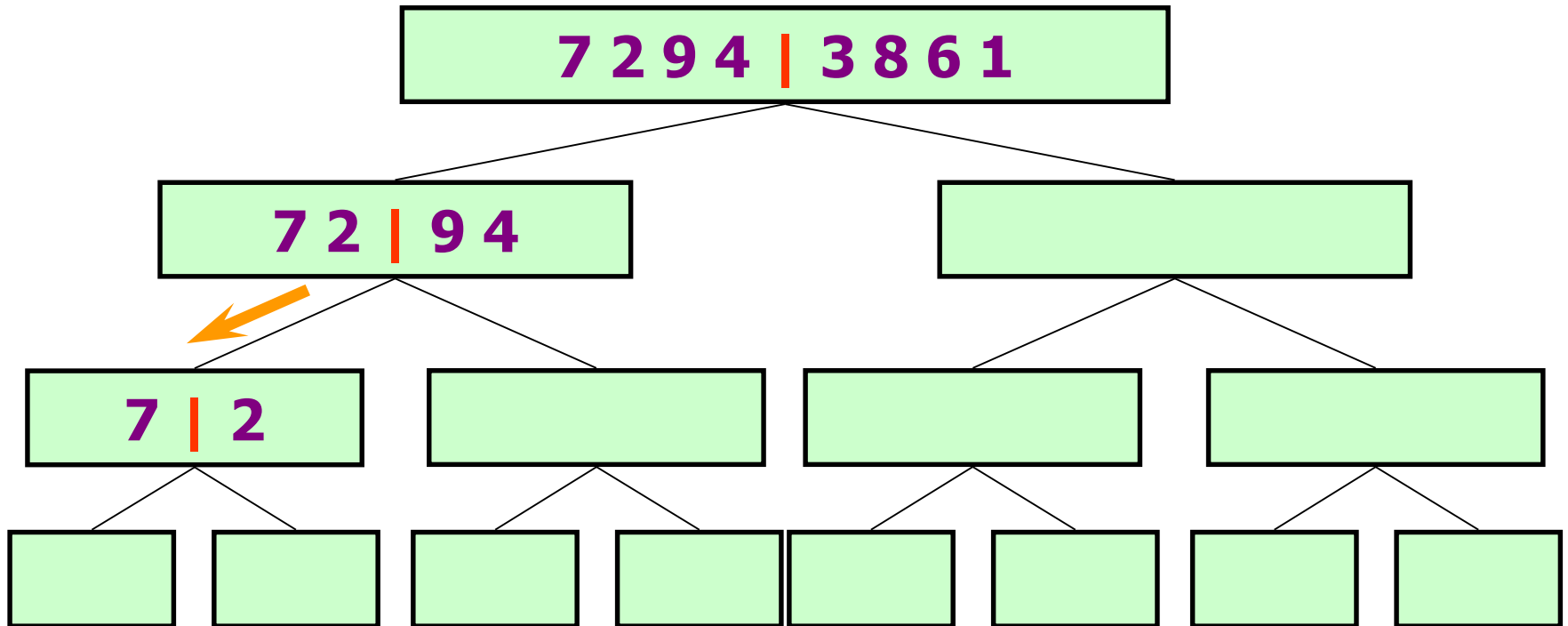
Merge Sort - example

Recursive call, partition



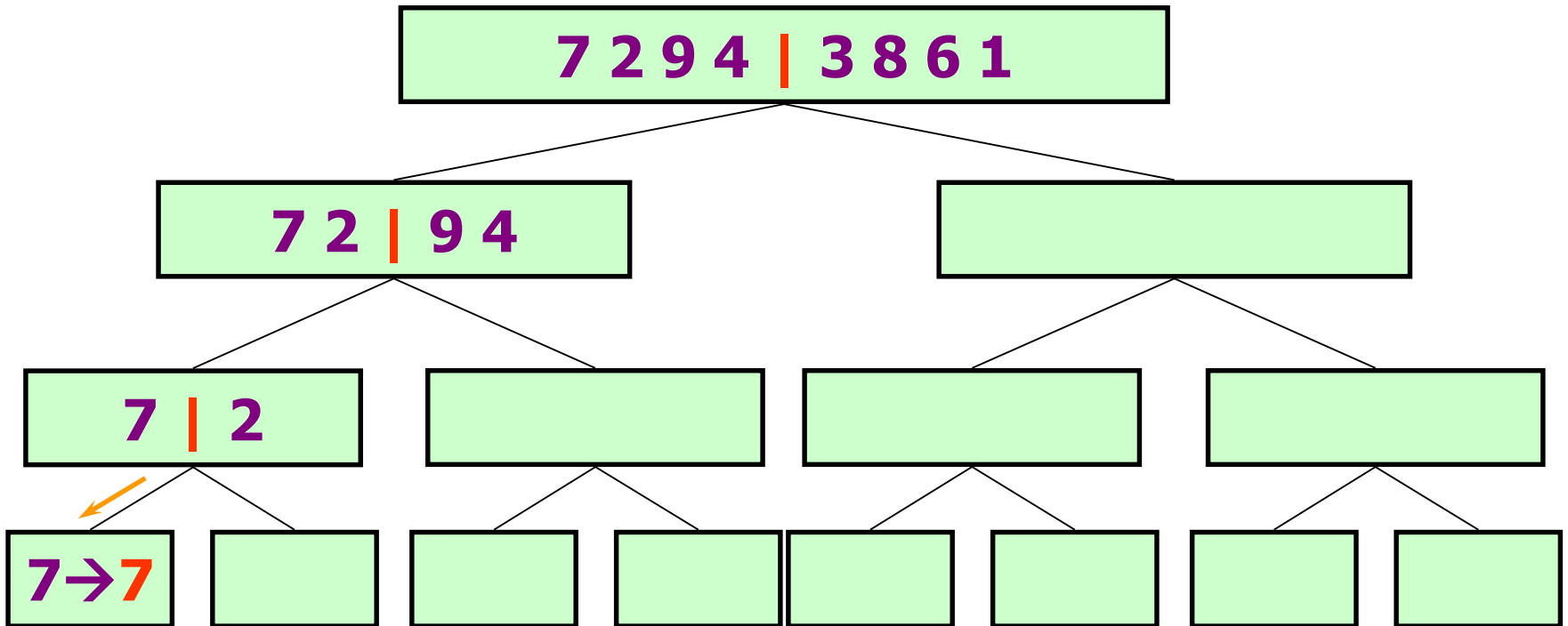
Merge Sort - example

Recursive call, partition



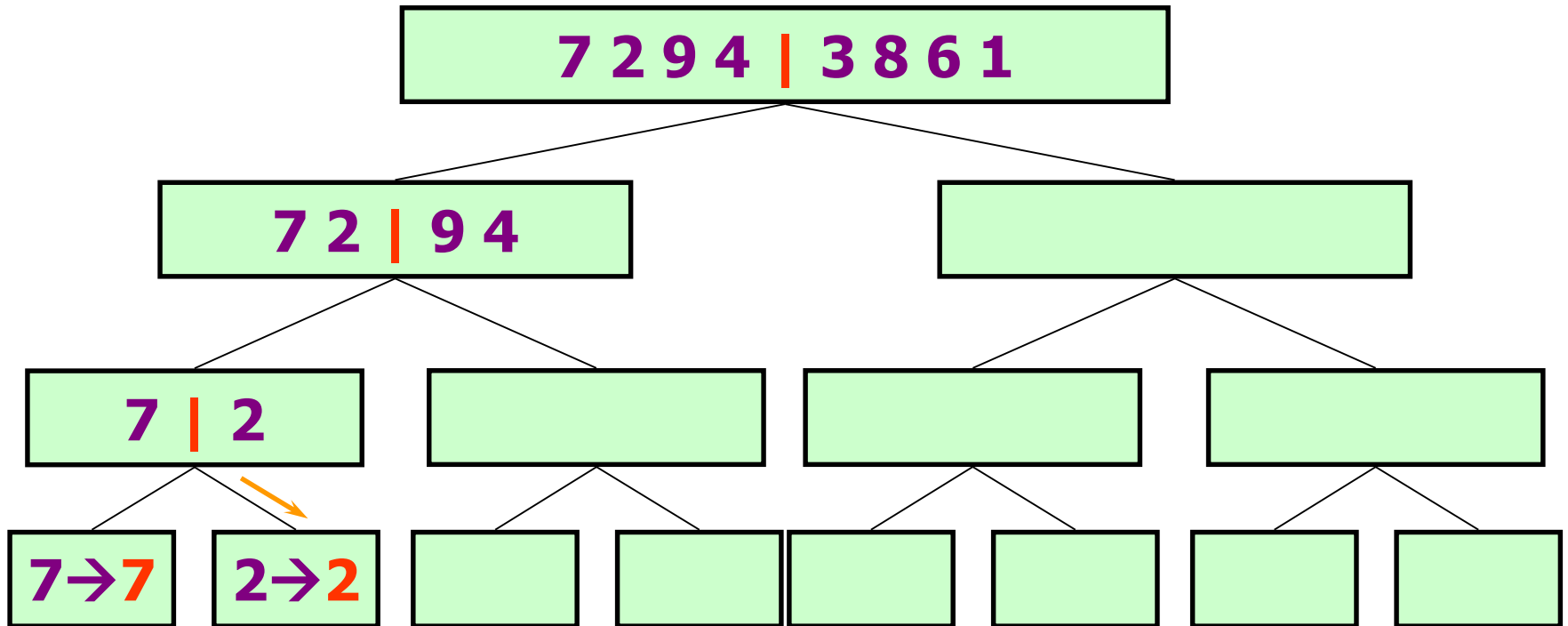
Merge Sort - example

Recursive call, base case



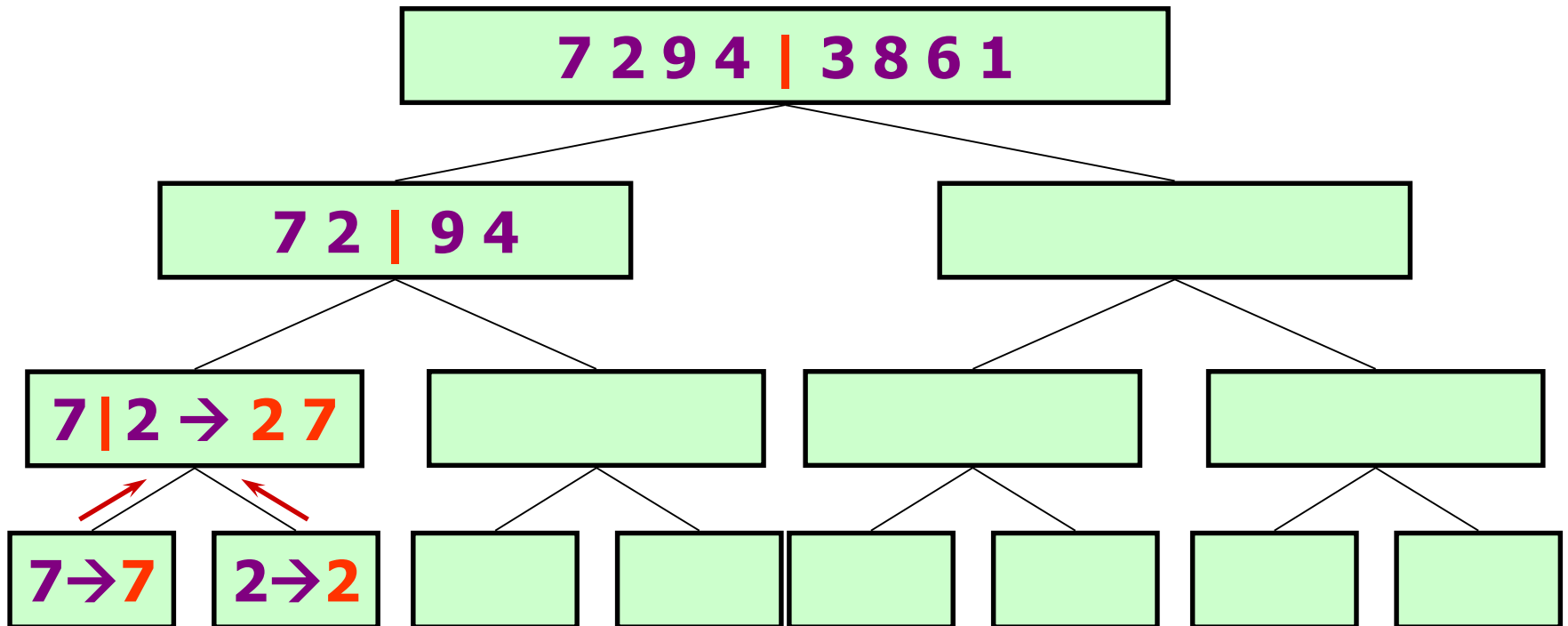
Merge Sort - example

Recursive call, base case



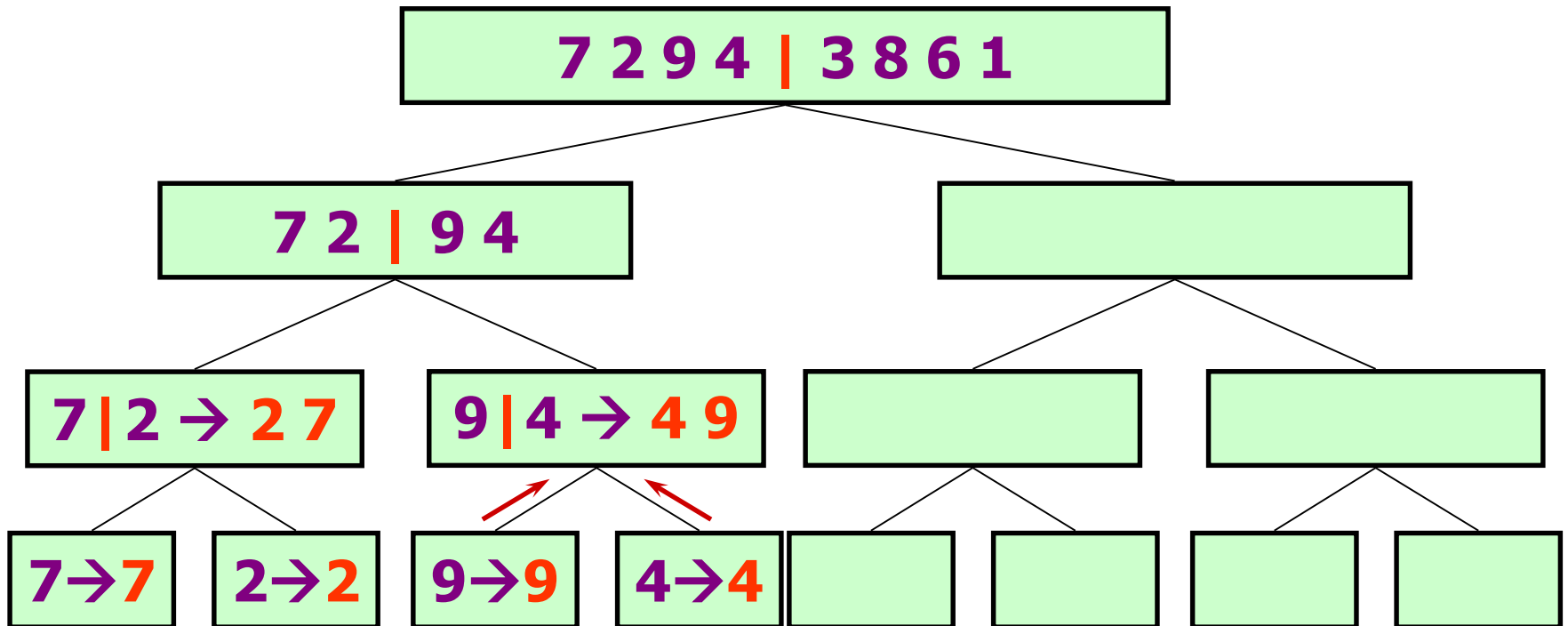
Merge Sort - example

Merge



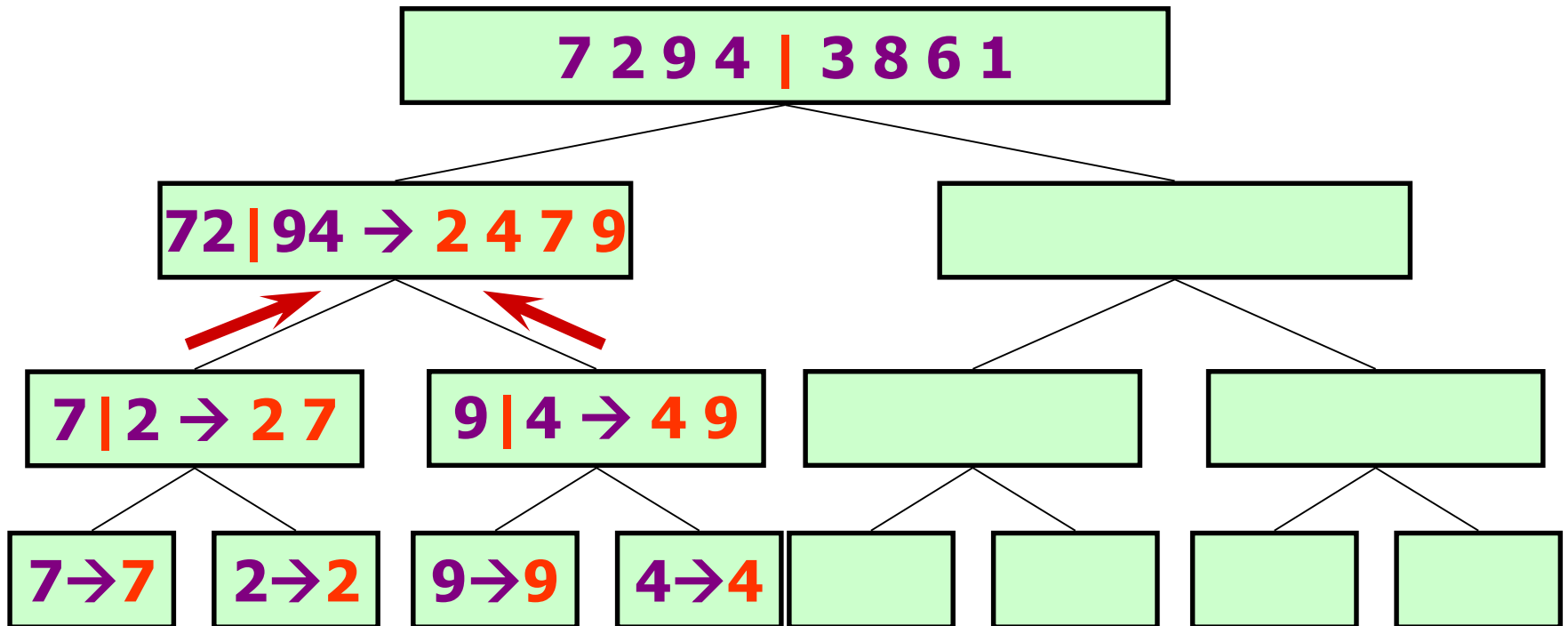
Merge Sort - example

Recursive call, base case, ..., base case, merge



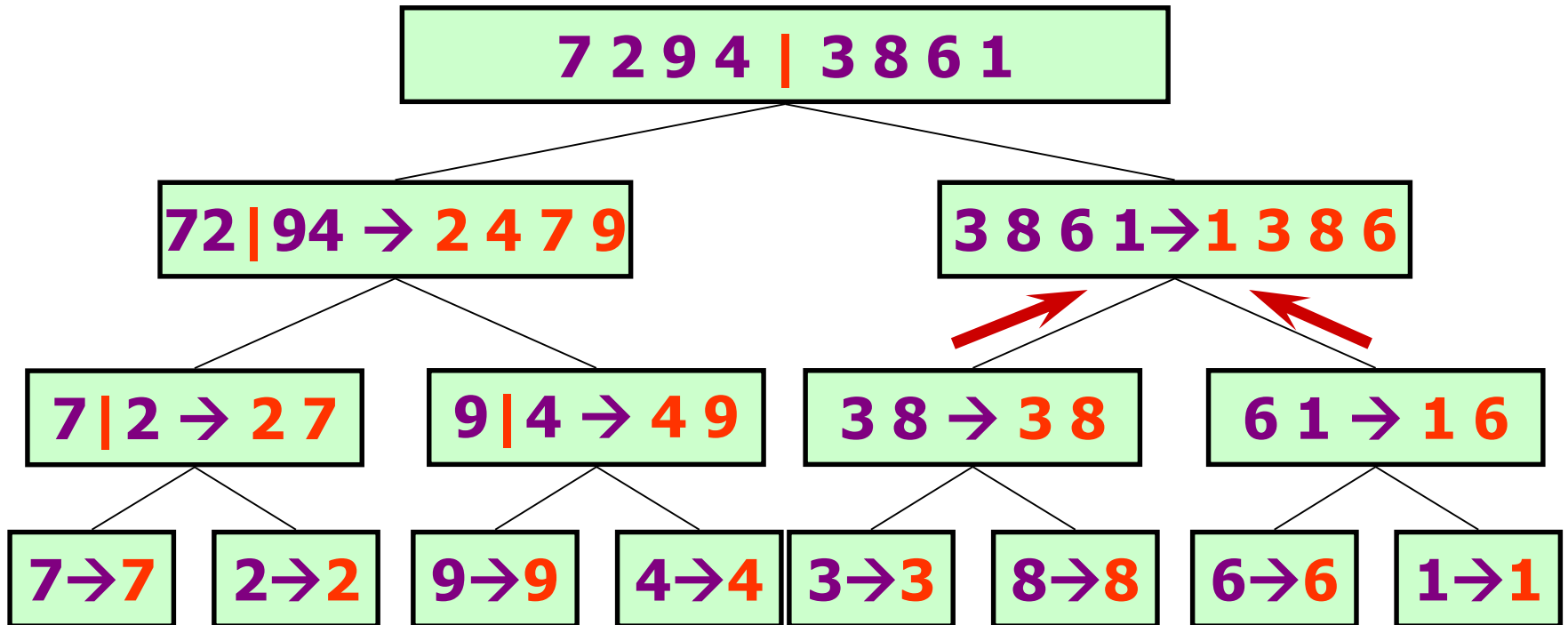
Merge Sort - example

Merge



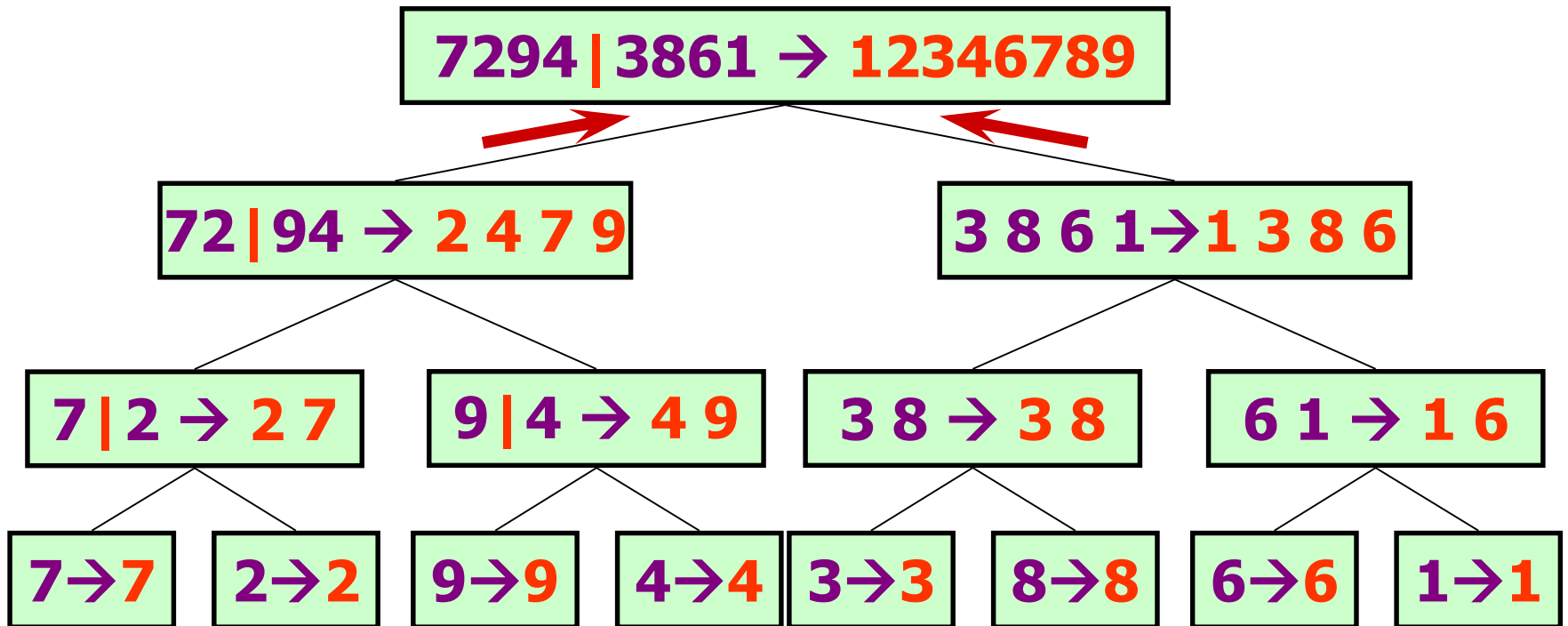
Merge Sort - example

Recursive call, ..., merge, merge



Merge Sort - example

Merge



Merge Sort - analysis

- A stable sorting algorithm
- Worst case run time: $O(n \log n)$.
- Need $O(n)$ extra space

Merge Sort – sample code

```
void mergesort(int low, int high)
{
    if (low<high)
    {
        int middle=(low+high)/2;
        mergesort(low, middle);
        mergesort(middle+1, high);
        merge(low, middle, high);
    }
}
```


Merge Sort – sample code (cont.)

```
void merge(int low, int middle, int high)
{
    int i, j, k;
    // copy both halves of a to auxiliary array b
    for (i=low; i<=high; i++)
        b[i]=a[i];
    i=low; j=middle+1; k=low;
    // copy back next-greatest element at each time
    while (i<=middle && j<=high)
        if (b[i]<=b[j])    a[k++]=b[i++];
        else                a[k++]=b[j++];
    // copy back remaining elements of first half (if any)
    while (i<=middle)
        a[k++]=b[i++];
}
```

Questions in previous exam

- Given the sequence (5,2,8,4,1,6,3,7), sort the sequence using Mergesort.
- Illustrate the result after each pass.
- How many comparisons have you performed?

Questions in previous exam

- Show all the inversion pairs in the list $(4,1,5,2,3)$?
- List them out in an organized manner
- i.e., sort them in an ascending order by the first element and then by the second element

Questions in previous exam

- Given an array with n elements to sort,
- Write down the **best** and **worst time complexity** when using **Bubble sort**, **Selection sort**, **Insertion sort** and **Merge sort** in terms of Big-O notation