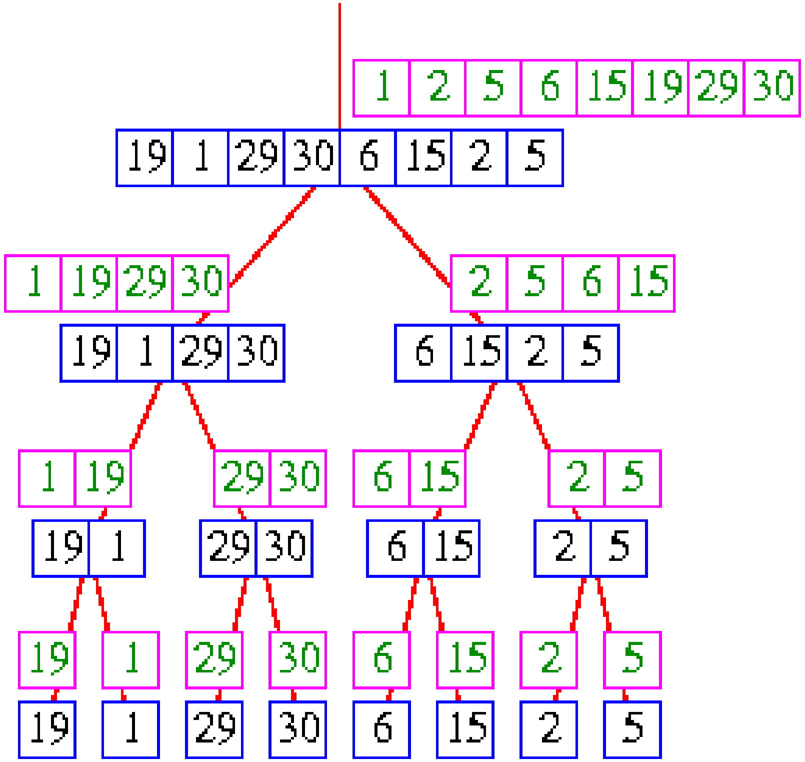
Merge Sort

A Divide and Conquer Method

Merge Sort Method

- Makes use of the divide and conquer method
- Each set is individually sorted and the resulting sorted sequences are merged to produce a single sorted sequence of n elements



Merge Sort Algorithm

```
ALGORITHM Mergesort(C[0..n-1])
{
  if n > 1
  { //divide
      copy C[0.. n/2  – 1] to A[0.. n/2  – 1]
      copy C[ n/2 .. n – 1] to B[0.. n/2  –
  1]
      //conquer
      Mergesort(A[0.. n/2  – 1)
      Mergesort(B[0.. n/2  – 1)
      Merge(A, B, C)  //combine
  }
}
```

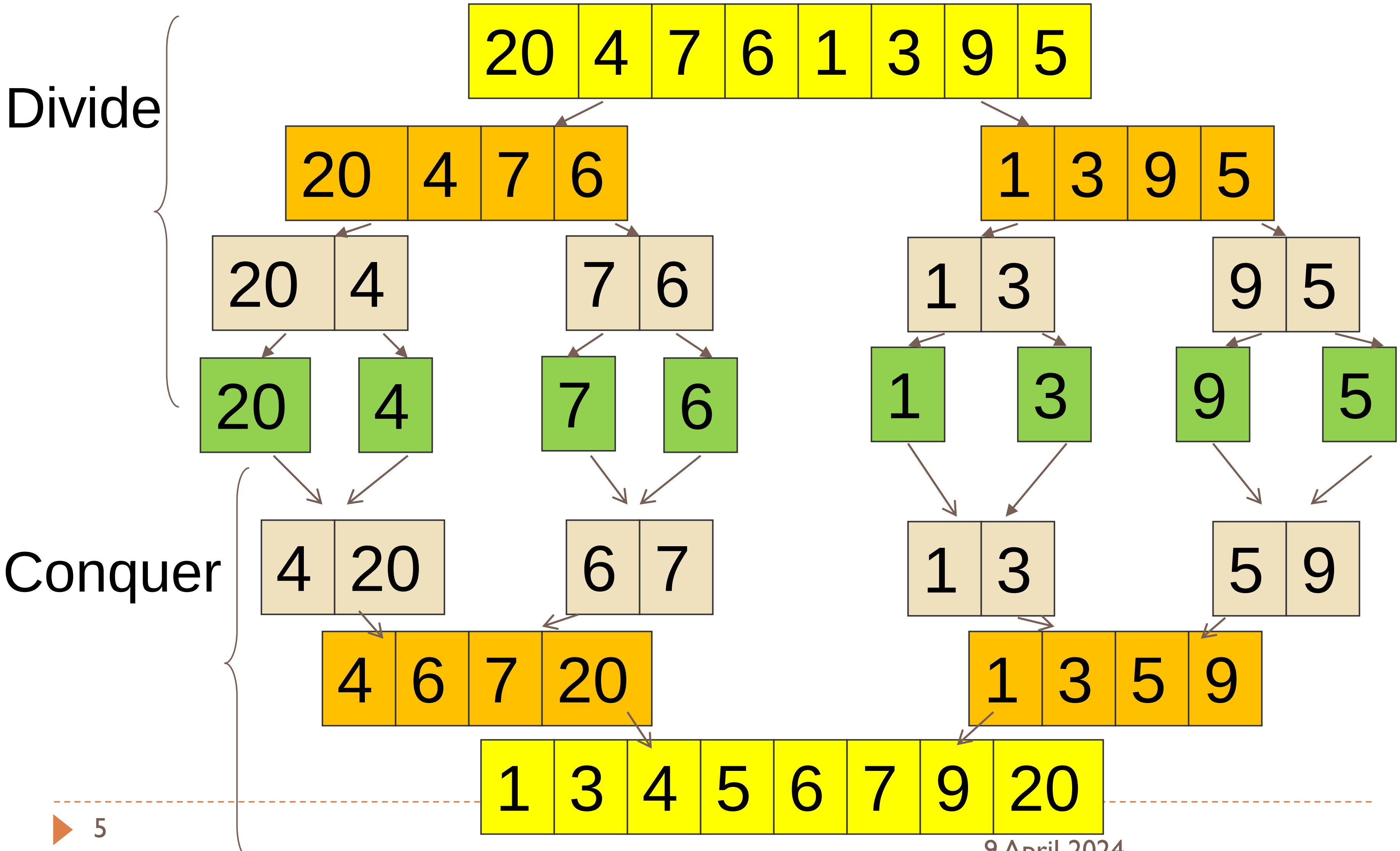
Merge Algorithm

```
ALGORITHM Merge(A,B,C)
{
    n ← size of array A;   m ← size of array B
    i ← 1; j ← 1,k ←1;
    while((i<=n)&&(j<=m))
    {
        if Ai < Bj
                Ck ← Ai
                i ← i +1; k ← k +1

        else
                Ck ← Bj
                j← j+1;  k ← k +1

    }
    for(;i<=n)
    {
    Ck ← Ai
    k ← k +1
    }
    for(;j<=m)
    {
    Ck ← Bj
    k← k+1
    }
    return C
}
```

Merge sort: Example



Merge Sort Analysis: Combine Step

- 2 arrays of size 1 can be easily merged to form a sorted array of size 2
- 2 sorted arrays of size *n and m* can be merged in $O(n+m)$ time to form a sorted array of size $n+m$

Analysis Of Merge Sort

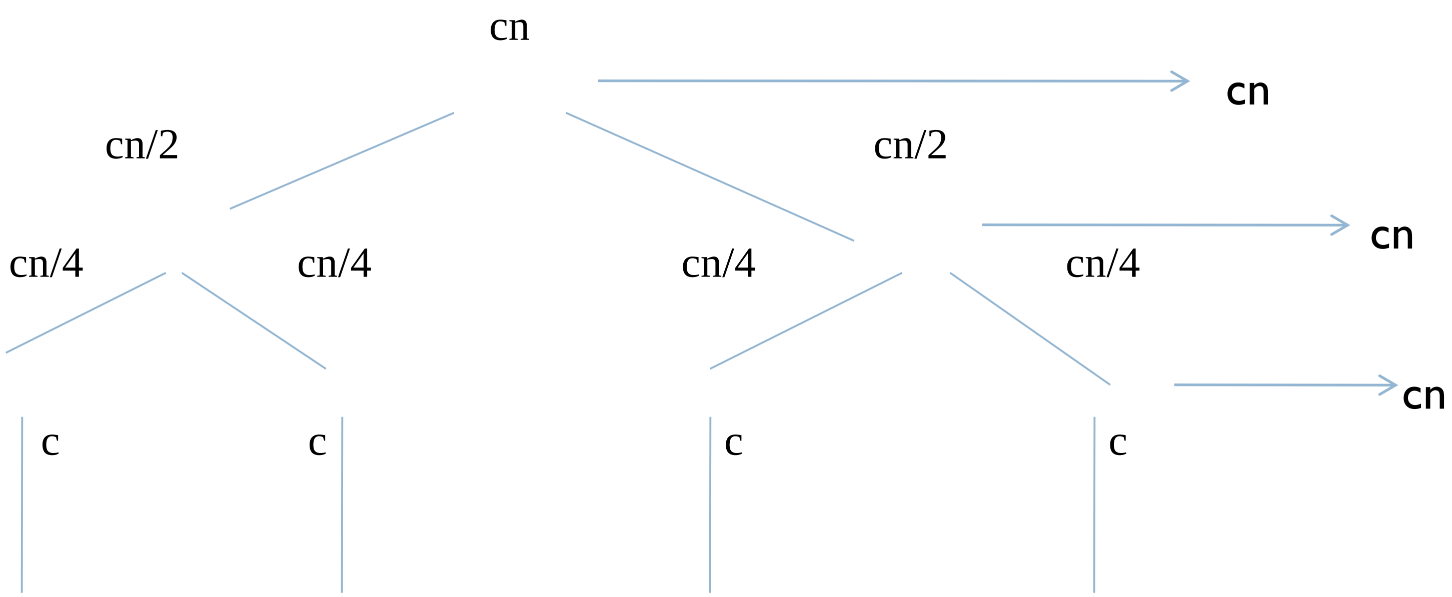
- Divide – In divide step we compute the middle of the array which takes constant time .
- Conquer- In conquer step , we recursively solve the subproblems , each of size n/2 which contributes to 2A(n/2) to the running time
- Combine – In combine step we just combine the elements in the sorted order using Merge procedure which takes $\theta(n)$ time and thus $C(n)= \theta(n)$

Complexity Analysis –Merge Sort

- The recurrence for merge sort can be written as
  - **A(n)={1 if n=1**

    {2A(n/2) +n if n>1

    Solve the recurrence using the recursion tree method as



Complexity Analysis –Merge Sort

- Ahe recurrence for merge sort can be written as
  - **A(n)={1 if n=1**

    {2A(n/2) + n if n>1

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

$=2[2A(n/4)+n/2]+n$

$=4A(n/4)+n+n$

$=4[2A(n/8)+n/4]+n+n$

$=8A(n/8)+n+n+n$

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

$=2^kA(n/2^k)+n+n+\ldots+n\ldots$(k times) ($n/2^k=1$ **=>** $n=2^k$ **=>** $\mathbf{k=\log_2 n}$**)**

$=2^kA(1)+kn = 2^k+kn = n+n\log_2 n = O(\mathbf{n\log_2 n})$

$T(n) = aT(n/b) + f(n)$, where $f(n) \in \Theta(n^k)$

By Masters Theorem

a=2,b=2,k=1

=>a=b^k => $T(n) \in \Theta(n^k \lg n)$

=> O(nlog₂n)

Pros & Cons of Merge Sort

- Pros
  - Large Size List
  - Can be implemented using Linked List easily
  - Supports External Sorting
  - Stable
- Cons
  - Takes Extra Space(Not in place sorting)
  - There's no small problem
  - Recursive


Thank you!