COMP108
Data Structures and Algorithms

Divide-and-Conquer Algorithms (Part II Merge Sort Algorithm)

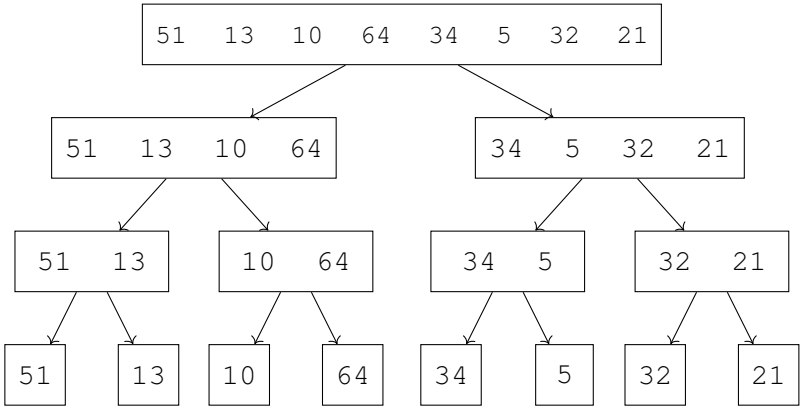
Professor Prudence Wong

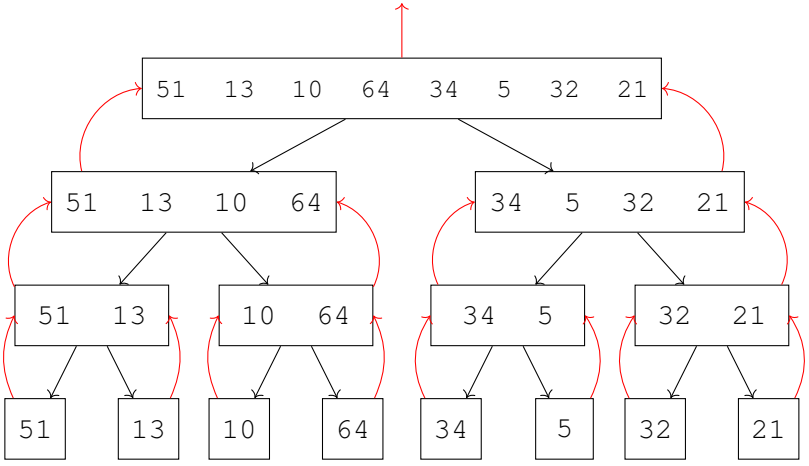
pwong@liverpool.ac.uk

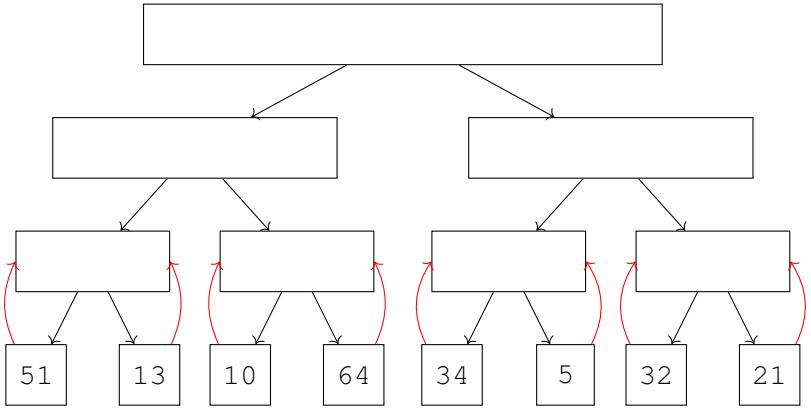
2022-23

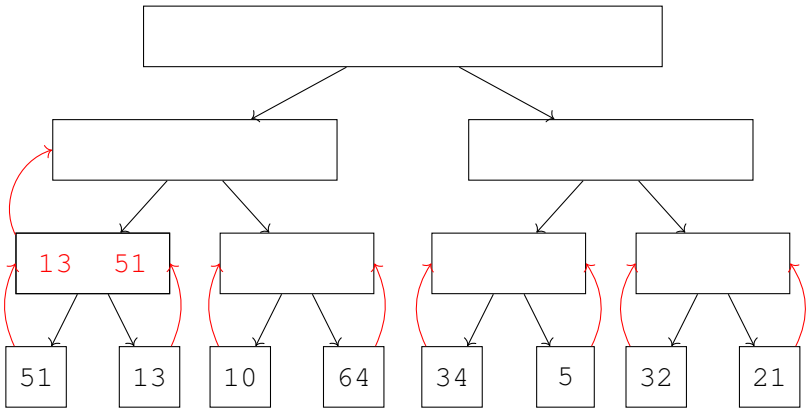
Merge Sort

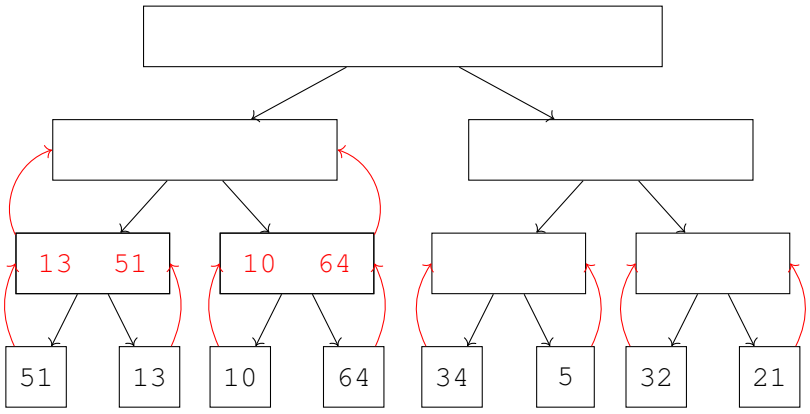
- using divide and conquer technique
- divide the sequence of n numbers into two halves
- recursively sort the two halves
- merge the two sorted halves into a single sorted sequence
  - It is easier to merge two sequences that are already sorted, compared to two non-sorted sequences.

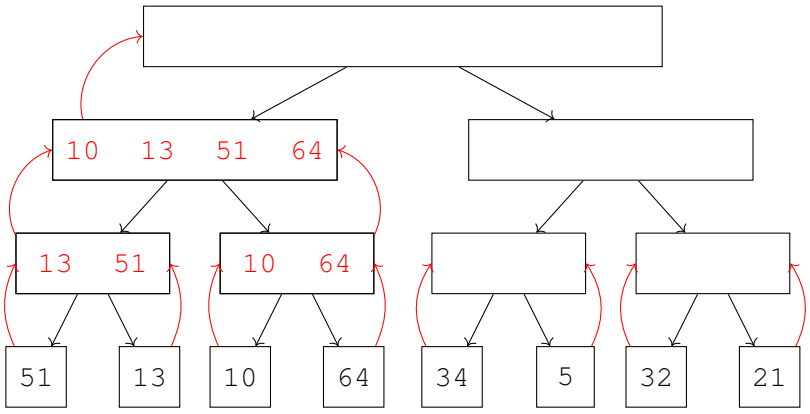

51 13 10 64 34 5 32 21
51 13 10 64
34 5 32 21
51 13
10 64
34 5
32 21
51
13
10
64
34
5
32
21

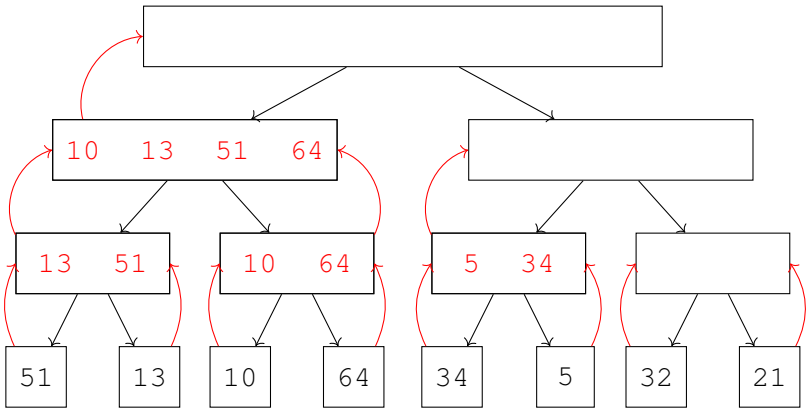

51 13 10 64 34 5 32 21
51 13 10 64
34 5 32 21
51 13
10 64
34 5
32 21
51
13
10
64
34
5
32
21

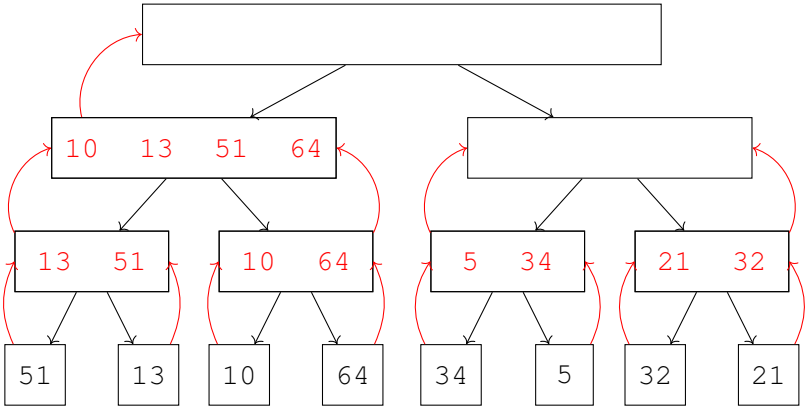

51
13
10
64
34
5
32
21

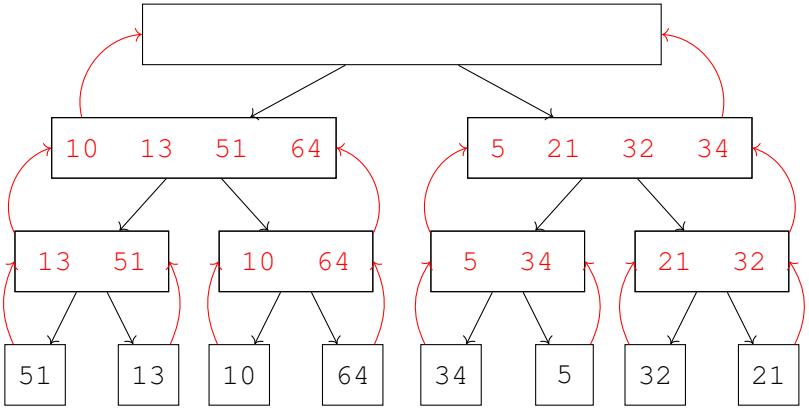

13 51
51
13
10
64
34
5
32
21

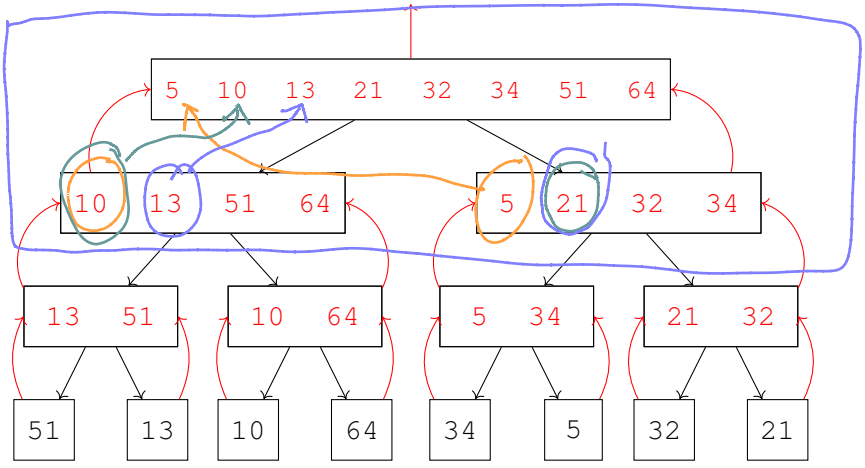

13 51
10 64
51
13
10
64
34
5
32
21


10 13 51 64
13 51
10 64
51
13
10
64
34
5
32
21


10 13 51 64
13 51
10 64
5 34
51
13
10
64
34
5
32
21


10 13 51 64
13 51
10 64
5 34
21 32
51
13
10
64
34
5
32
21


10 13 51 64
5 21 32 34
13 51
10 64
5 34
21 32
51
13
10
64
34
5
32
21


5 10 13 21 32 34 51 64
10 13 51 64
5 21 32 34
13 51
10 64
5 34
21 32
51
13
10
64
34
5
32
21

Merge Sort Algorithm

For simplicity, assume n is a power of 2

Algorithm MergeSort($A[1..n]$)

if $n > 1$ then

begin

end

Merge Sort Algorithm

For simplicity, assume n is a power of 2

Algorithm MergeSort($A[1..n]$)

if $n > 1$ then

begin

copy $A[1..\frac{n}{2}]$ to $B[1..\frac{n}{2}]$

copy $A[(\frac{n}{2}+1)..n]$ to $C[1..\frac{n}{2}]$

end

Merge Sort Algorithm

For simplicity, assume n is a power of 2

Algorithm MergeSort($A[1..n]$)

if $n > 1$ then

begin

copy $A[1..\frac{n}{2}]$ to $B[1..\frac{n}{2}]$

copy $A[(\frac{n}{2}+1)..n]$ to $C[1..\frac{n}{2}]$

MergeSort(B)

MergSort(C)

end

Merge Sort Algorithm

For simplicity, assume n is a power of 2

Algorithm MergeSort($A[1..n]$)

if $n > 1$ then

begin

copy $A[1..\frac{n}{2}]$ to $B[1..\frac{n}{2}]$

copy $A[(\frac{n}{2}+1)..n]$ to $C[1..\frac{n}{2}]$

MergeSort(B)

MergSort(C)

Merge(B, C, A)

end

when B and C are separately sorted,
we merge B and C back to A in correct order

Merge Sort Algorithm

For simplicity, assume n is a power of 2

Algorithm MergeSort($A[1..n]$)

if $n > 1$ then

begin

copy $A[1..\frac{n}{2}]$ to $B[1..\frac{n}{2}]$

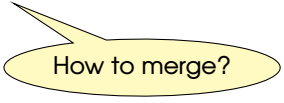
copy $A[(\frac{n}{2}+1)..n]$ to $C[1..\frac{n}{2}]$

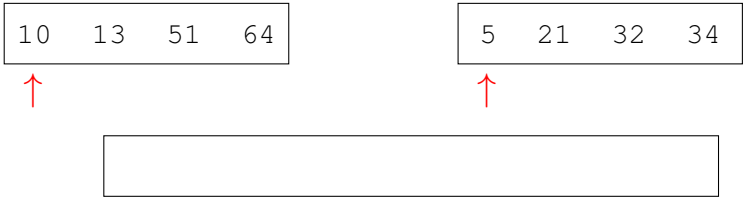
MergeSort(B)

MergSort(C)

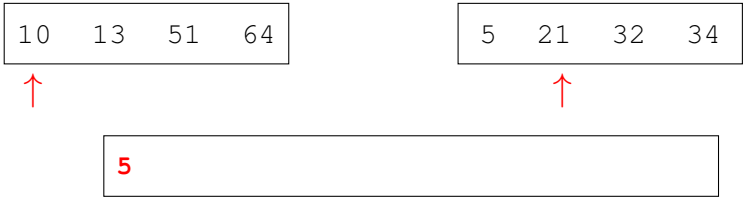
Merge(B, C, A)

end

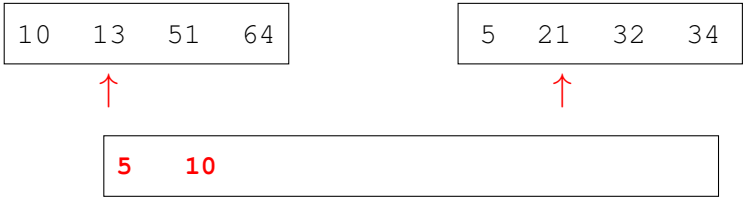




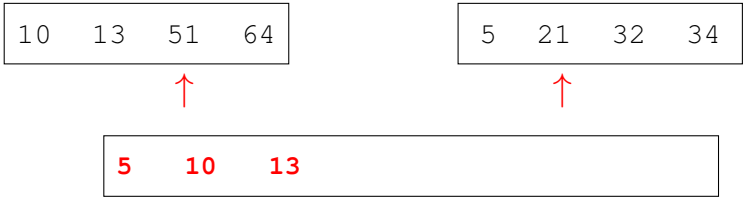
To merge two sorted sequences, we keep two **pointers**, one to each sequence
Compare the two numbers pointed, copy the **smaller** one to the result and **advance** the corresponding pointer



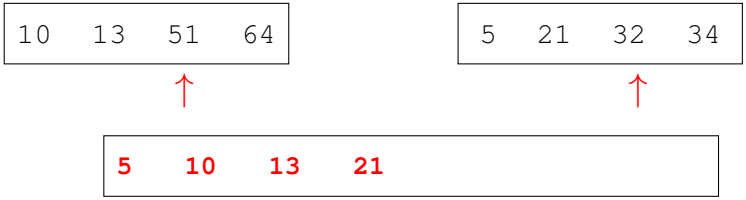
To merge two sorted sequences, we keep two **pointers**, one to each sequence
Compare the two numbers pointed, copy the **smaller** one to the result and **advance** the corresponding pointer



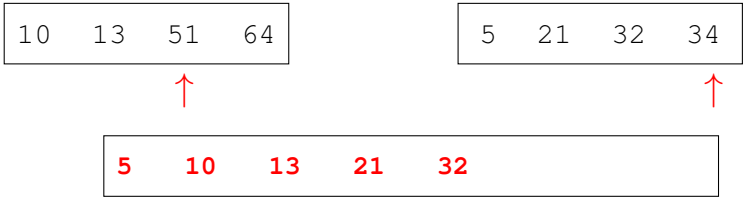
Repeat the process · · ·



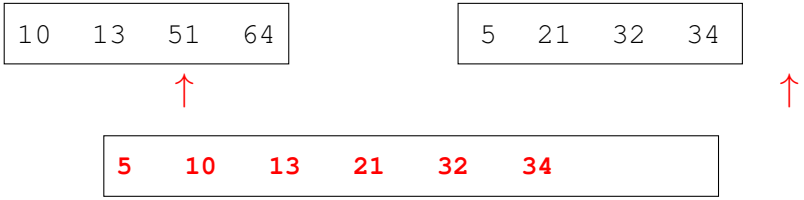
Repeat the process · · ·



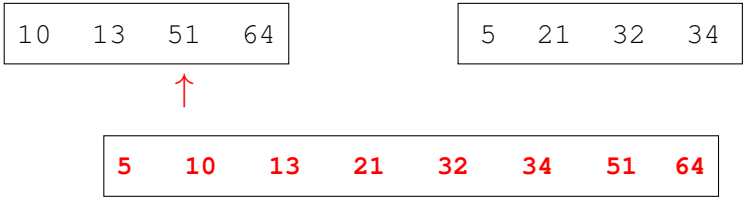
Repeat the process · · ·



Repeat the process · · ·



When we reach the **end** of one sequence, simply copy the **remaining** numbers in the other sequence to the result

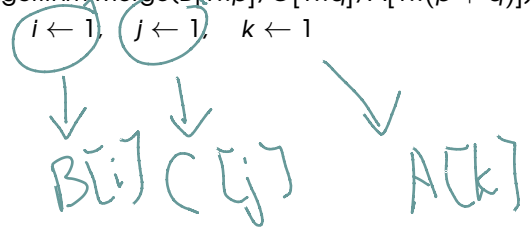


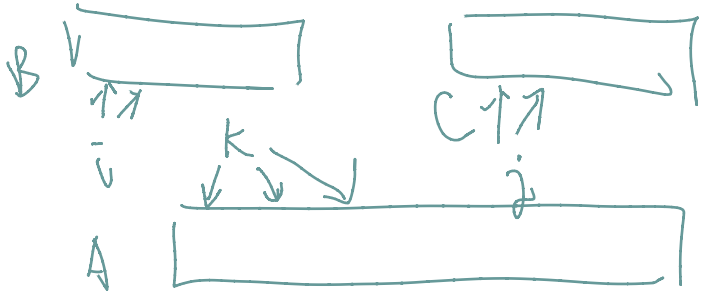
Then we obtain the final sorted sequence

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow 1$





Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

$k \leftarrow k + 1$

end

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

else

$k \leftarrow k + 1$

end

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$\mathbf{A[k] \leftarrow B[i], \quad i \leftarrow i + 1}$

else

$k \leftarrow k + 1$

end

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$\mathbf{A[k] \leftarrow B[i], \quad i \leftarrow i + 1}$

else

$\mathbf{A[k] \leftarrow C[j], \quad j \leftarrow j + 1}$

$k \leftarrow k + 1$

end

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$A[k] \leftarrow B[i], \quad i \leftarrow i+1$

else

$A[k] \leftarrow C[j], \quad j \leftarrow j+1$

$k \leftarrow k+1$

end

if $i == p+1$ then `B is exhausted => copy remaining of C`

else

`means j becomes q+1 => C is exhausted => copy remaining of B`

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$\mathbf{A[k] \leftarrow B[i], \quad i \leftarrow i+1}$

else

$\mathbf{A[k] \leftarrow C[j], \quad j \leftarrow j+1}$

$k \leftarrow k+1$

end

if $i == p+1$ then

copy $\mathbf{C[j..q]}$ to $\mathbf{A[k..(p+q)]}$

else

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$A[k] \leftarrow B[i], \quad i \leftarrow i+1$

else

$A[k] \leftarrow C[j], \quad j \leftarrow j+1$

$k \leftarrow k+1$

end

if $i == p+1$ then

copy $C[j..q]$ to $A[k..(p+q)]$

else

copy $B[i..p]$ to $A[k..(p+q)]$

Merge Algorithm

Algorithm Merge($B[1..p]$, $C[1..q]$, $A[1..(p+q)]$)

$i \leftarrow 1, \quad j \leftarrow 1, \quad k \leftarrow 1$

while $i \leq p$ AND $j \leq q$ do

begin

if $B[i] \leq C[j]$ then

$A[k] \leftarrow B[i], \quad i \leftarrow i + 1$

else

$A[k] \leftarrow C[j], \quad j \leftarrow j + 1$

$k \leftarrow k + 1$

end

if $i == p + 1$ then

copy $C[j..q]$ to $A[k..(p+q)]$

else

copy $B[i..p]$ to $A[k..(p+q)]$

Merge Algorithm

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty

Merge Algorithm

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5

Merge Algorithm

p=4

B: | 10 | 13 | 51 | 64 |

q=4

C: | 5 | 21 | 32 | 34 |

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10

Merge Algorithm

p=4

B: | 10 | 13 | 51 | 64 |

q=4

C: | 5 | 21 | 32 | 34 |

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10
end of 3rd	3	2	4	5 10 13

Merge Algorithm

p=4

B: | 10 | 13 | 51 | 64 |

q=4

C: | 5 | 21 | 32 | 34 |

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10
end of 3rd	3	2	4	5 10 13
end of 4th	3	3	5	5 10 13 21

Merge Algorithm

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10
end of 3rd	3	2	4	5 10 13
end of 4th	3	3	5	5 10 13 21
end of 5th	3	4	6	5 10 13 21 32

Merge Algorithm

p=4

B: | 10 | 13 | 51 | 64 |

q=4

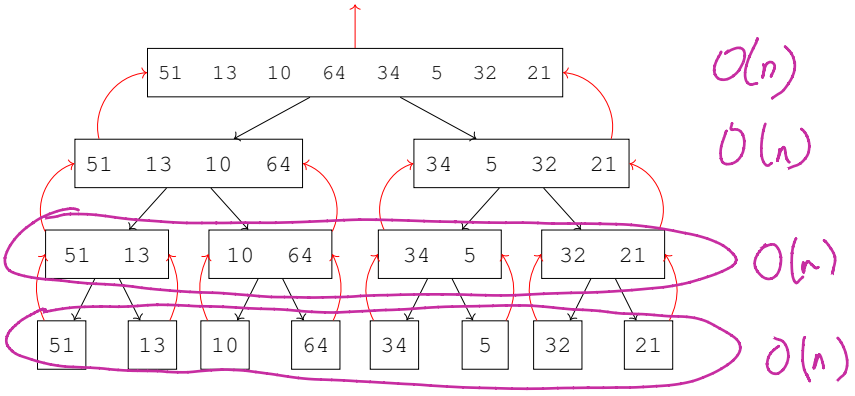
C: | 5 | 21 | 32 | 34 |

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10
end of 3rd	3	2	4	5 10 13
end of 4th	3	3	5	5 10 13 21
end of 5th	3	4	6	5 10 13 21 32
end of 6th	3	5	7	5 10 13 21 32 34

Merge Algorithm

	i	*j*	*k*	*A*[ ]
before loop	1	1	1	empty
end of 1st iteration	1	2	2	5
end of 2nd iteration	2	2	3	5 10
end of 3rd	3	2	4	5 10 13
end of 4th	3	3	5	5 10 13 21
end of 5th	3	4	6	5 10 13 21 32
end of 6th	3	5	7	5 10 13 21 32 34
after if-else				5 10 13 21 32 34 51 64

Time complexity analysis



- Each node takes $O(r)$ time when there are r integers.
- Each level takes $O(n)$ time because total number of integers is n.
- There are $O(\log n)$ levels.
- Overall: $\boldsymbol{O(n \log n)}$ time.

Summary

Summary: Merge Sort Algorithm

Next: Fibonacci Numbers

For note taking