COMP108
Data Structures and Algorithms

Divide-and-Conquer Algorithms (Part I)

Professor Prudence Wong

pwong@liverpool.ac.uk

2022-23

Outline

Divide-and-Conquer algorithms

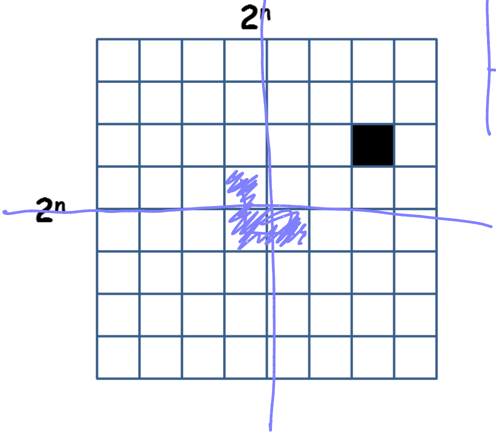
- Basic idea
- Learn a few examples

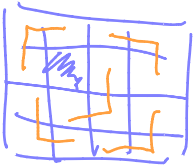
Learning outcomes:

- Able to describe the principle of divide-and-conquer algorithms
- Able to design divide-and-conquer algorithm for some simple problems

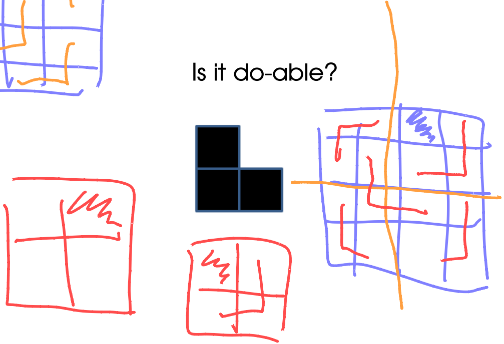
Triomino Puzzle

- **Input:** **2^n-by-2^n** chessboard with **one** missing square & many **L-shaped** tiles of **3** adjacent squares
- **Question:** **Cover** the chessboard with L-shaped tiles without overlapping





Is it do-able?



Divide-and-conquer algorithms

One of the best-known algorithm design techniques

Idea:

- A problem instance is divided into several smaller instances of the same problem, ideally of about same size
- The smaller instances are solved, typically recursively
- The solutions for the smaller instances are combined to get a solution to the large instance

Some problems we have seen before

Finding the sum of all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1

Iterative version:

sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

Some problems we have seen before

Finding the sum of all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: let's divide them into two halves
  $\underbrace{4\ 6\ 3\ 2}$ AND $\underbrace{8\ 7\ 5\ 1}$

  Sum1 + Sum2

Iterative version:
sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

Some problems we have seen before

Finding the sum of all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: let's divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the sum of each half, then we can simply add these two sums

Iterative version:

sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

Some problems we have seen before

Finding the sum of all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: let's divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the sum of each half, then we can simply add these two sums
- Recursively: How to find the sum of each half?
  Apply divide-and-conquer on each half

Iterative version:

sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

Some problems we have seen before

Finding the sum of all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: let's divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the sum of each half, then we can simply add these two sums
- Recursively: How to find the sum of each half? *Apply* divide-and-conquer on each half
- When to stop: When there is only one number left, simply return this number (sum of one number is the number itself)

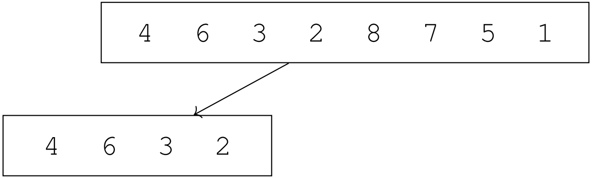
Iterative version:

sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

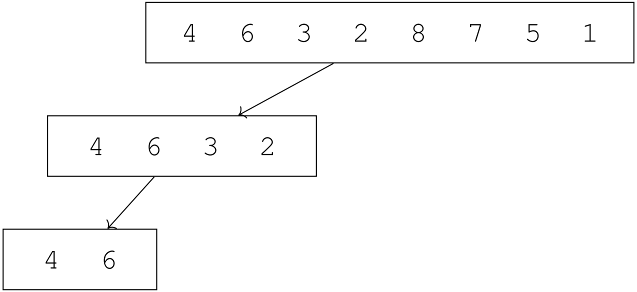
Divide-and-conquer to find the sum

4	6	3	2	8	7	5	1

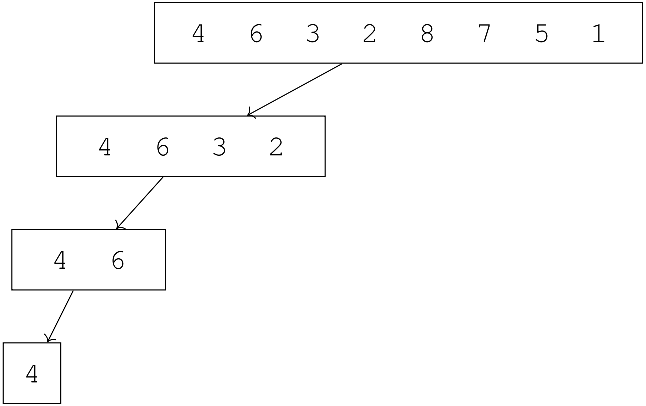
Divide-and-conquer to find the sum



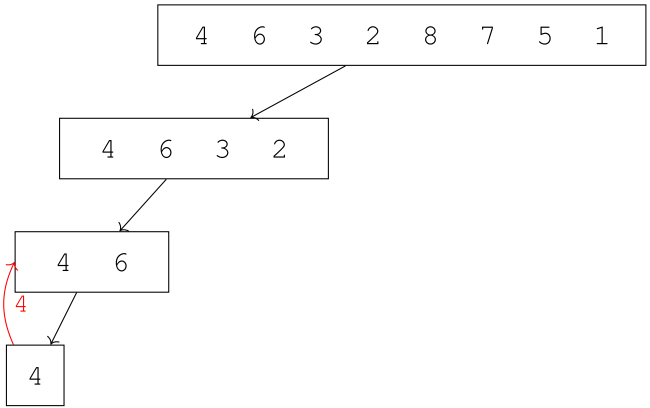
Divide-and-conquer to find the sum



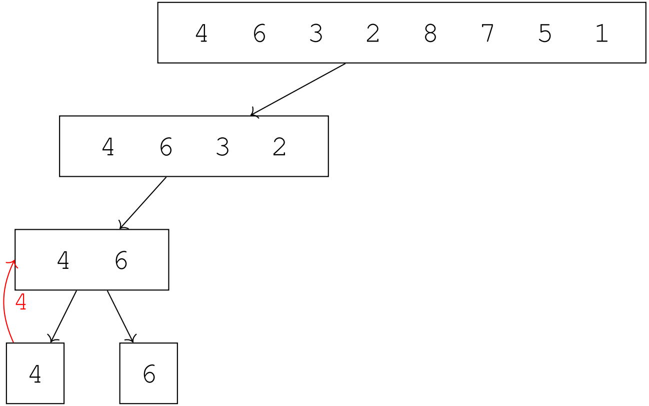
Divide-and-conquer to find the sum



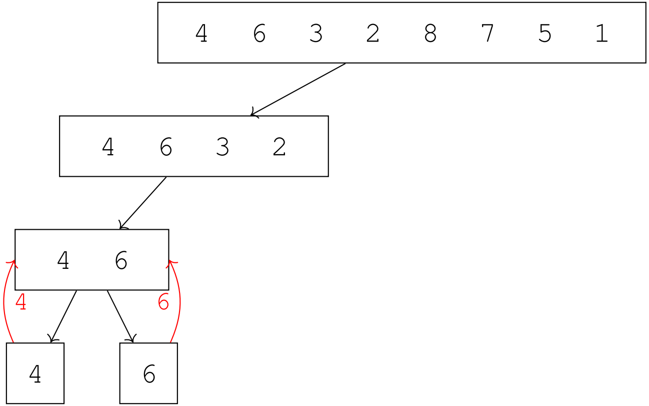
Divide-and-conquer to find the sum



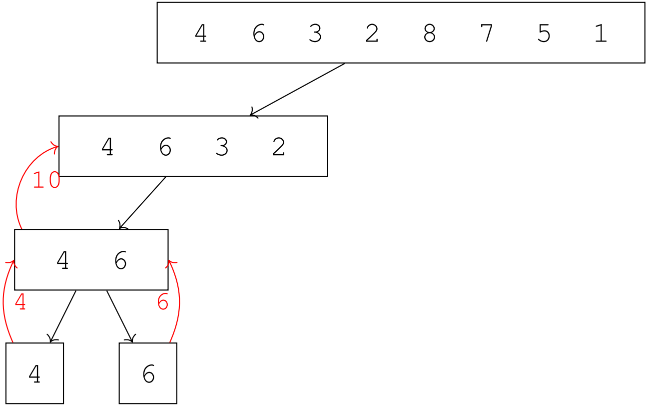
Divide-and-conquer to find the sum



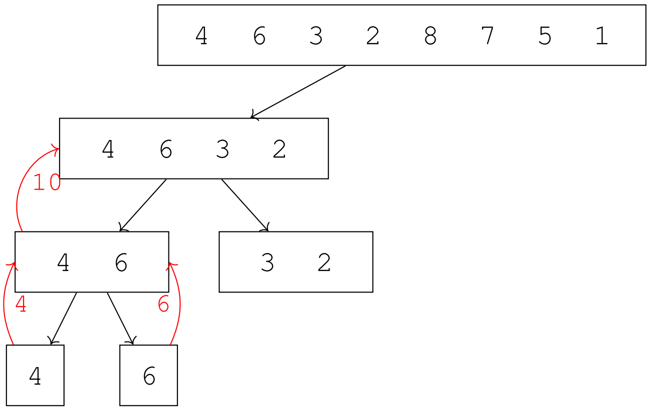
Divide-and-conquer to find the sum



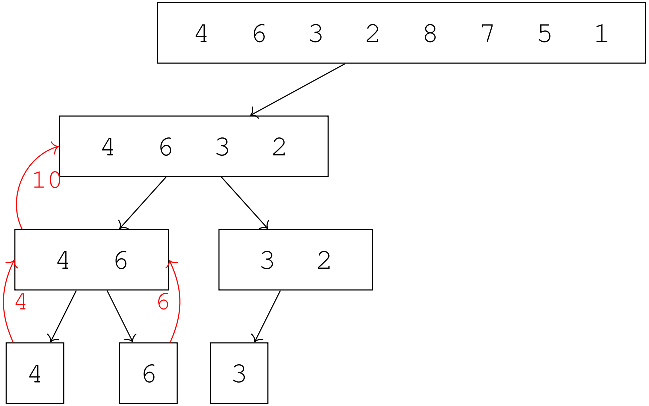
Divide-and-conquer to find the sum



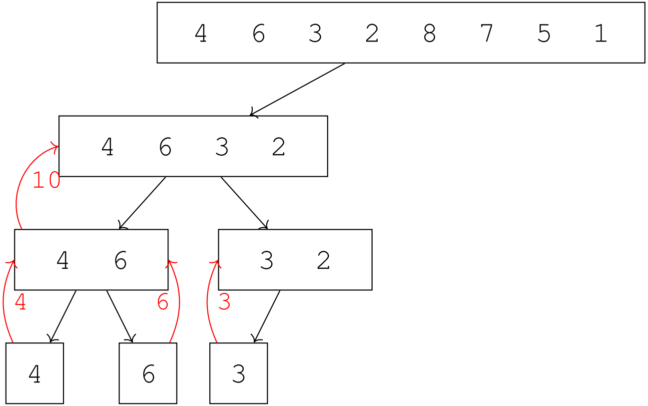
Divide-and-conquer to find the sum



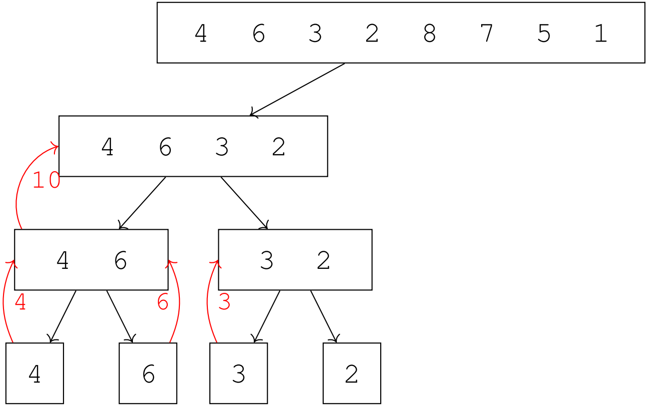
Divide-and-conquer to find the sum



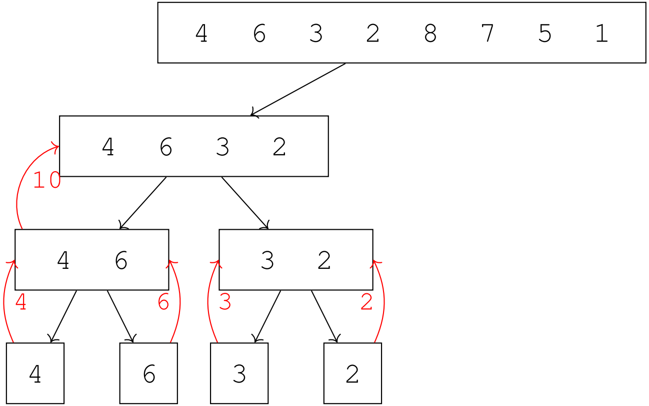
Divide-and-conquer to find the sum



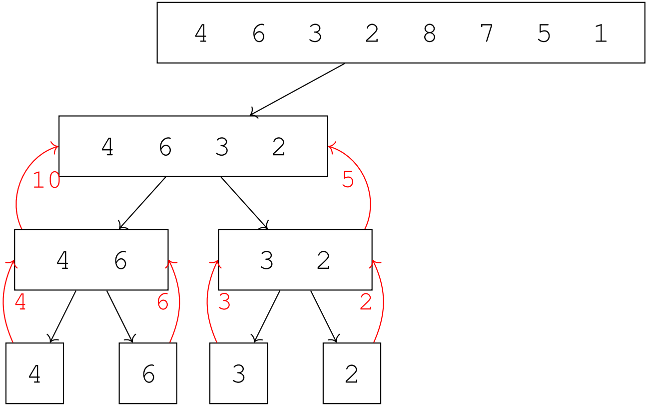
Divide-and-conquer to find the sum



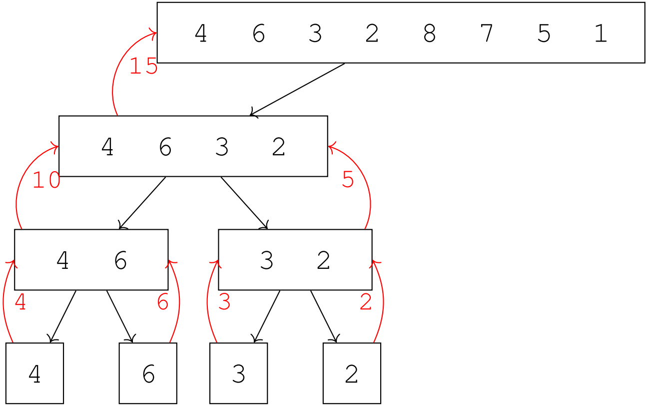
Divide-and-conquer to find the sum



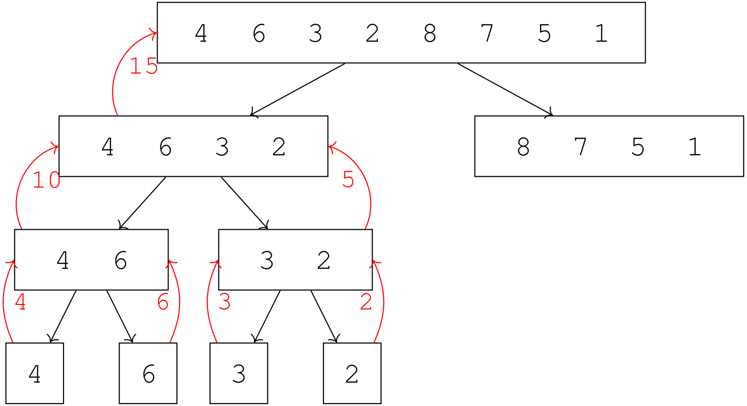
Divide-and-conquer to find the sum



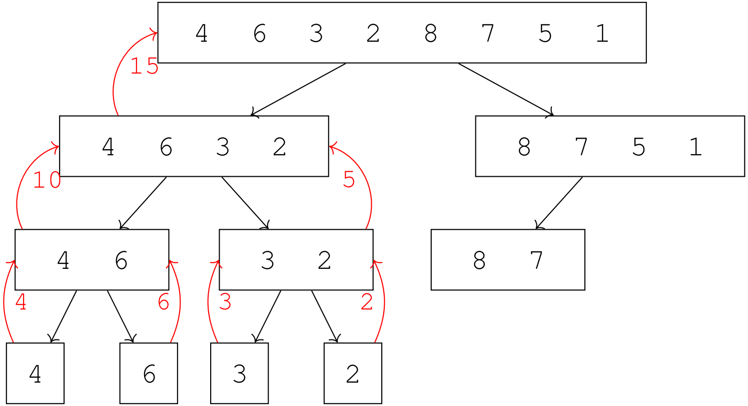
Divide-and-conquer to find the sum



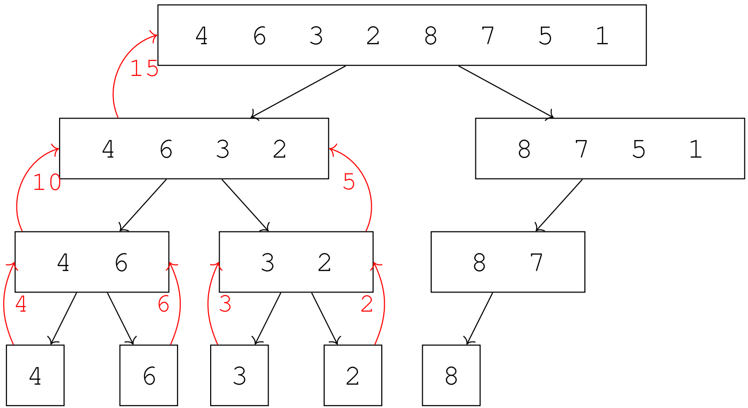
Divide-and-conquer to find the sum



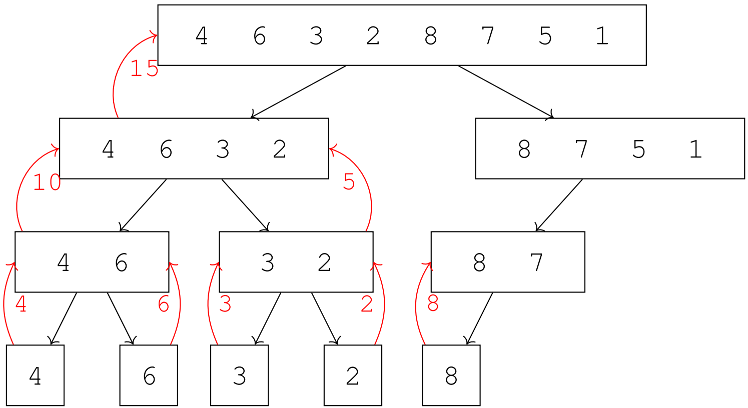
Divide-and-conquer to find the sum



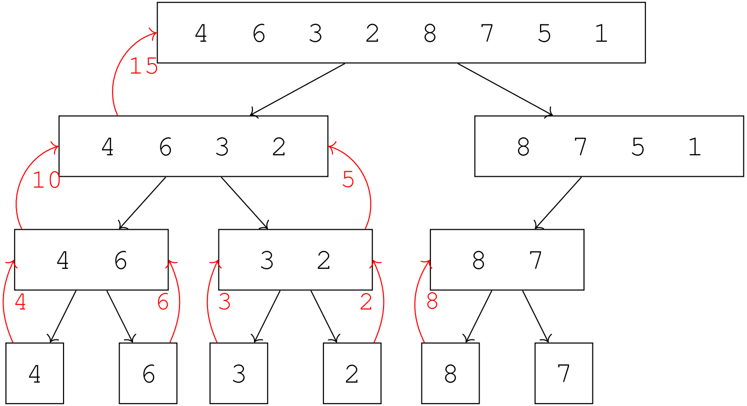
Divide-and-conquer to find the sum



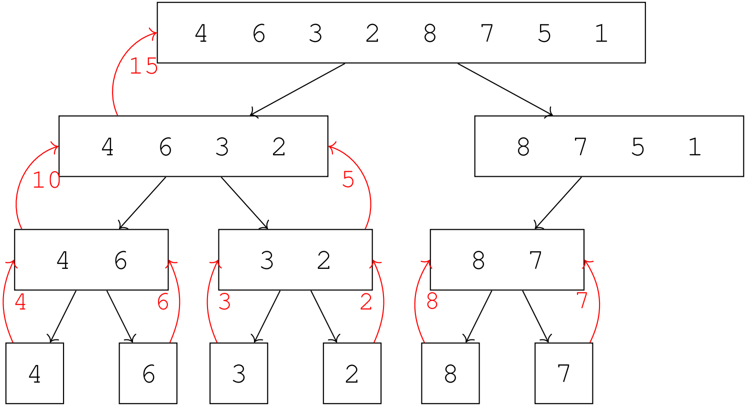
Divide-and-conquer to find the sum



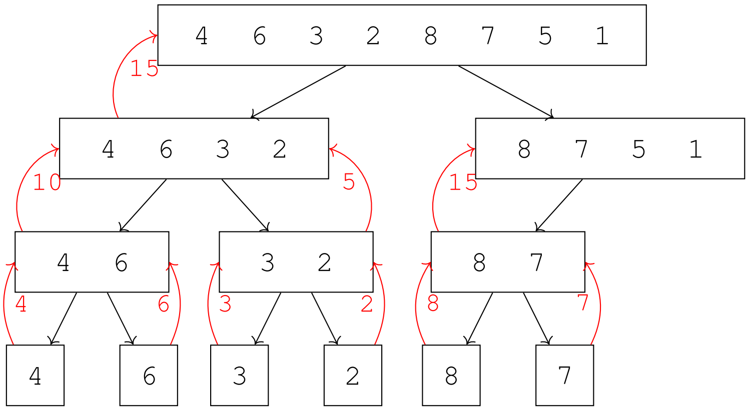
Divide-and-conquer to find the sum



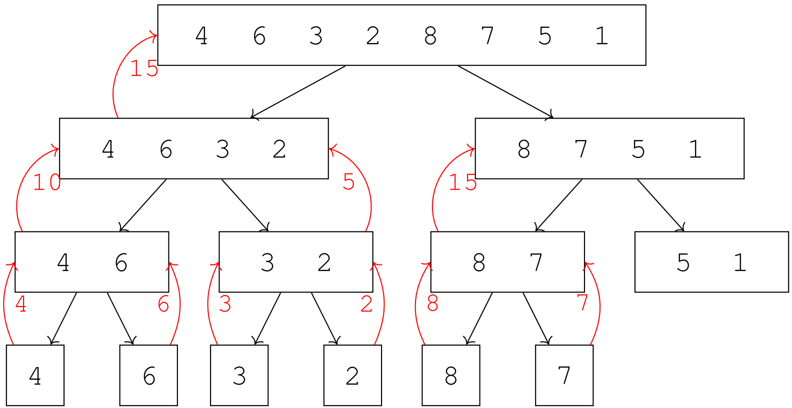
Divide-and-conquer to find the sum



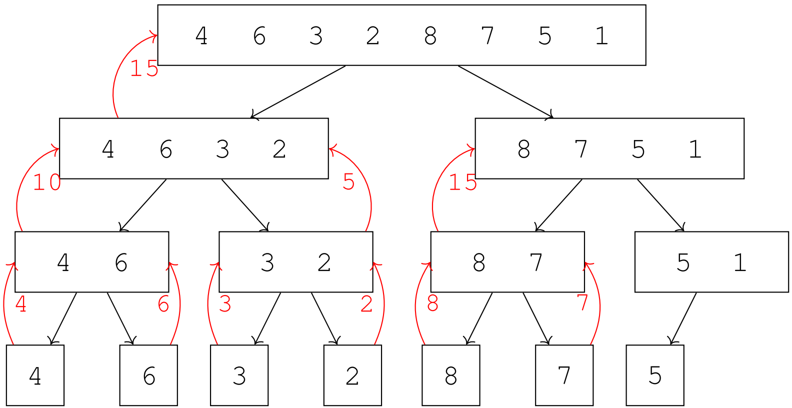
Divide-and-conquer to find the sum



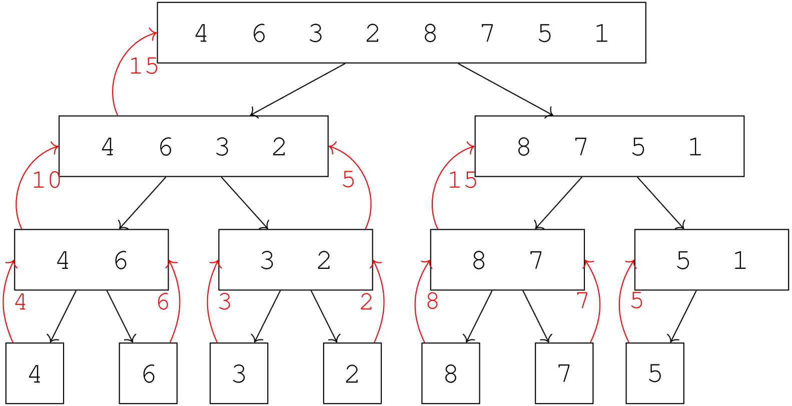
Divide-and-conquer to find the sum



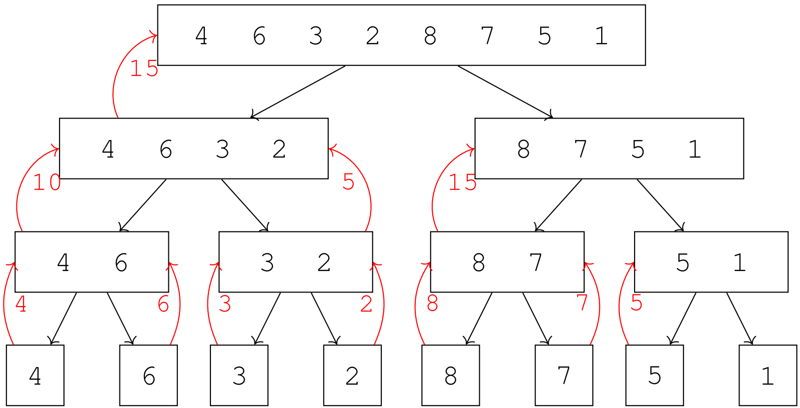
Divide-and-conquer to find the sum



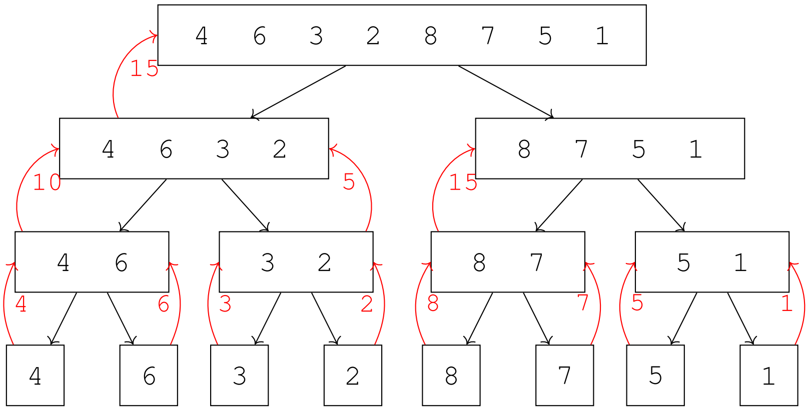
Divide-and-conquer to find the sum



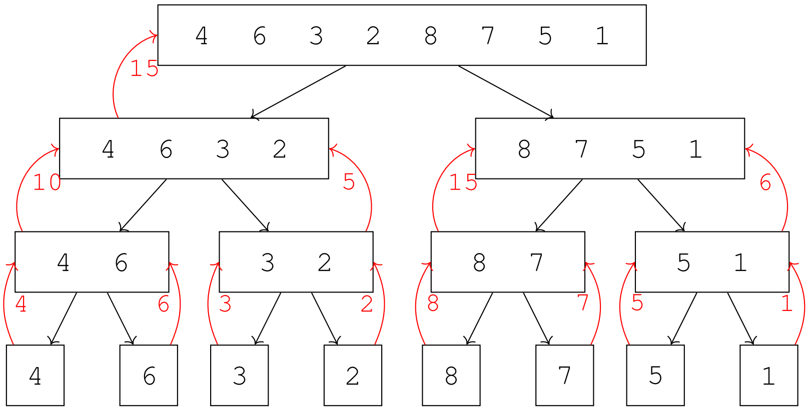
Divide-and-conquer to find the sum



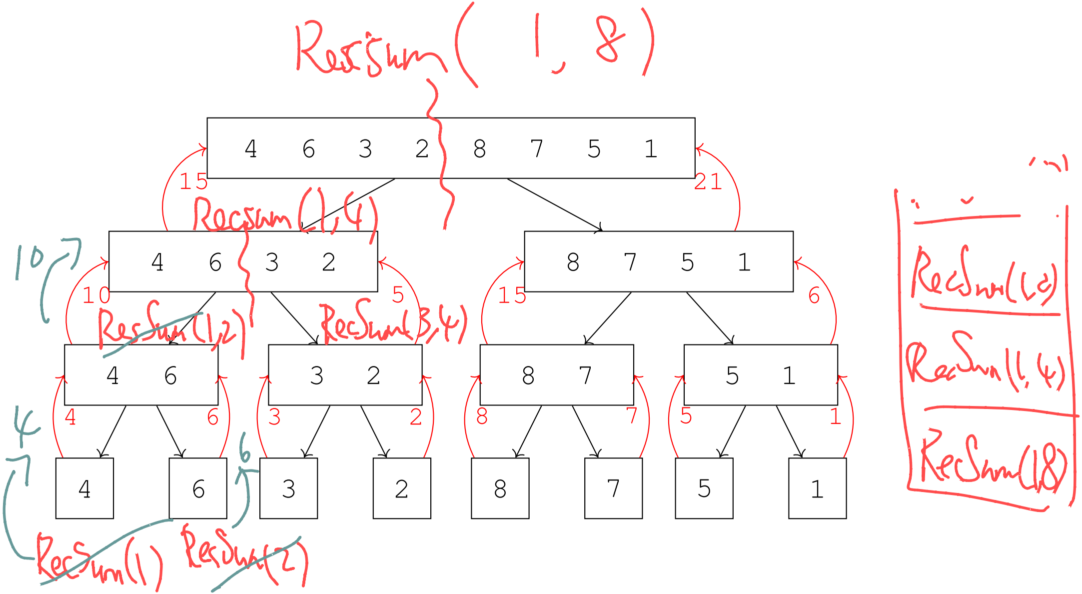
Divide-and-conquer to find the sum



Divide-and-conquer to find the sum

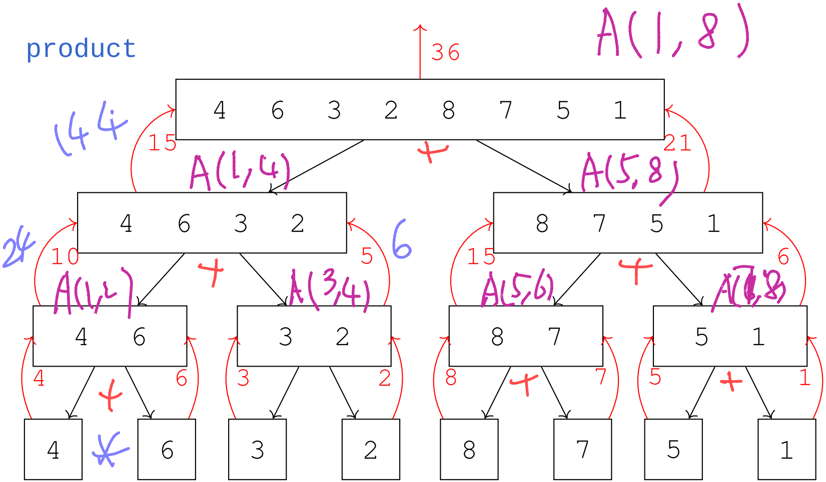


Divide-and-conquer to find the sum



Divide-and-conquer to find the sum

product



Divide-and-conquer to find the sum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecSum(A, 1, n)

Algorithm RecSum(A[ ], p, q)

```
p is index of entry of first element in the
current call
q … last ...
```

Iterative version:

sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
  sum $\leftarrow$ sum + $A[i]$
  i $\leftarrow$ i + 1
end
output sum

Divide-and-conquer to find the sum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecSum(A, 1, n)

```
Algorithm RecSum(A[ ], p, q)
    if p == q then
        return A[p]
    else
    begin



    end
```

Iterative version:

```
sum ← 0
i ← 1
while i ≤ n do
begin
   sum ← sum + A[i]
   i ← i + 1
end
output sum
```

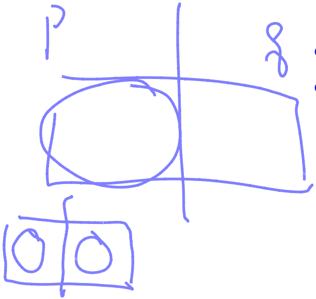
Divide-and-conquer to find the sum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecSum(A, 1, n)

Algorithm RecSum($A[\,]$, p, q)
    if $p == q$ then
        return $A[p]$
    else
    begin
        sum1 $\leftarrow$ RecSum(A, p, $\frac{p+q-1}{2}$)
        sum2 $\leftarrow$ RecSum(A, $\frac{p+q+1}{2}$, q)
        return sum1 + sum2
    end

*



Iterative version:
sum $\leftarrow$ 0
i $\leftarrow$ 1
while i $\leq$ n do
begin
    sum $\leftarrow$ sum + $A[i]$
    i $\leftarrow$ i + 1
end
output sum

Another problem we have seen before

Finding the minimum over all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1

Iterative version:

```
min ← A[1]
i ← 2
while i ≤ n do
begin
   if A[i] ≤ min then
      min ← A[i]
   i ← i + 1
end
output min
```

Another problem we have seen before

Finding the minimum over all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: divide them into two halves
  4 6 3 2 AND 8 7 5 1

Iterative version:
min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

Another problem we have seen before

Finding the minimum over all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the minimum of each half, then we can compare these two numbers and return the smaller one

Iterative version:

min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

Another problem we have seen before

Finding the minimum over all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the minimum of each half, then we can compare these two numbers and return the smaller one
- Recursively: How to find the minimum of each half?
  Apply divide-and-conquer on each half

Iterative version:

min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

Another problem we have seen before

Finding the minimum over all numbers in an array

- Suppose we have 8 numbers:
  4 6 3 2 8 7 5 1
- Divide: divide them into two halves
  4 6 3 2 AND 8 7 5 1
- Conquer: If we know the minimum of each half, then we can compare these two numbers and return the smaller one
- Recursively: How to find the minimum of each half?
  Apply divide-and-conquer on each half
- When to stop: When there is only one number left, simply return this number (the minimum of one number is the number itself)

Iterative version:

min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

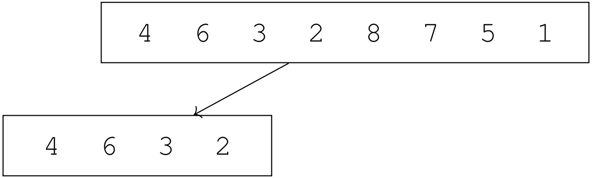
Divide-and-conquer to find the minimum

4	6	3	2	8	7	5	1

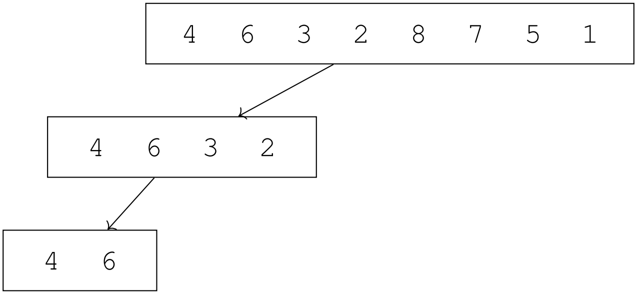
2

1

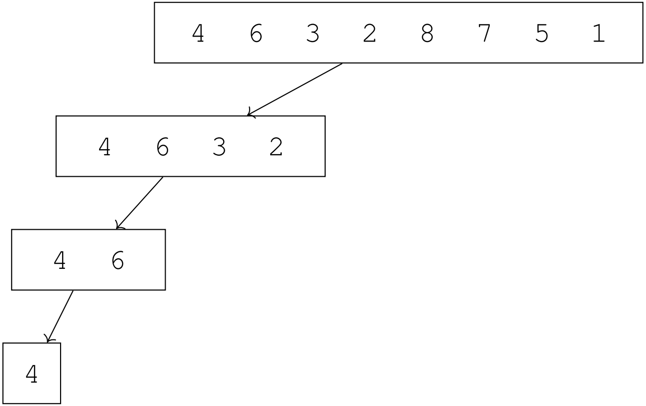
Divide-and-conquer to find the minimum



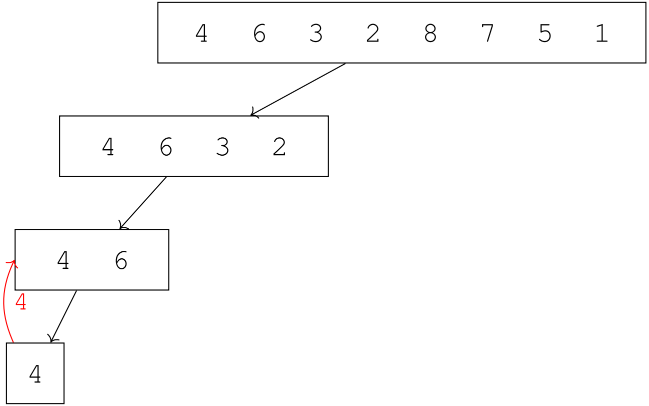
Divide-and-conquer to find the minimum



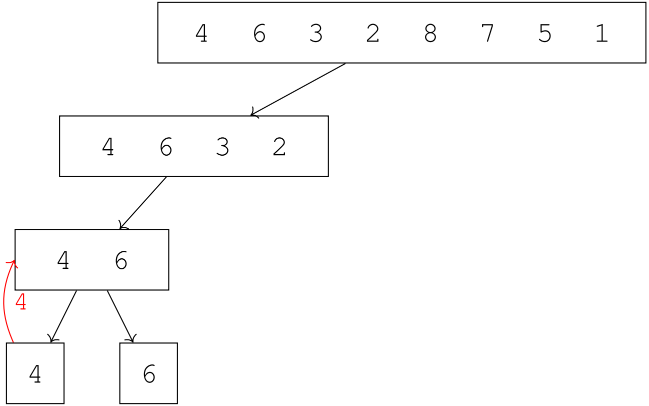
Divide-and-conquer to find the minimum



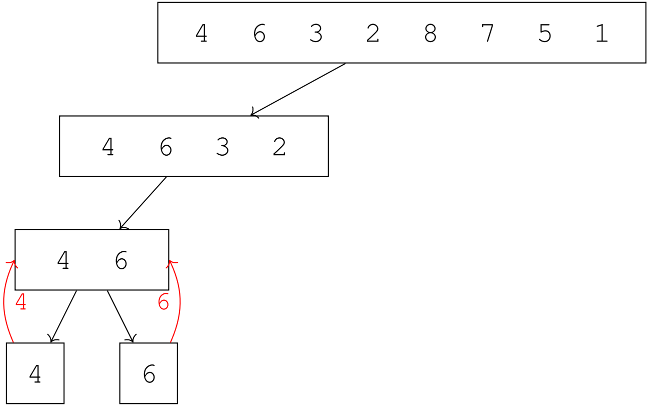
Divide-and-conquer to find the minimum



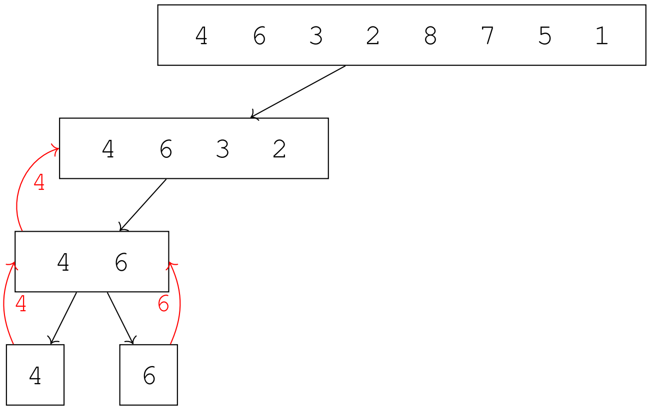
Divide-and-conquer to find the minimum



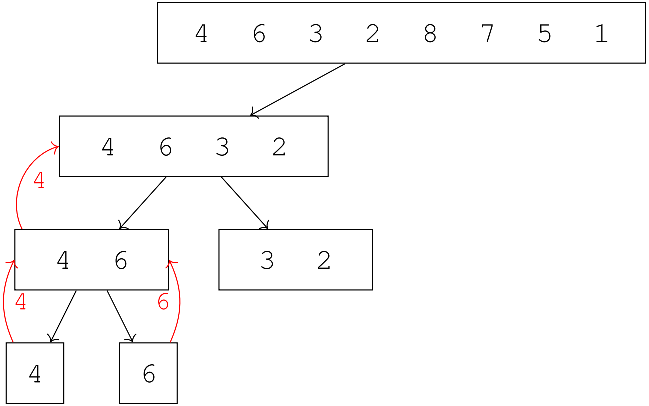
Divide-and-conquer to find the minimum



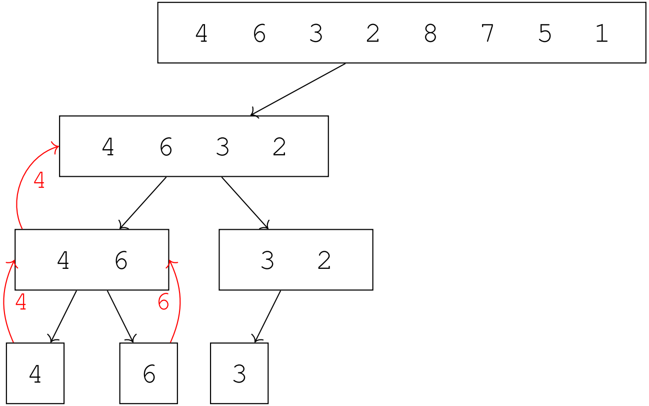
Divide-and-conquer to find the minimum



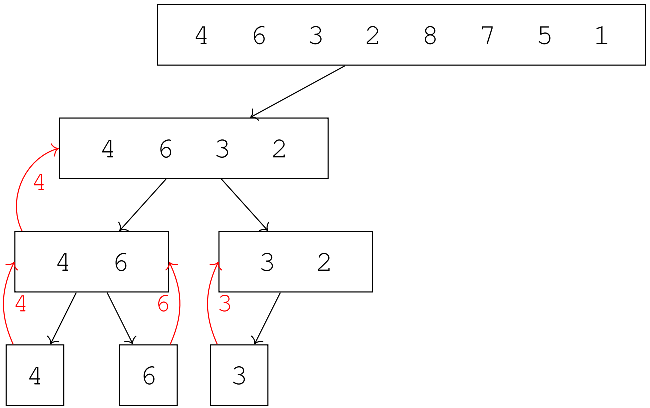
Divide-and-conquer to find the minimum



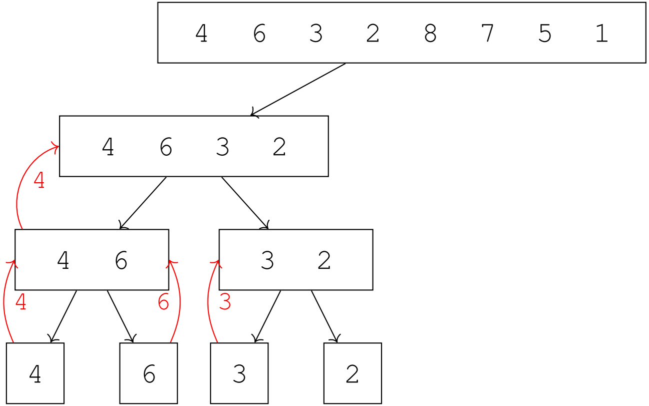
Divide-and-conquer to find the minimum



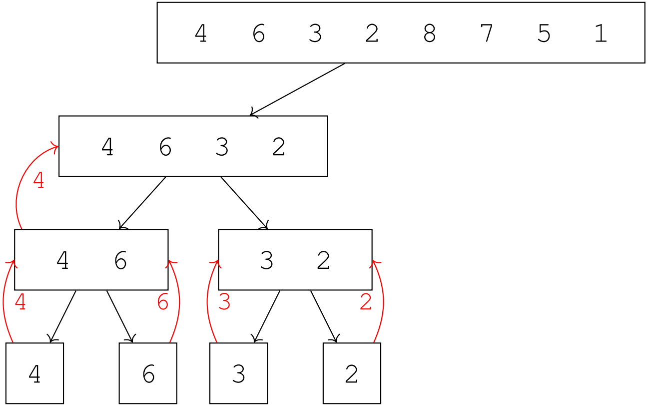
Divide-and-conquer to find the minimum



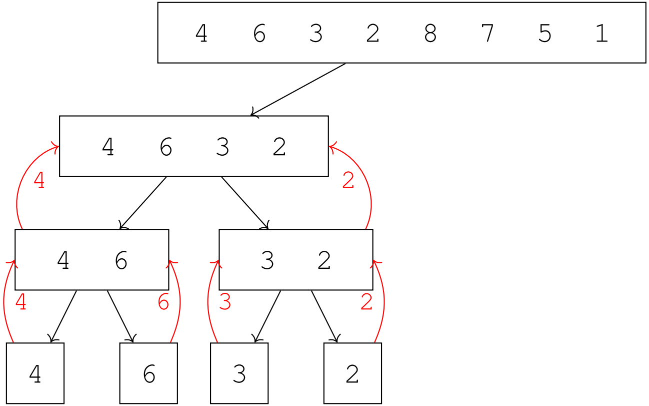
Divide-and-conquer to find the minimum



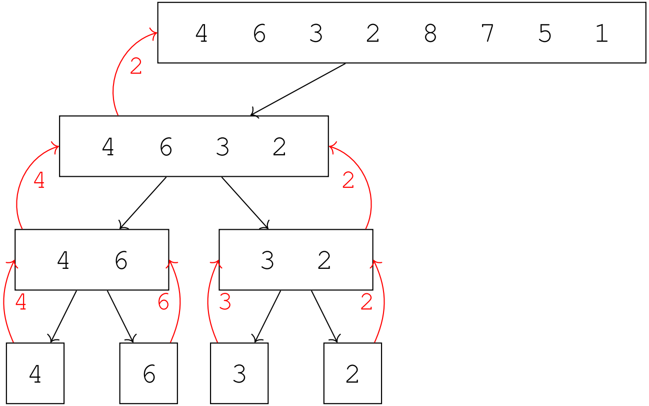
Divide-and-conquer to find the minimum



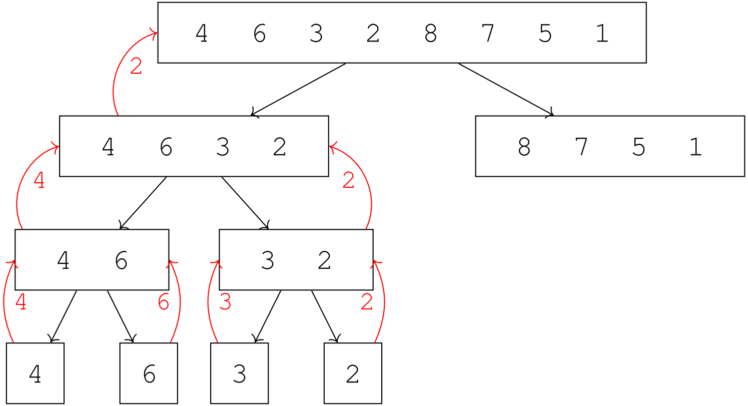
Divide-and-conquer to find the minimum



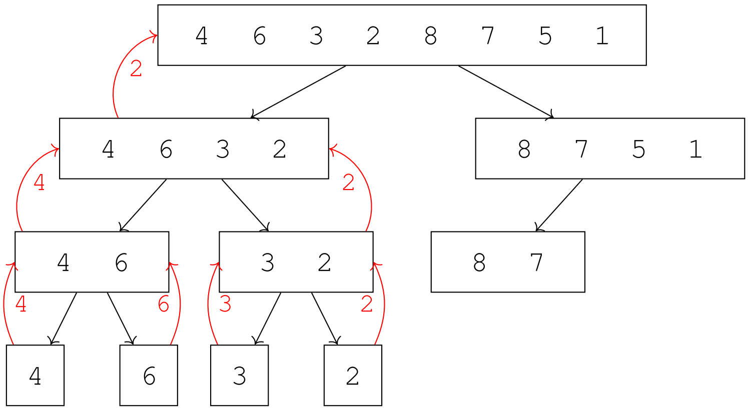
Divide-and-conquer to find the minimum



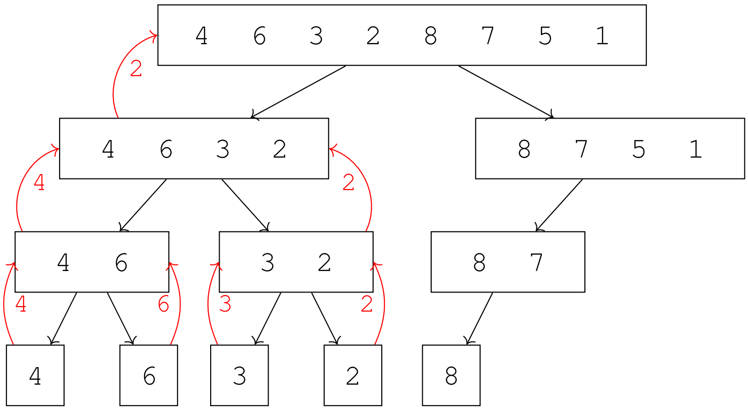
Divide-and-conquer to find the minimum



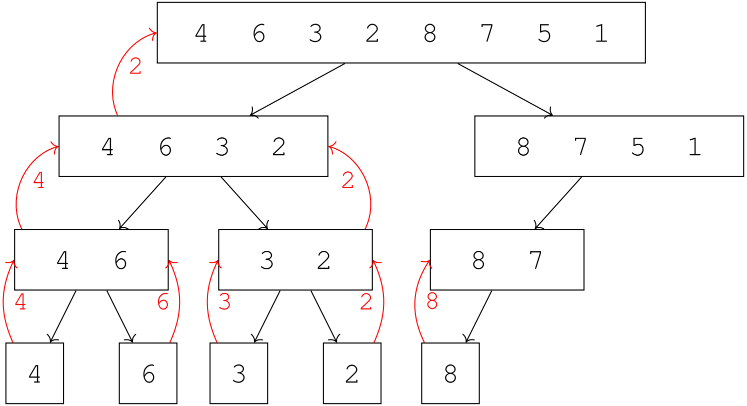
Divide-and-conquer to find the minimum



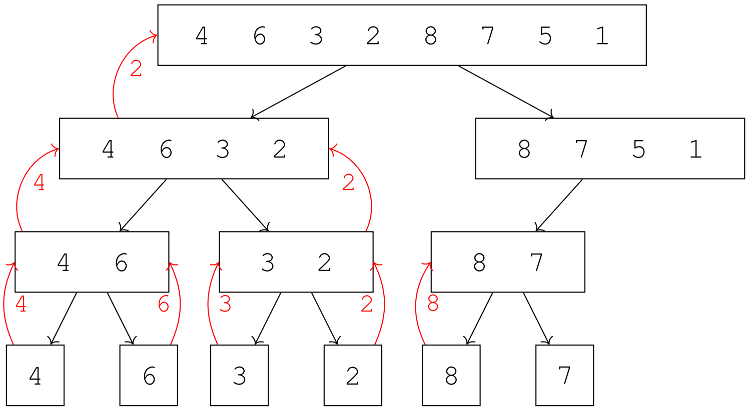
Divide-and-conquer to find the minimum



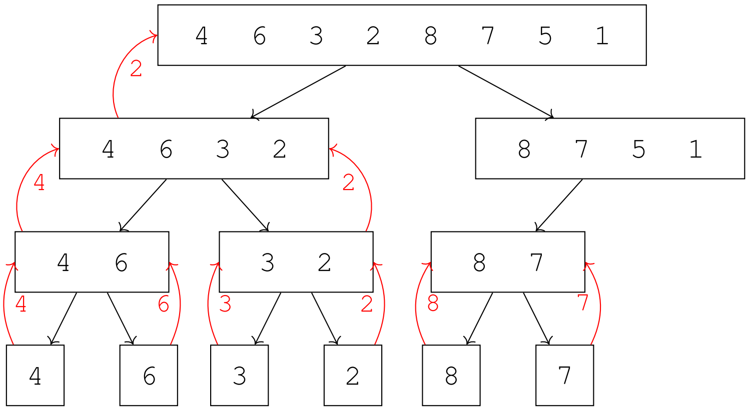
Divide-and-conquer to find the minimum



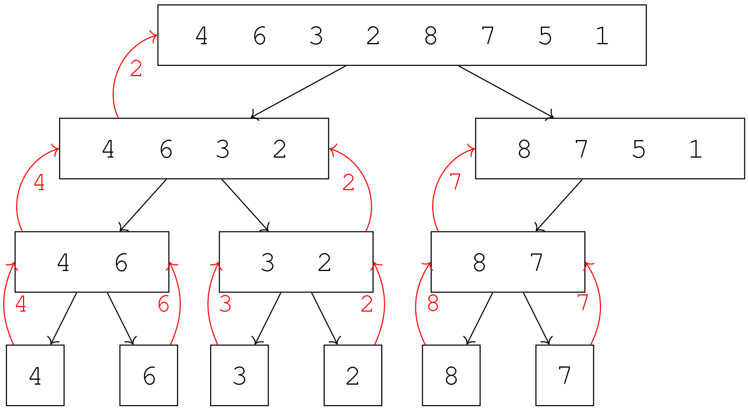
Divide-and-conquer to find the minimum



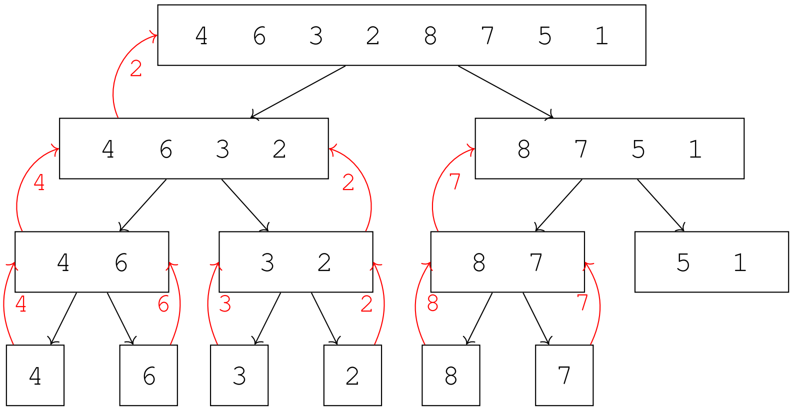
Divide-and-conquer to find the **minimum**



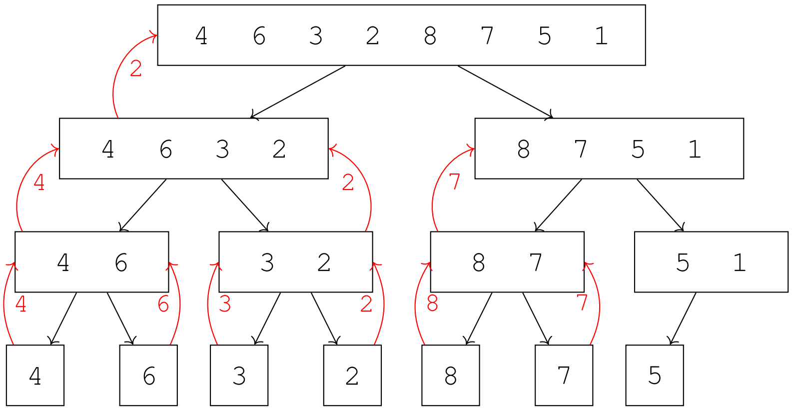
Divide-and-conquer to find the minimum



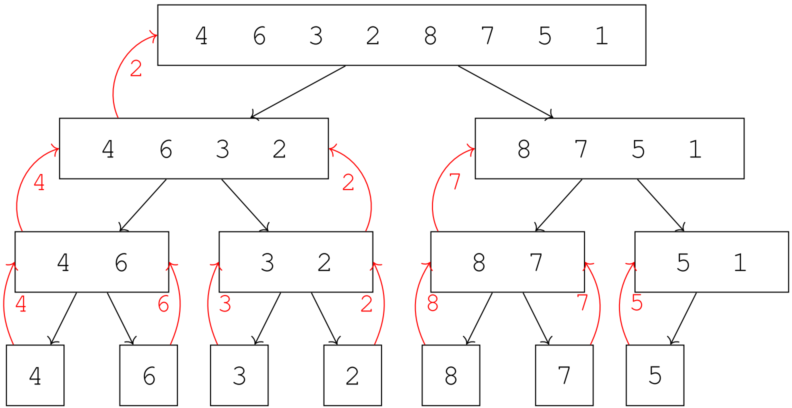
Divide-and-conquer to find the minimum



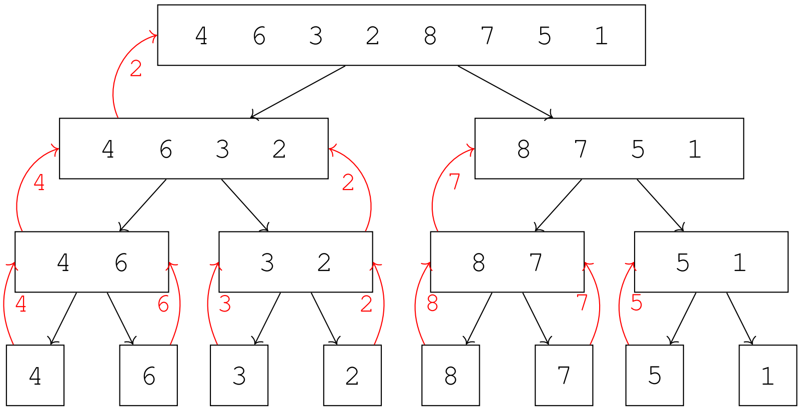
Divide-and-conquer to find the minimum



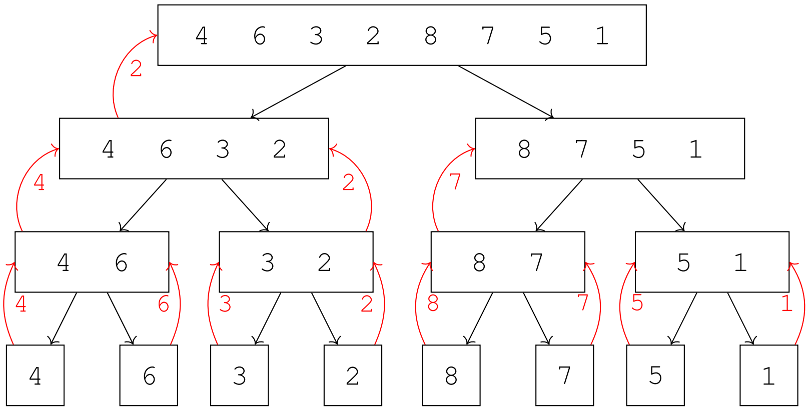
Divide-and-conquer to find the minimum



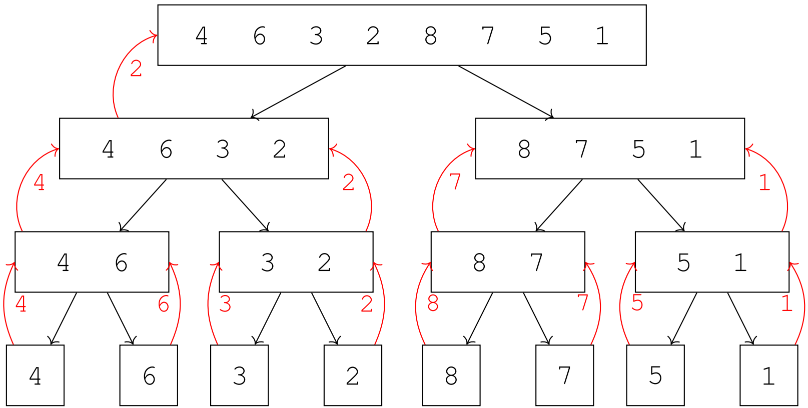
Divide-and-conquer to find the minimum



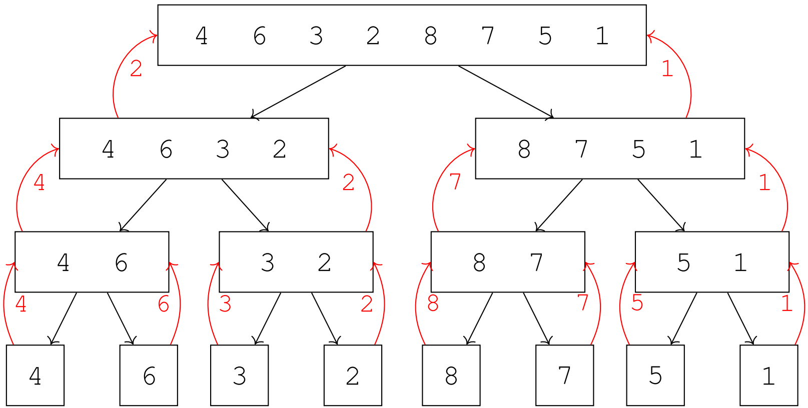
Divide-and-conquer to find the minimum



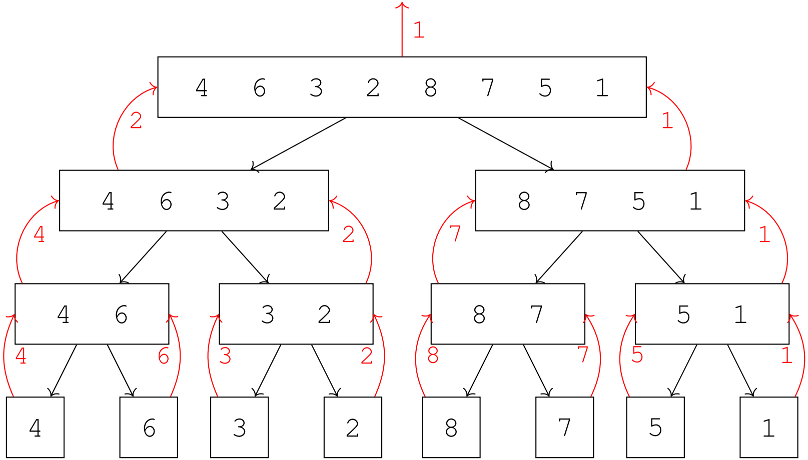
Divide-and-conquer to find the minimum



Divide-and-conquer to find the minimum

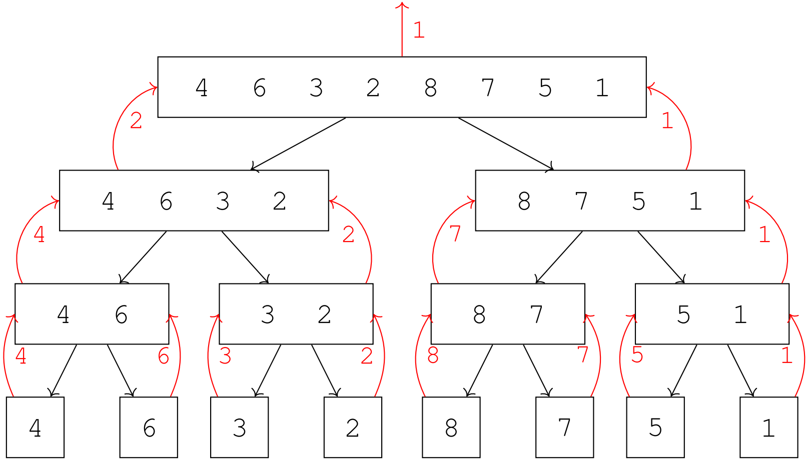


Divide-and-conquer to find the minimum



Divide-and-conquer to find the minimum

What about the maximum?



Divide-and-conquer to find the minimum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecMin(A, 1, n)

Algorithm RecMin($A[\,]$, p, q)

Iterative version:

min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

Divide-and-conquer to find the minimum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecMin(A, 1, n)

Algorithm RecMin($A[\,]$, p, q)
    if $p == q$ then
        return $A[p]$
    else begin

    end

Iterative version:
min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
    if $A[i] \leq$ min then
        min $\leftarrow A[i]$
    i $\leftarrow$ i + 1
end
output min

Divide-and-conquer to find the minimum

For simplicity, assume n is a power of 2

We can call the following algorithm by RecMin(A, 1, n)

Algorithm RecMin($A[\,]$, p, q)
  if $p == q$ then
    return $A[p]$
  else begin
    answer1 $\leftarrow$ RecMin(A, p, $\frac{p+q-1}{2}$)
    answer2 $\leftarrow$ RecMin(A, $\frac{p+q+1}{2}$, q)

  end

Iterative version:
min $\leftarrow A[1]$
i $\leftarrow$ 2
while i $\leq$ n do
begin
  if $A[i] \leq$ min then
    min $\leftarrow A[i]$
  i $\leftarrow$ i + 1
end
output min

Divide-and-conquer to find the minimum

For simplicity, assume n is a power of 2

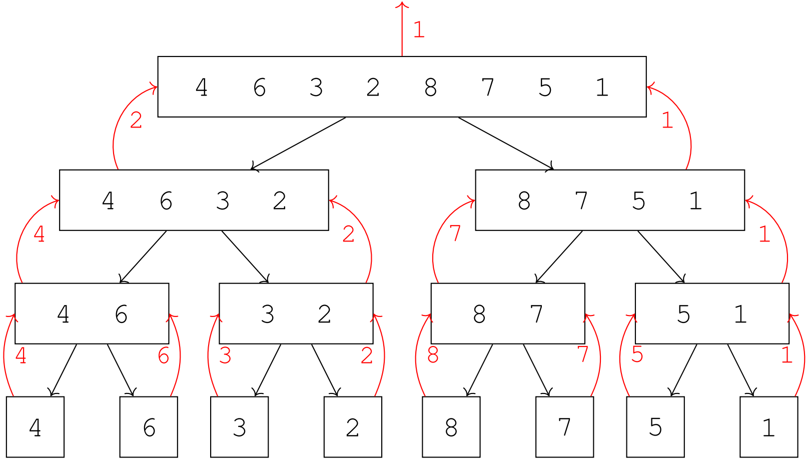
We can call the following algorithm by RecMin(A, 1, n)

```
Algorithm RecMin(A[ ], p, q)
    if p == q then
        return A[p]
    else begin
        answer1 ← RecMin(A, p, (p+q-1)/2)
        answer2 ← RecMin(A, (p+q+1)/2, q)
        if answer1 ≤ answer2 then
            return answer1
        else return answer2
    end
```

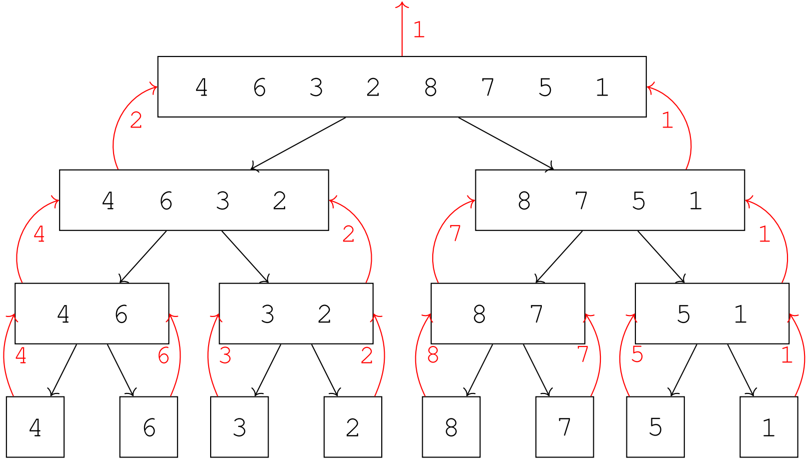
Iterative version:

```
min ← A[1]
i ← 2
while i ≤ n do
begin
    if A[i] ≤ min then
        min ← A[i]
    i ← i + 1
end
output min
```

Time complexity analysis



Time complexity analysis



$O(n)$ Why?

Summary

Summary: Basic Divide-and-Conquer algorithms

Next: Merge Sort Algorithm

For note taking