Searching

TEACHING TEAM

ALGORITHM AND DATA STRUCTURE

2022/2023

Outlines

- Sequential Search
- Binary Search
- Pengayaan: Merge Sort

Searching

- When we are working with data collection, we probably need to **look for** or **search** a data from the collection. In other word we may need to retrieve a data from it by certain condition.
- **Searching** is a process to find data from the collection based on the searching criteria.
- The most important step in searching is checking whether the data matches with the searching criteria
- Searching algorithm is an algorithm depicts step by step process to perform a searching process.

Searching

The results or output of the searching process might be vary

- Message
  - Found / Available
  - Not found / Nothing
- Index array
  - index = 13
  - i = 7
  - idx = -1 (if the data in the search can not be found)
- Boolean Value
  - TRUE
  - FALSE

Searching

- Kind of searching algorithm:
  - Sequantial Search
  - Binary Search

Sequential Search

- Sequential search is usually known as Linear Search
- It is the simplest method
- It could be implemented to a random data
- Basic concept:
  - The data is compared one by one with keywords, in sequence (starting the first data until the data that matches the keywords)

Sequential Search

- There are 2 possibilities in Sequensial Search:
  - The best case: if the data that we need to find is located in the leading array index (first array element), so that the time needed to search data is very fast (minimum)
  - The worst case: if the data that we need to find is located in the last array index (the last array element), so that the time needed to search the data is very long (maximum).

Sequential Search

1. Input **keyword** to be searched
2. Start the looping from the **first element** of array (i=0), and compare it with the **keyword**.
3. If the data in element array is similar with the keyword, then return the index where the data is found. The index indicates the location where the data is found
4. If the data in element array does not match with the keyword, then continue to the next loop for the next i (back to step 1) to compare keyword with the data in the next element
5. If until the last element of array, and there is no data that matches with keyword, then return -1. It means that the keyword is not found in the array.

Sequential Search

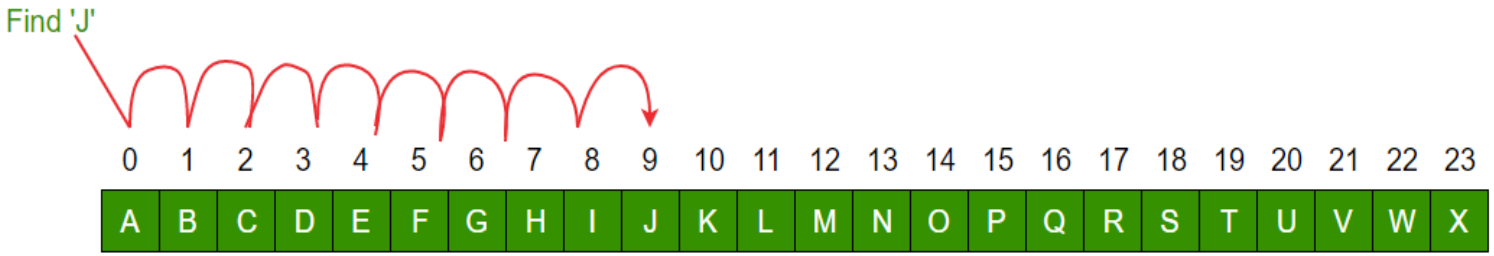
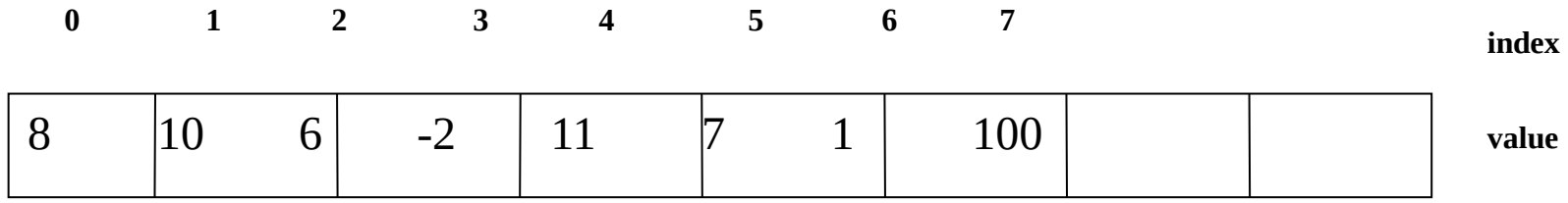


Illustration of Sequential Search

- For example there is a one-dimensional array as follows:



- Then the program will request data to be searched, for example **6 (x = 6)**.
- Iteration :
  6 = 8 (No!)
  6 = 10 (No!)
  6 = 6 (Yes!) => output : "data found" at index to-2
- If until the last data not found the same data then the output: "the data sought does not exist".

Sequential Search

```
//there is an array data[n], the first index is 0 and the last index is n-1, and x is the keyword that we need to find


for i=0 to n-1 step by 1 do
        if data[i] == x then
                return i
        endif
end for

return -1
```

```java
package week6;

public class SequentialSearch {
    public int sequentialSearch (int[] data, int key){
        for(int i=0;i<data.length;i++){
            if(data[i]==key){
                return i;
            }
        }
        return -1;
    }
}
```

```java
package week6;

public class Main {
    public static void main(String[] args) {
        int list[] = {3,45,34,2,76,345,23,6,12};
        int key = 6;

        SequentialSearch ss = new SequentialSearch();
        int position = ss.sequentialSearch(list, key);
        if(position==-1){
            System.out.println("The key is not found");
        }else{
            System.out.println("The key is found at index "+position);
        }
    }
}
```

Advantages

- Data sets do not have to be in order (sequential search can work in a random data)
- If the keyword is located in the front position, then the data will be found quickly (**best case)**
- Insertion and deletion of elements in a data set does not affect the searching process, because the data does not need to be sorted. In other search algorithms, the data must be rearranged after the insertion or deletion of elements
- Is a very simple search algorithm, saving resources and memory

Disadvantages Sequential Search

- If the **keyword** is located in the back or last position, the search process will take a long time (**worst case)**
- Computer load will increase if the amount of data in the array is very large, so it is not suitable for large data

Binary Search

- Basic concept → data is divided into two parts and we will perform the searching process in a correct part
- Binary Search is implemented on the sorted data
- Algorithm:
  - The data is already sorted
  - Initialize **begin = 0** dan **end = n-1**
  - Then find position of **middle = (begin+end)/2**
  - Compare the key with the data in the element array at index **middle**
    - If the key is smaller than the data in the **middle**, then the searching will continue with the left side data. It means that the position of **end** will be updated to **middle-1**
    - If the key is bigger than the data in the **middle**, then the searching will continue with the right side data. It means that the position of **begin** will be updated to **middle+1**
  - The binary search will be finished when **the keyword is found** or the value of **begin>end**

Binary Search

Step 1

3	9	11	12	15	17	20	23	31	35

begin **middle** **end**

begin = 0, end = 9, middle = (0+9)/2 = 4

- The key that we need to search is 17
- **Compare** 17 with the data at **middle** index.
- Since the key is **bigger** than the data at **middle** index, then the new **begin** = **middle+1** = 5, **end** = 9, **middle** = (5+9)/2 = 7

Step 2

3	9	11	12	15	17	20	23	31	35

begin **middle** **end**

- **Compare** 17 with the data at **middle** index (23).
- Since the key is **smaller** than the data at **middle** index, then the new **end** = **middle-1** = 6, **begin** = 5, **middle** = (6+5)/2 = 5

Step 3

3	9	11	12	15	17	20	23	31	35

begin=middle end

- Compare 17 with the data at **middle** index.
- Since the data at middle index is the same with the key (17), then the searching process is finished

Binary Search

```
//there is an array data[n], the first index is 0 and the last index is n-1, and x is the key
that we need to find


begin=0
end=n-1
while begin <= end do
        middle = (begin+end)/2
        if data[middle] == x then
                return middle
        else if x > data[middle] then
                begin = middle+1
        else
                end = middle-1
        end if
end while
return -1
```

```java
public class BinarySearch {
    public void bubbleSort(int data[]){
        for(int i=0;i<data.length-1;i++){
            for(int j=0;j<data.length-1-i;j++){
                if(data[j]>data[j+1]){
                    int tmp = data[j];
                    data[j] = data[j+1];
                    data[j+1] = tmp;
                }
            }
        }
    }
    public int binarySearch(int data[], int key){
        bubbleSort(data);//sort in ascending mode the data
        int begin=0, end=data.length-1;
        int middle =0;
        while(begin<=end){ //loop until the key is found or the begin>end
            middle = (begin+end)/2; //calculate middle index
            if(data[middle]==key){//the key is found
                return middle;
            }else if(key>data[middle]){
                begin = middle+1;//shift begin
            }else{
                end = middle-1;//shift end
            }
        }
        return -1;//if this statement is executed, then the key is not found
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        int list[] = {3,45,34,2,76,345,23,6,12};
        int key = 6;

        BinarySearch bs = new BinarySearch();
        int position = bs.binarySearch(list, key);
        if(position==-1){
            System.out.println("The key is not found");
        }else{
            System.out.println("The key is found at index "+position);
        }
    }
}
```

Best & Worst Case

- **Best case** : if the data sought is located in the middle position.
- **Worst case** : if the data sought is not found.
- Example :

  DATA = 5 6 9 2 8 1 7 4 3

  bestcase when x = 5 (T(n)=1)

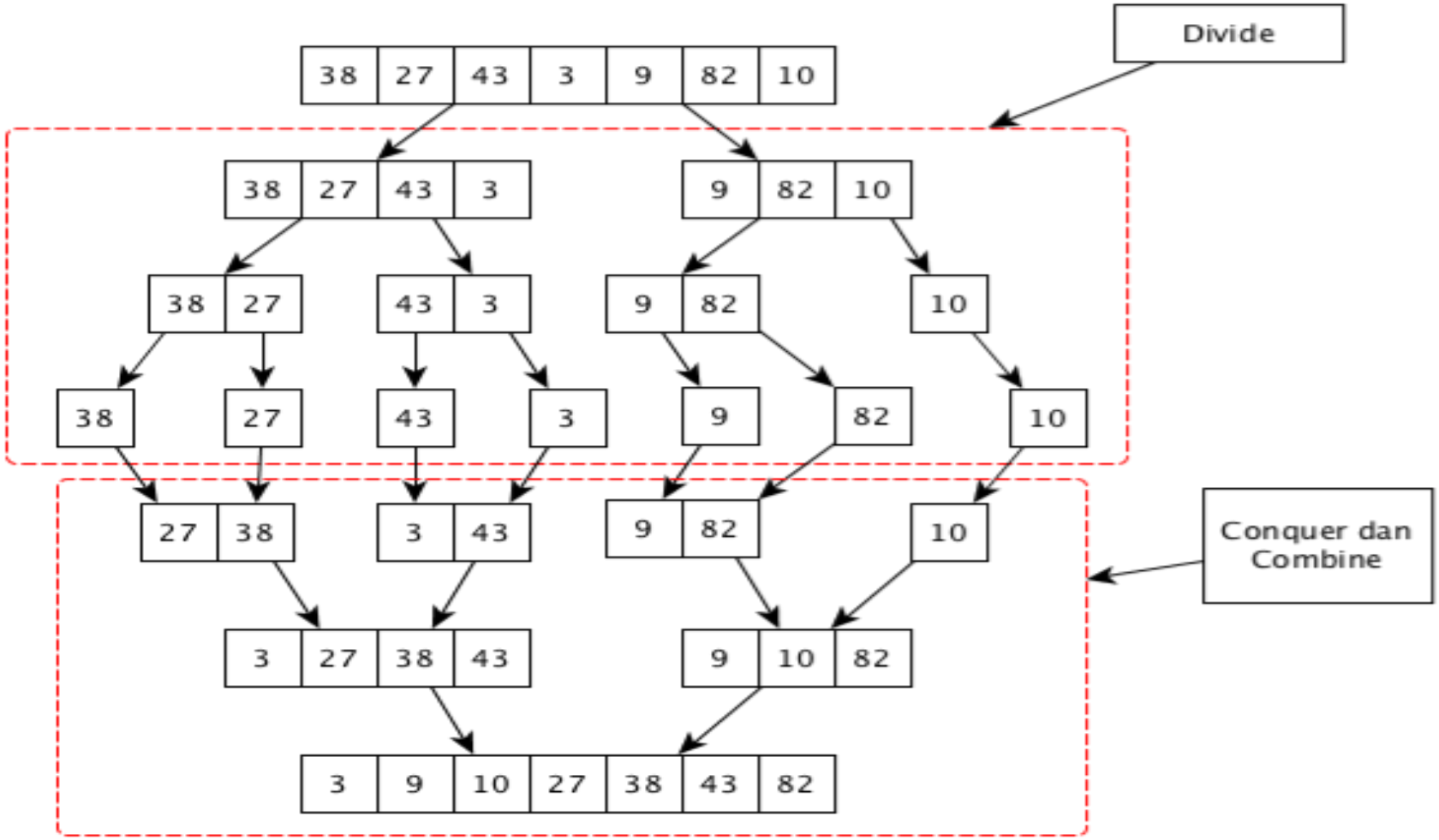
  worstcase when x = 25 (T(n) = 5 or n/2)

  *x = key/the data sought

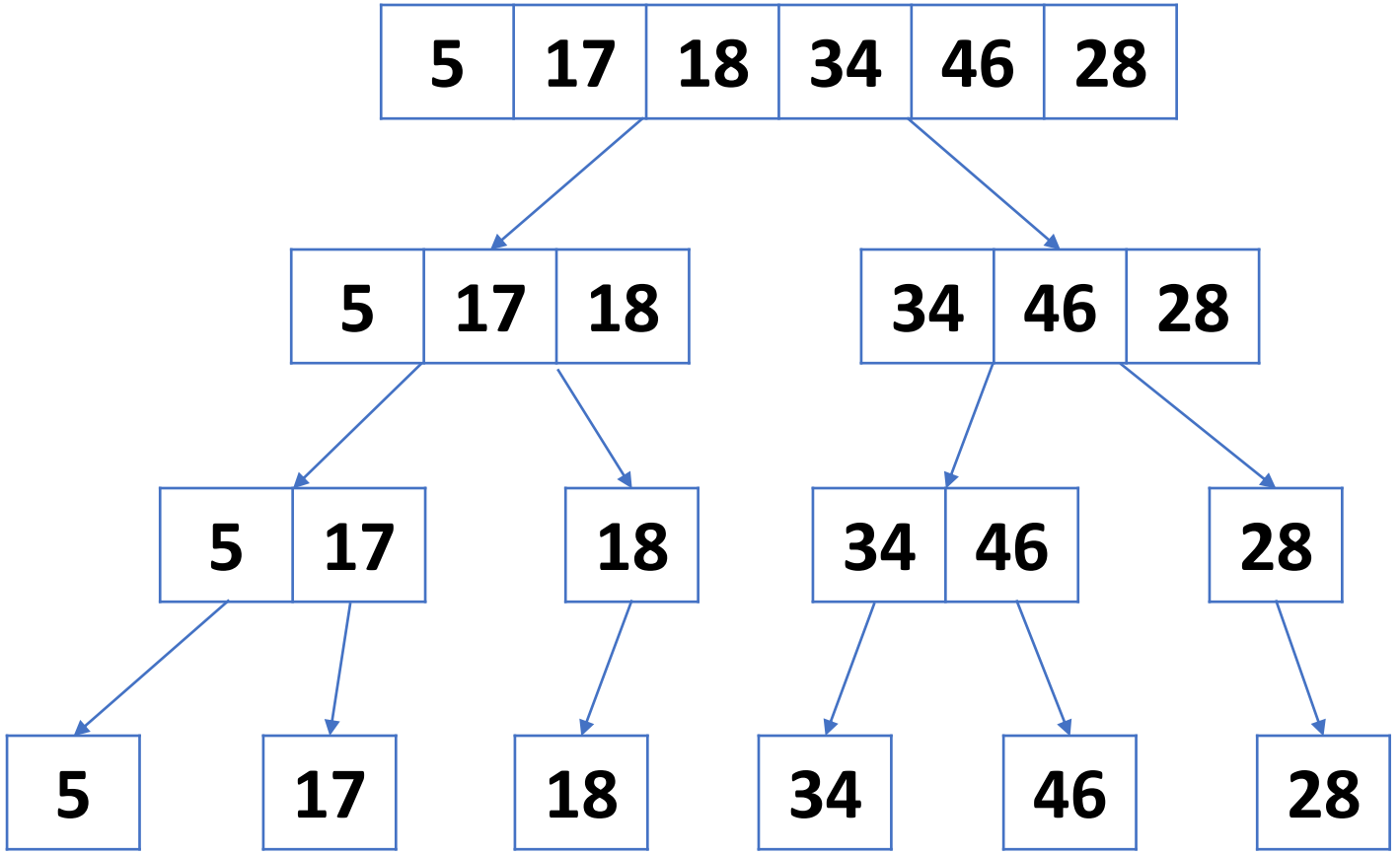
Enrichment Divide n Conquer (Merge Sort)

- Ordering with this method is often also called the Divide and Conquer method.
- This method consists of 3 stages.
  1. Divide divides the problem or collection of data into smaller parts.
  2. Conquer sort from the smallest part.
  3. And the last stage is Combine, combining or combining solutions from the smallest parts to become the main solution.

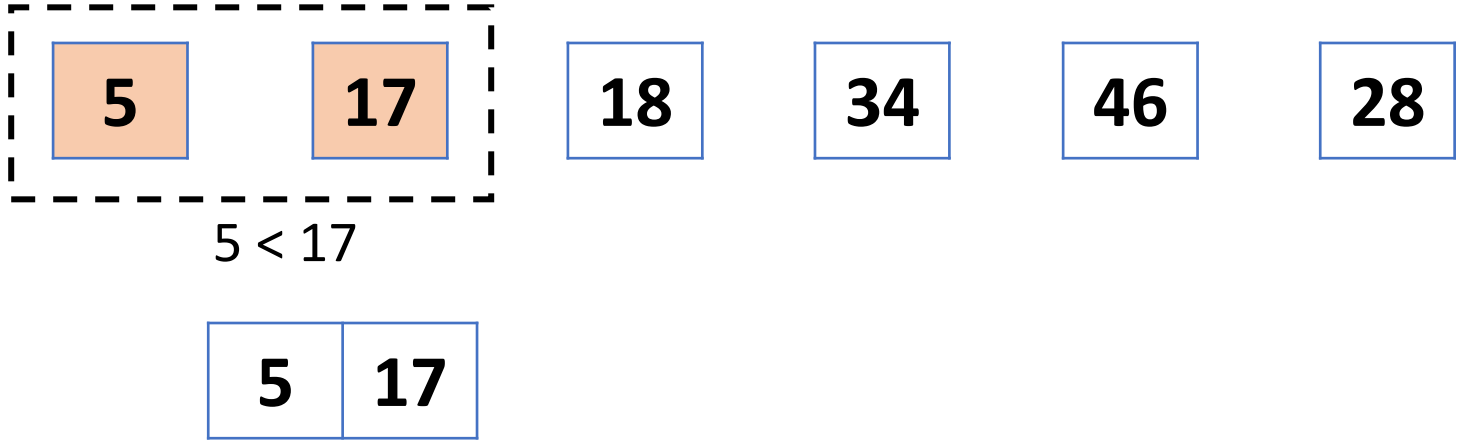
Example 1: Merge Sort (Ascending) – 1



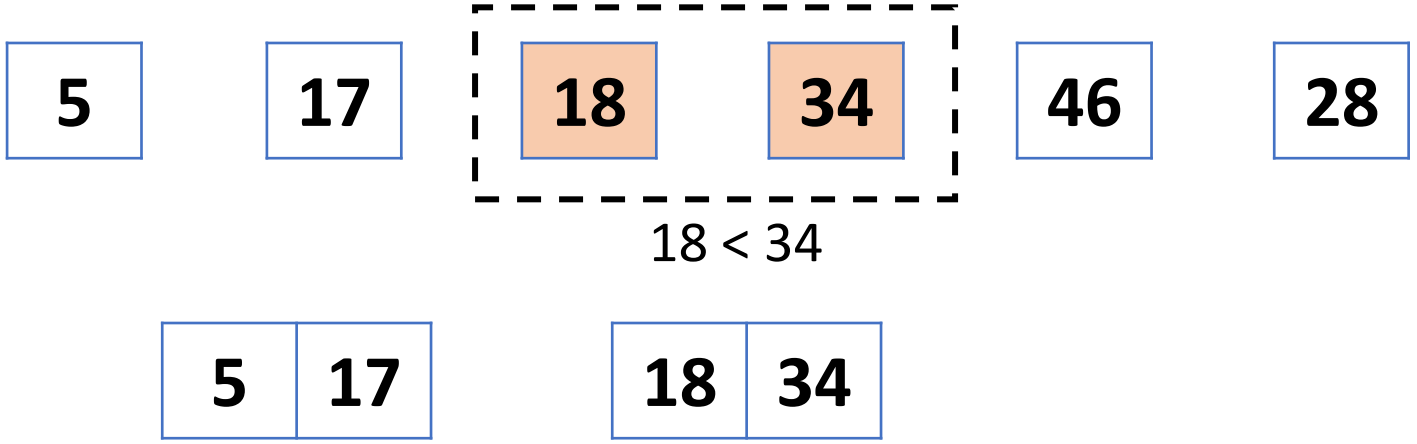
Example 2 - Ascending *Divide*



Example 2 - Ascending *Conquer*

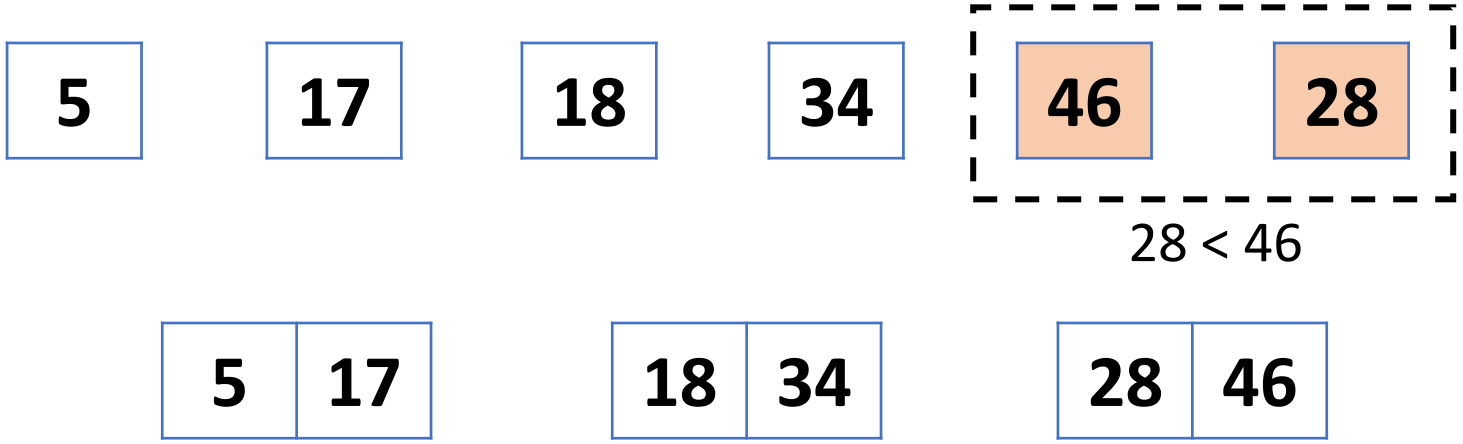


Example 2 - Ascending *Conquer*

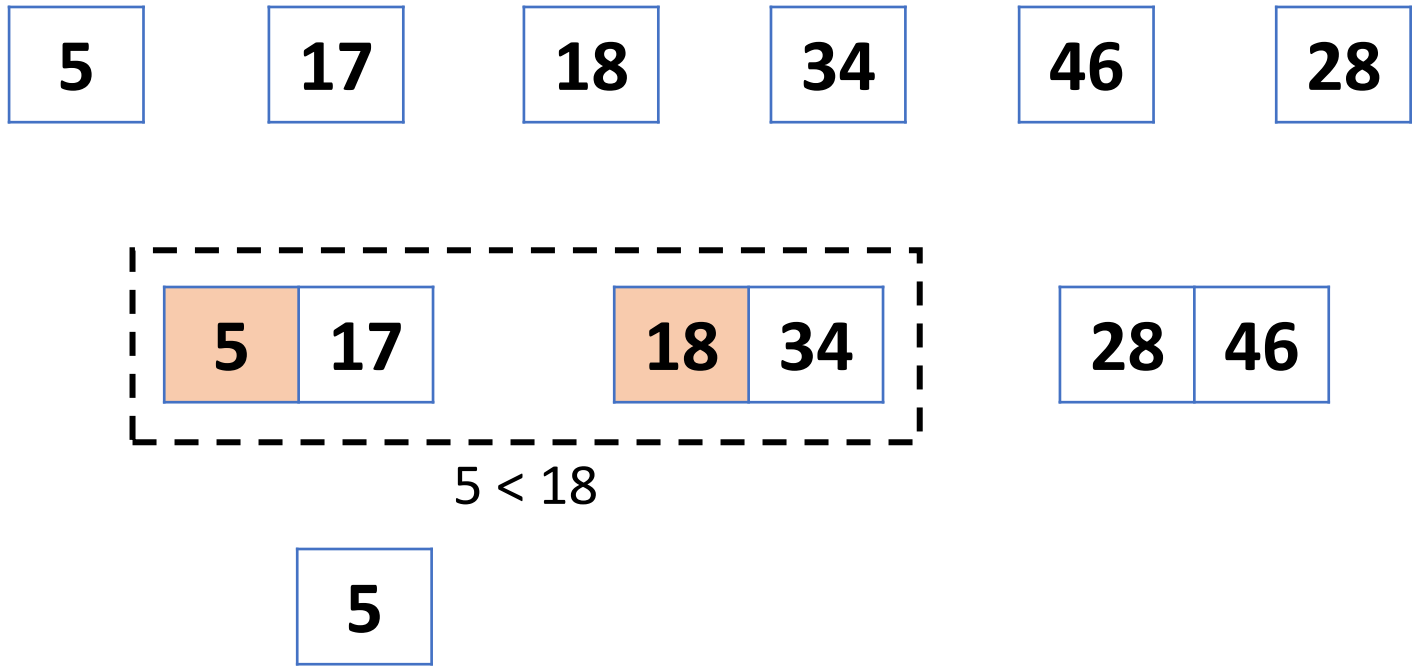


Example 2 - Ascending

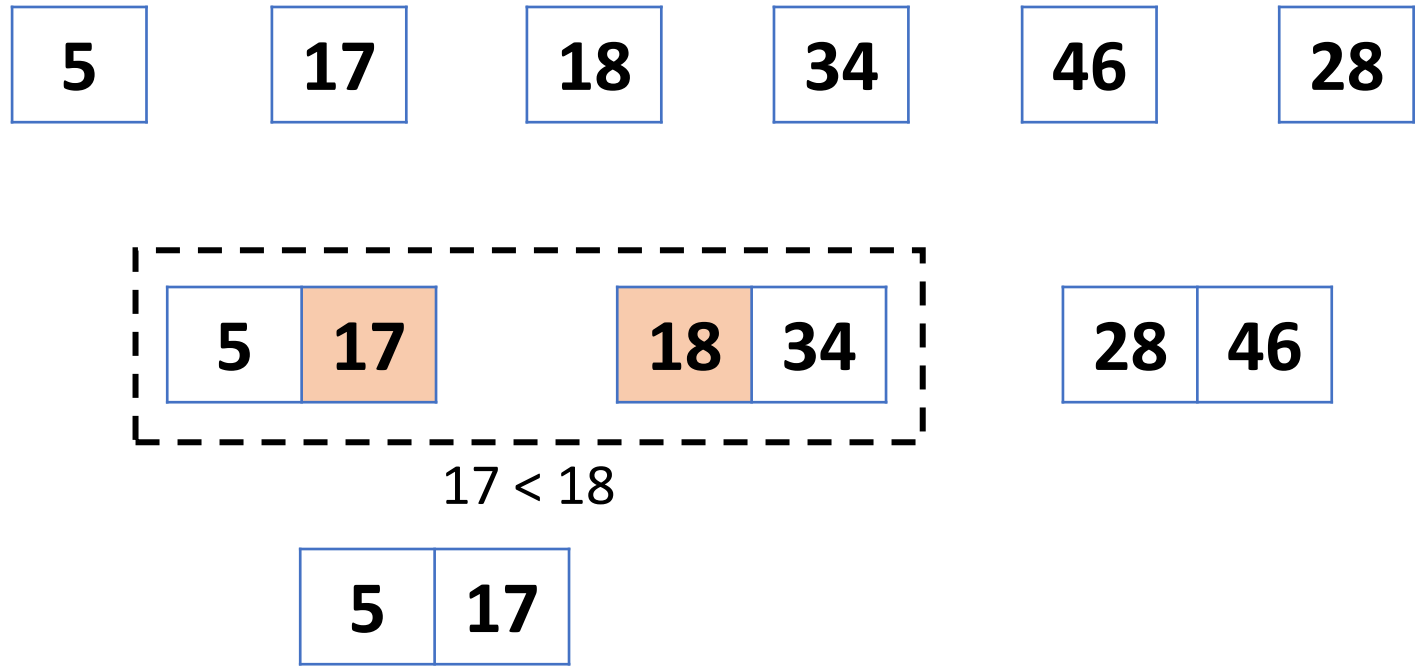
Conquer



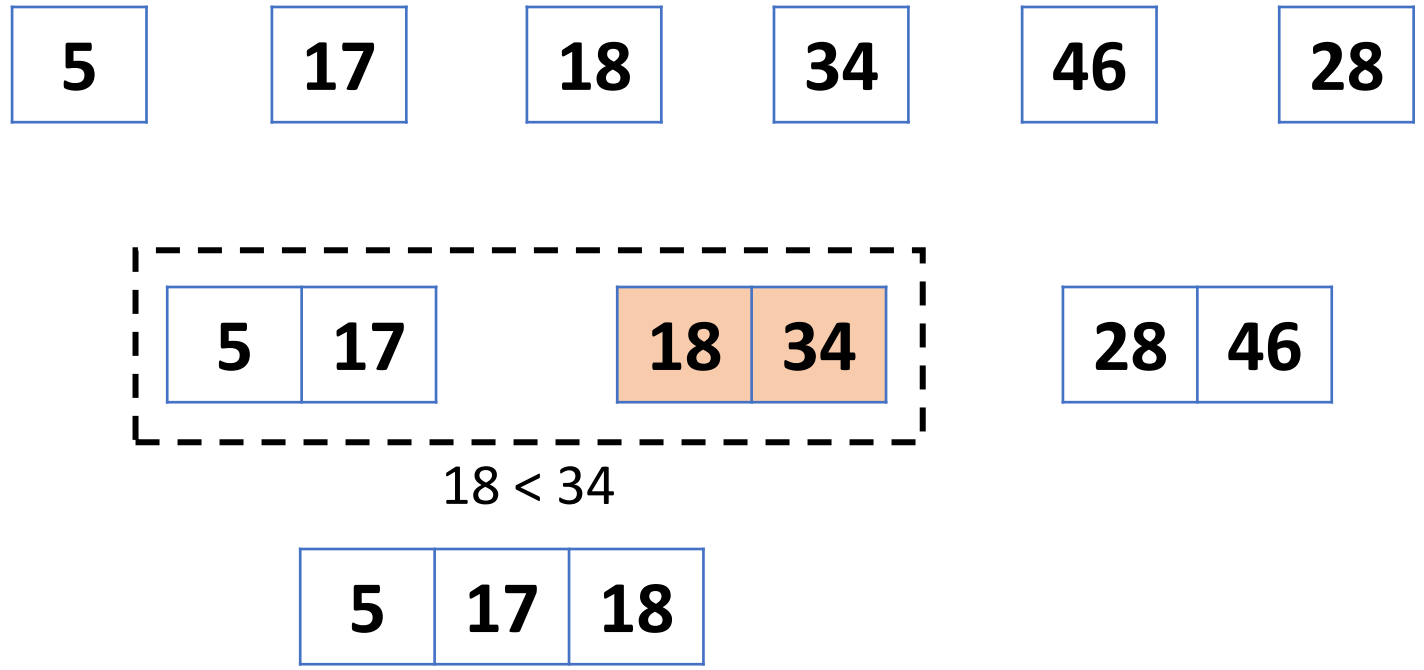
Example 2 - Ascending *Conquer*



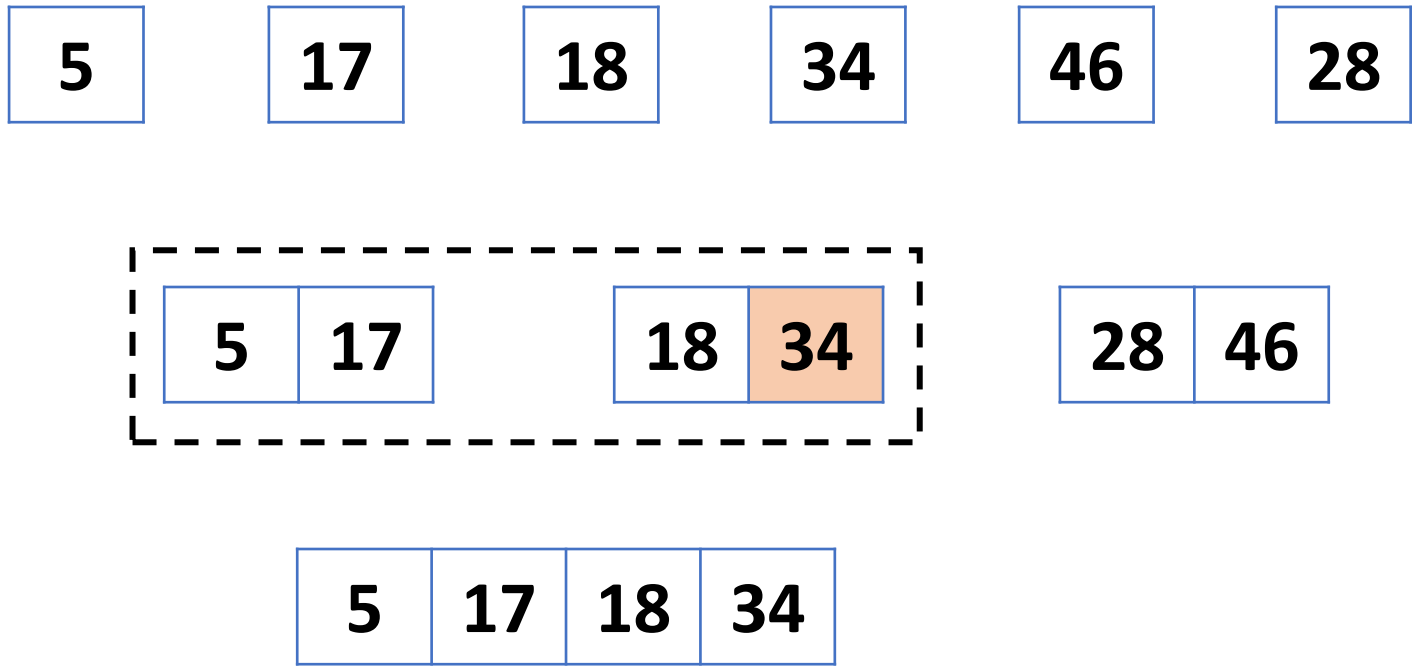
Example 2 - Ascending *Conquer*



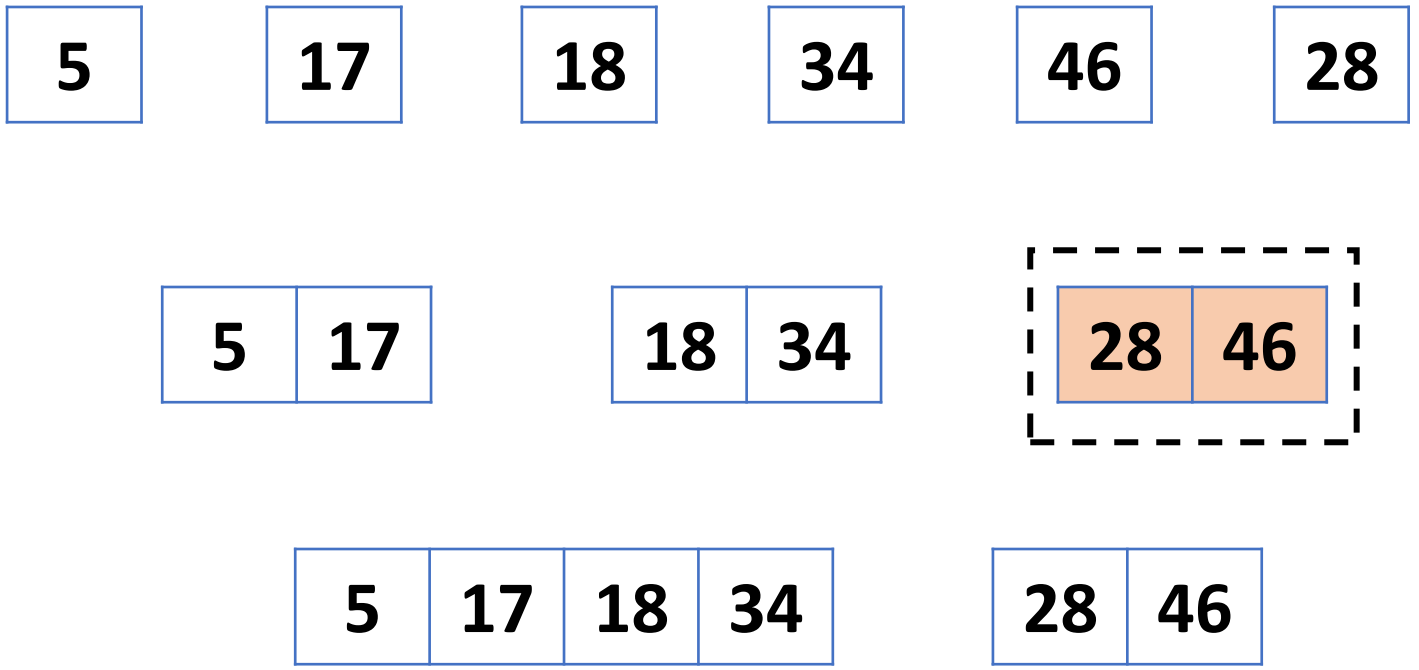
Example 2 - Ascending *Conquer*



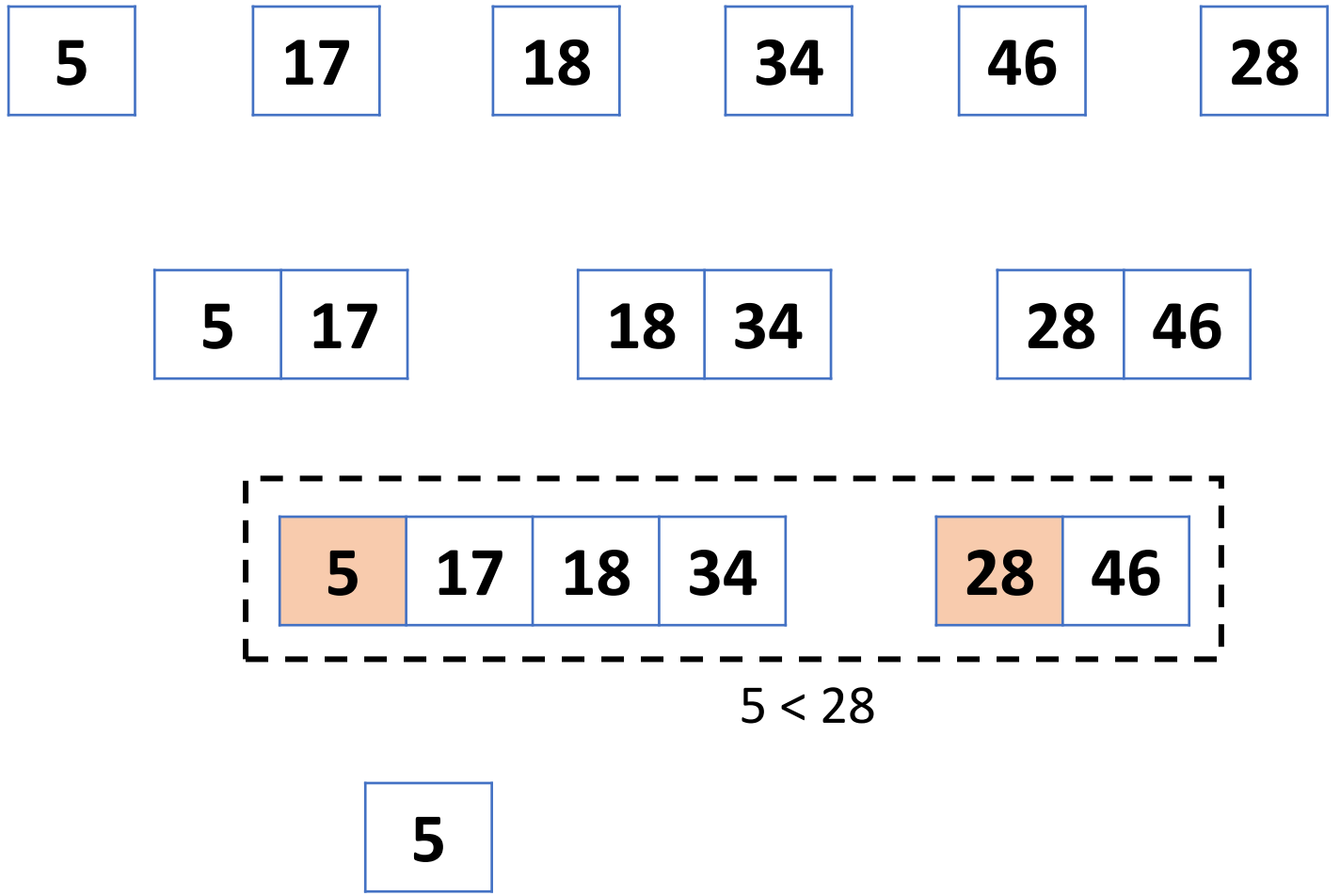
Example 2 - Ascending *Conquer*



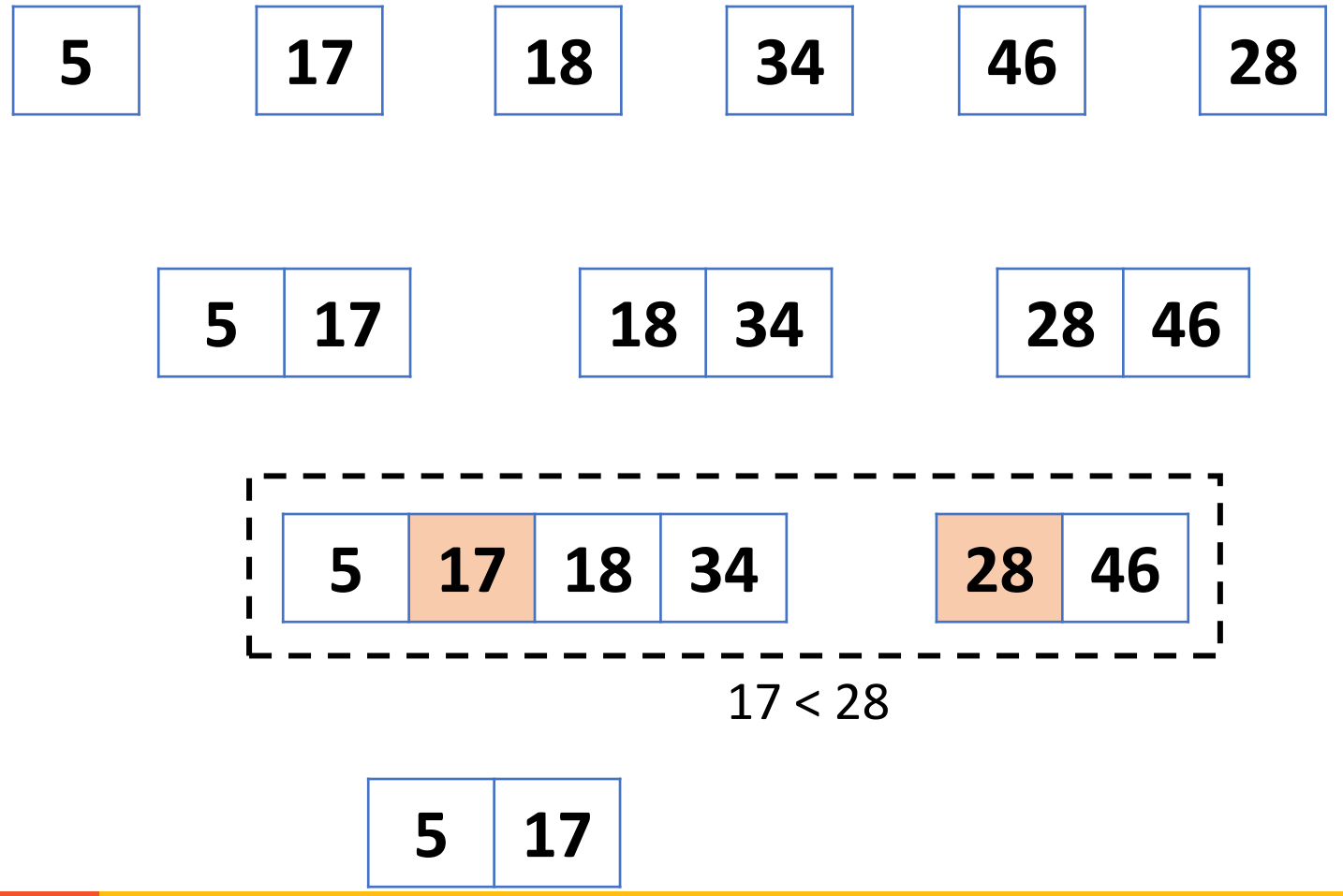
Example 2 - Ascending *Conquer*



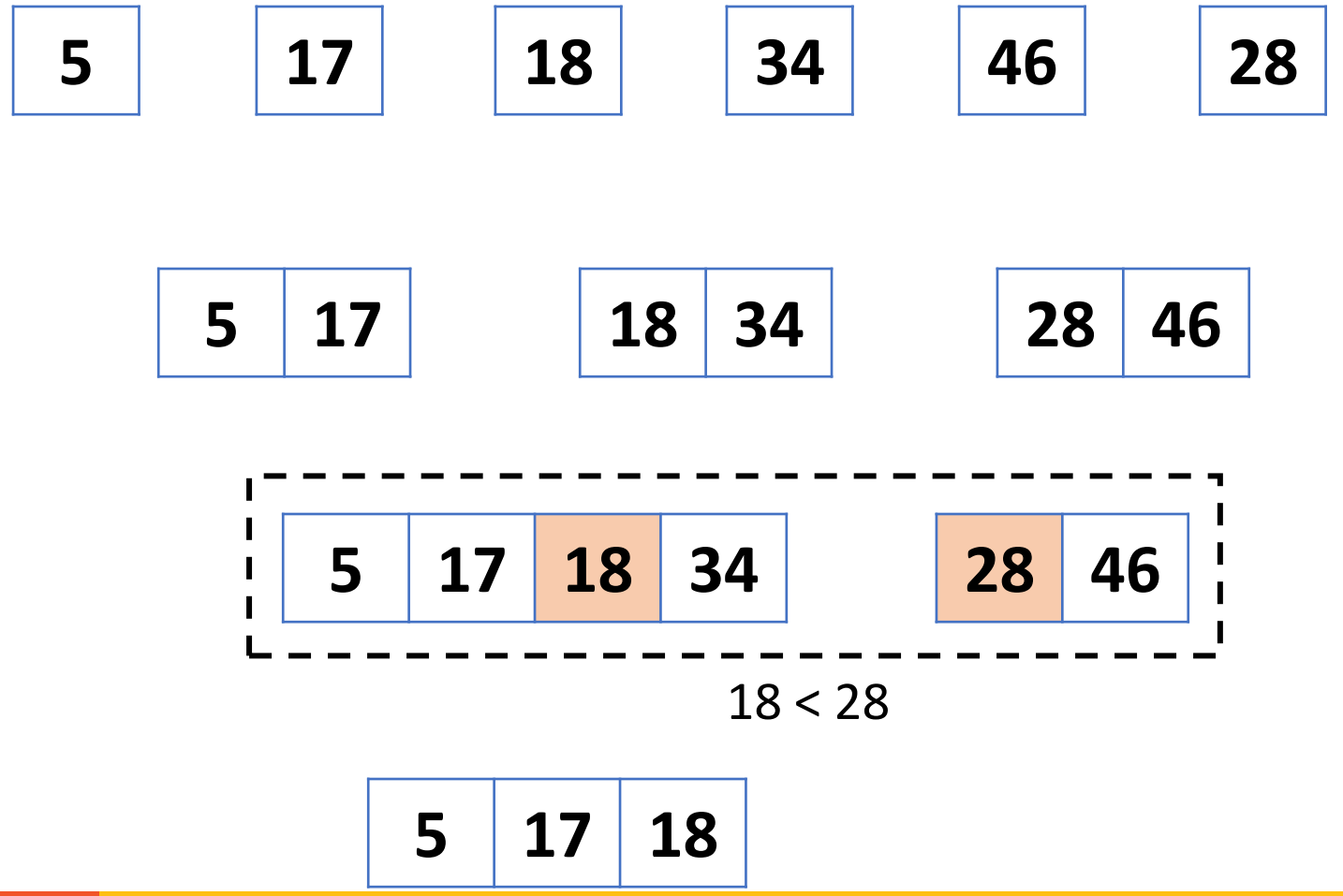
Example 2 - Ascending *Conquer*



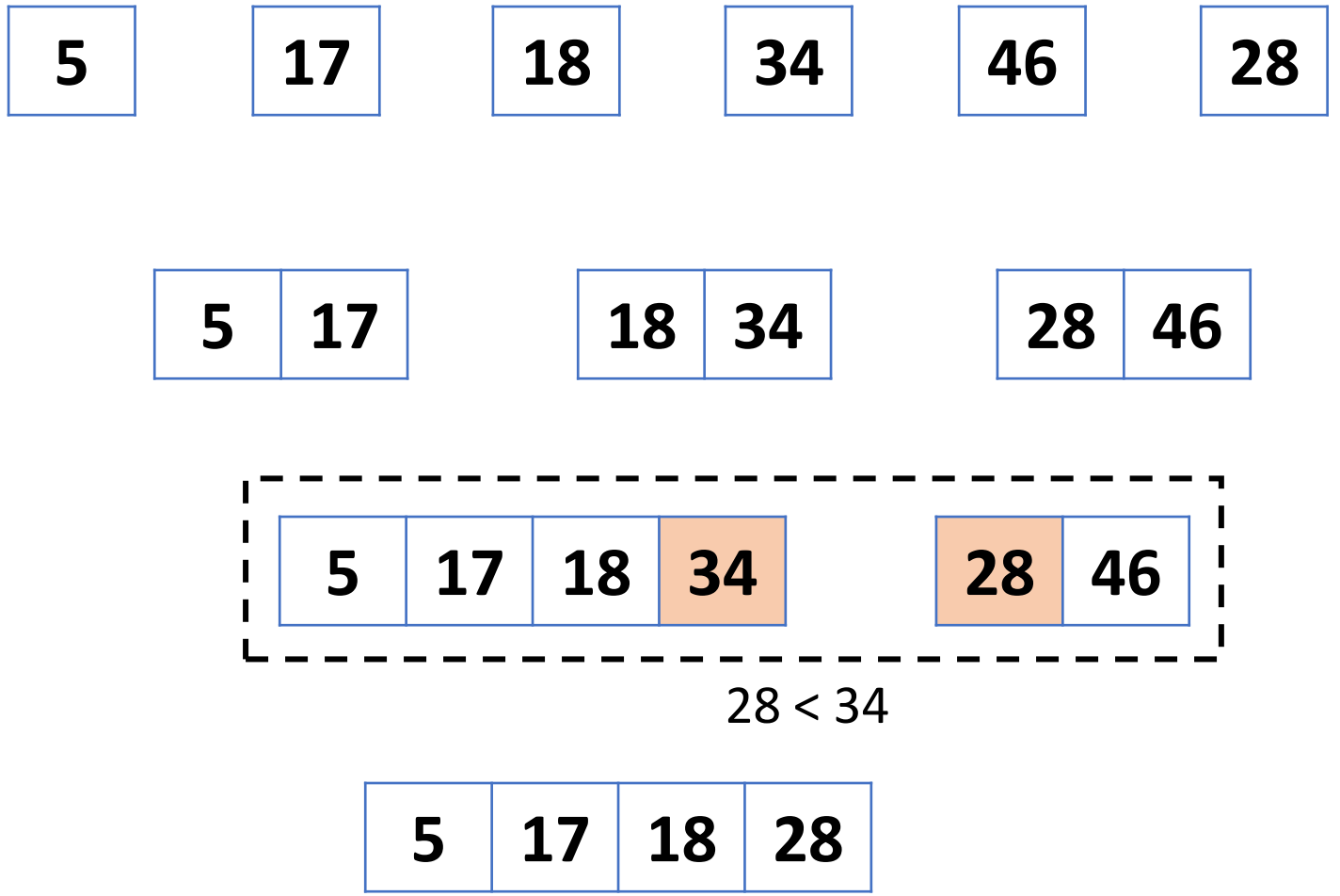
Example 2 - Ascending *Conquer*



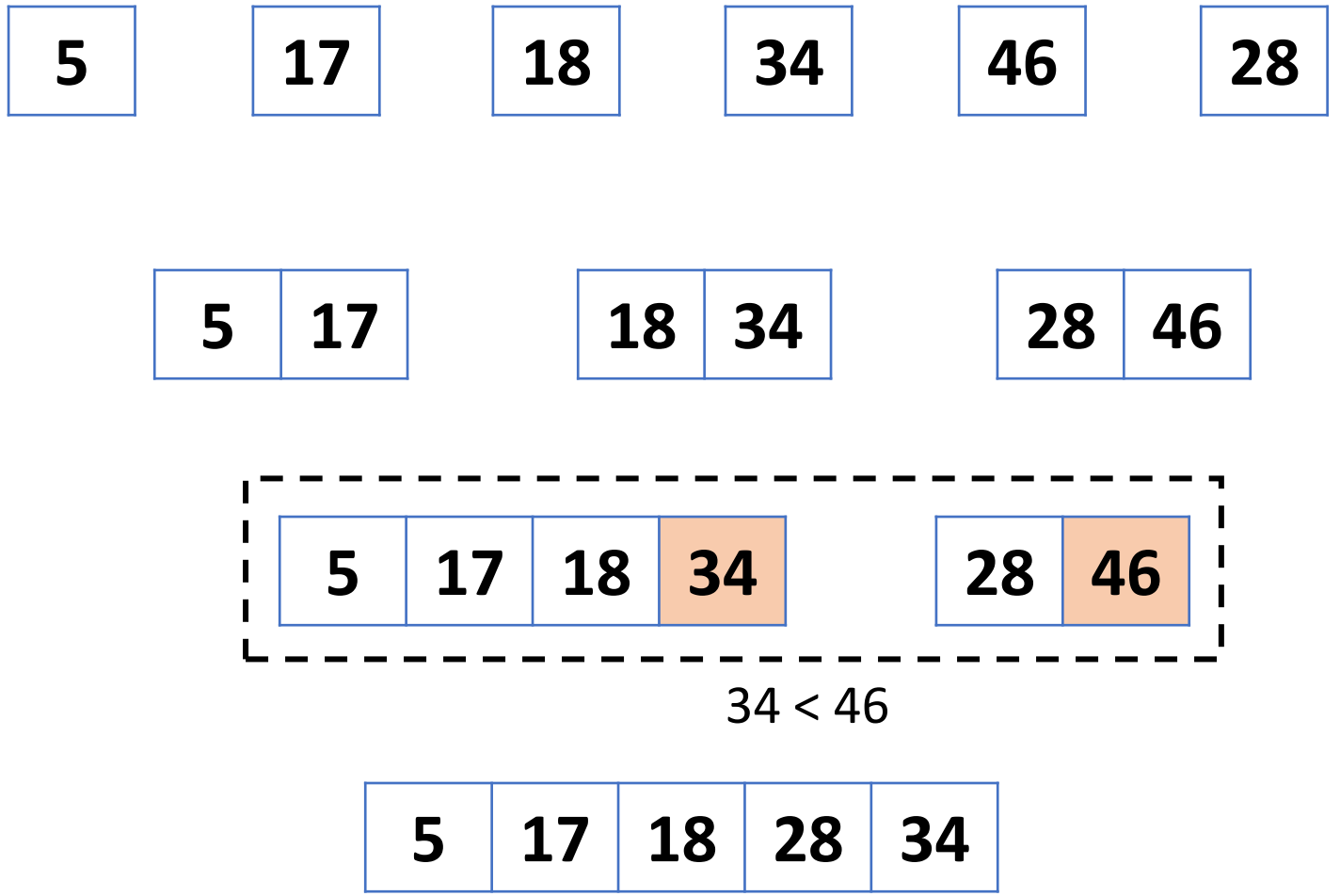
Example 2 - Ascending *Conquer*



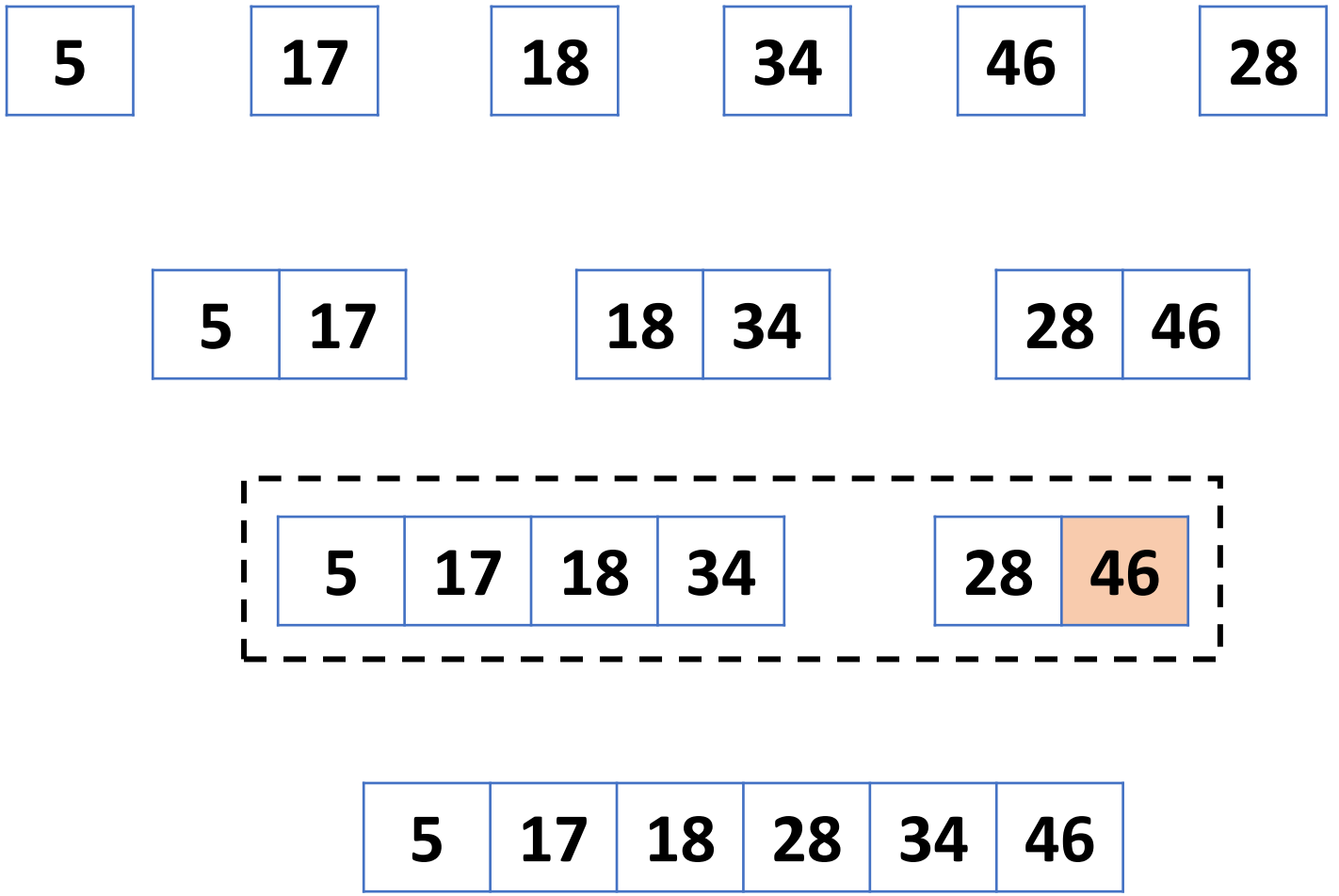
Example 2 - Ascending *Conquer*



Example 2 - Ascending *Conquer*



Example 2 - Ascending *Conquer*



Assignment

1. Make a flowchart for binary search algorithm!
2. Make a flowchart for sequential search algorithm!
3. If there is an array {20,35,14,7,67,89,23,46}
   - Data you want to find (x = 35)
   - Describe the searching process with sequential search
   - Describe the searching process with binary seach (first sort the array with a sorting algorithm)


thank you
danke
謝謝
ngiyabonga
شكراً جزيلاً
рахмат
teşekkür ederim
спасибо
gracias
tapadh leat
Баярлалаа
dank je
bedankt
dziękuję
hvala
mauruuru
obrigado
sukriya
kop khun krap
go raibh maith agat
mochchakkeram
arigatô
dakujem
takk
merci
terima kasih
grazie
sagolun
merci
мерси
감사합니다
ευχαριστώ
xièxie
faafetai lava
kiitos
dankie
vinaka
спасибі
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
tenki
murakoze
chokrane
mamnun
дякую
djiere dieuf
tau
enkosi
bayarlalaa
gràcie
köszönöm
akun
dankon
aciü
chnorakaloutioun
gratias ago
grācies
sulpáy
taiku
sobodi
dēkuji
mesi
didi madloba
najis tuke
kam sah hamnida
rahmat
ありがとう
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
trugarez
তোমাকে ধন্যবাদ
nanni
nandri
dhanyavad
barka
welalin
tack
kia ora
mersi
spas