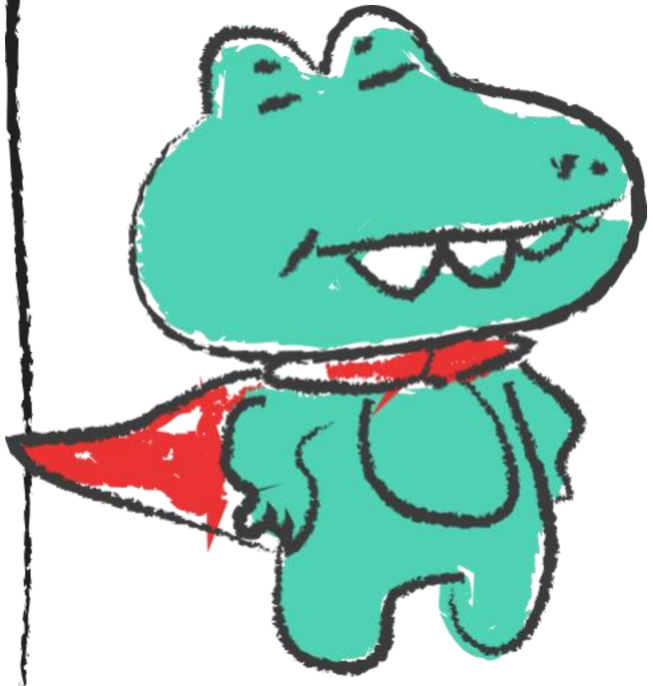
Quick Sort



Smart Device engineering
19011773 Moon_EeSun

Table of Contents

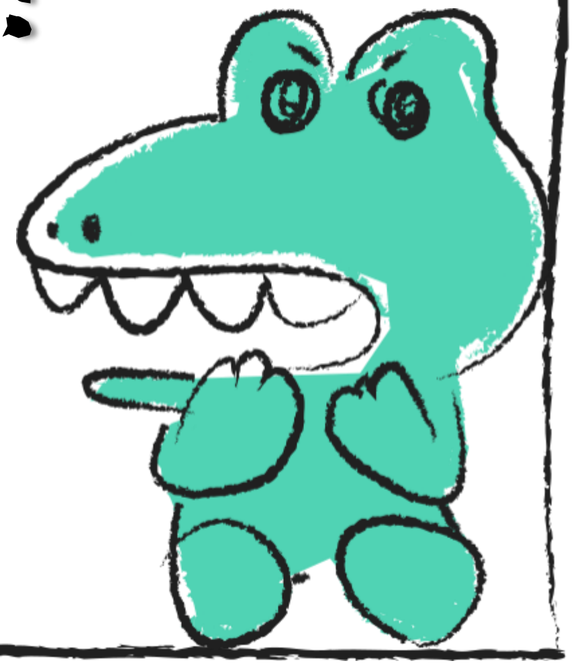
First. Description & Techniques

Second. Example

Third. Pseudocode

Fourth. C-code

What is Quick Sort?



First. Description & Techniques

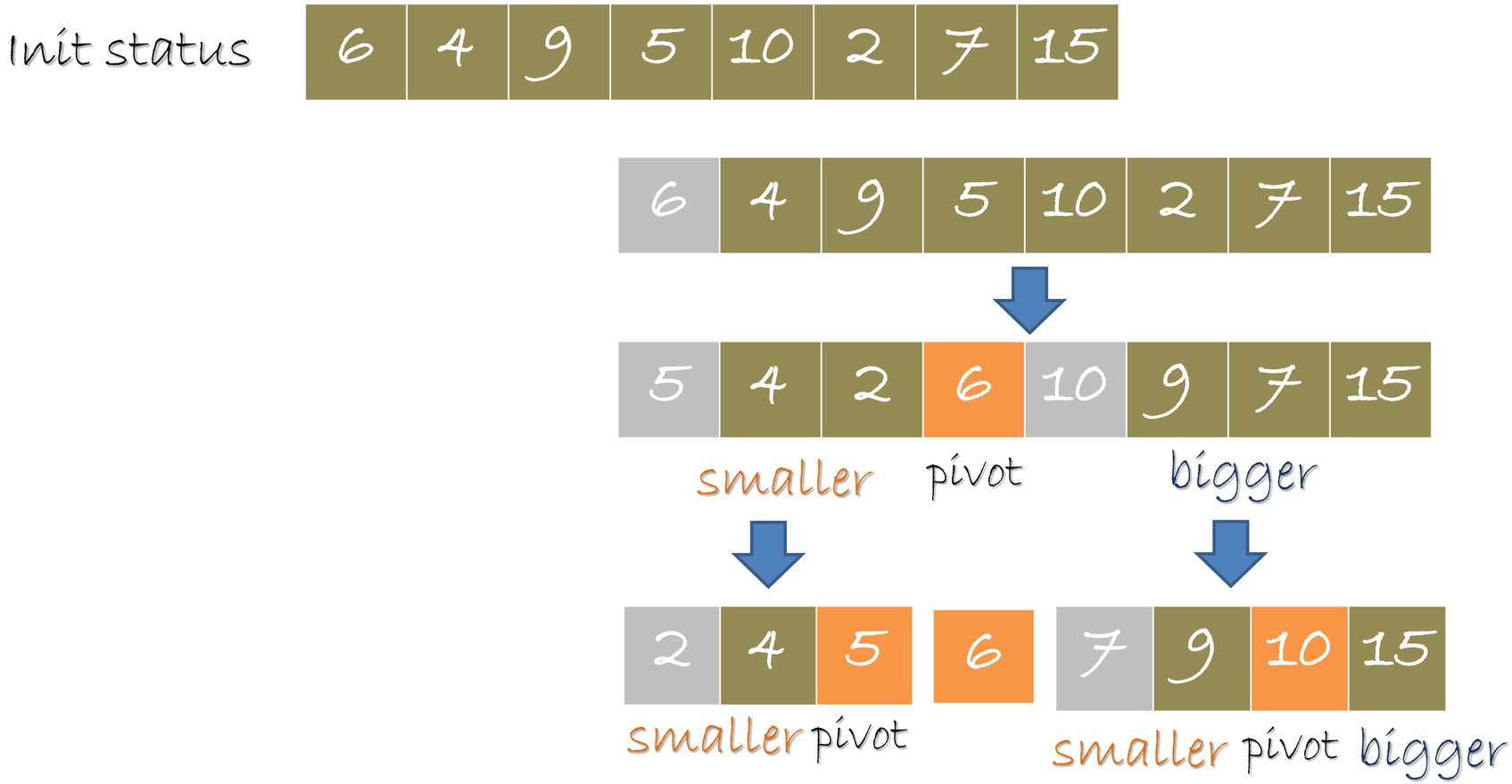
Quick sort : To divide one list into two unequal sizes based on pivots, arrange the segmented list, and then combine the two sorted partial lists to make the whole list sorted.

Divide and Conquer method

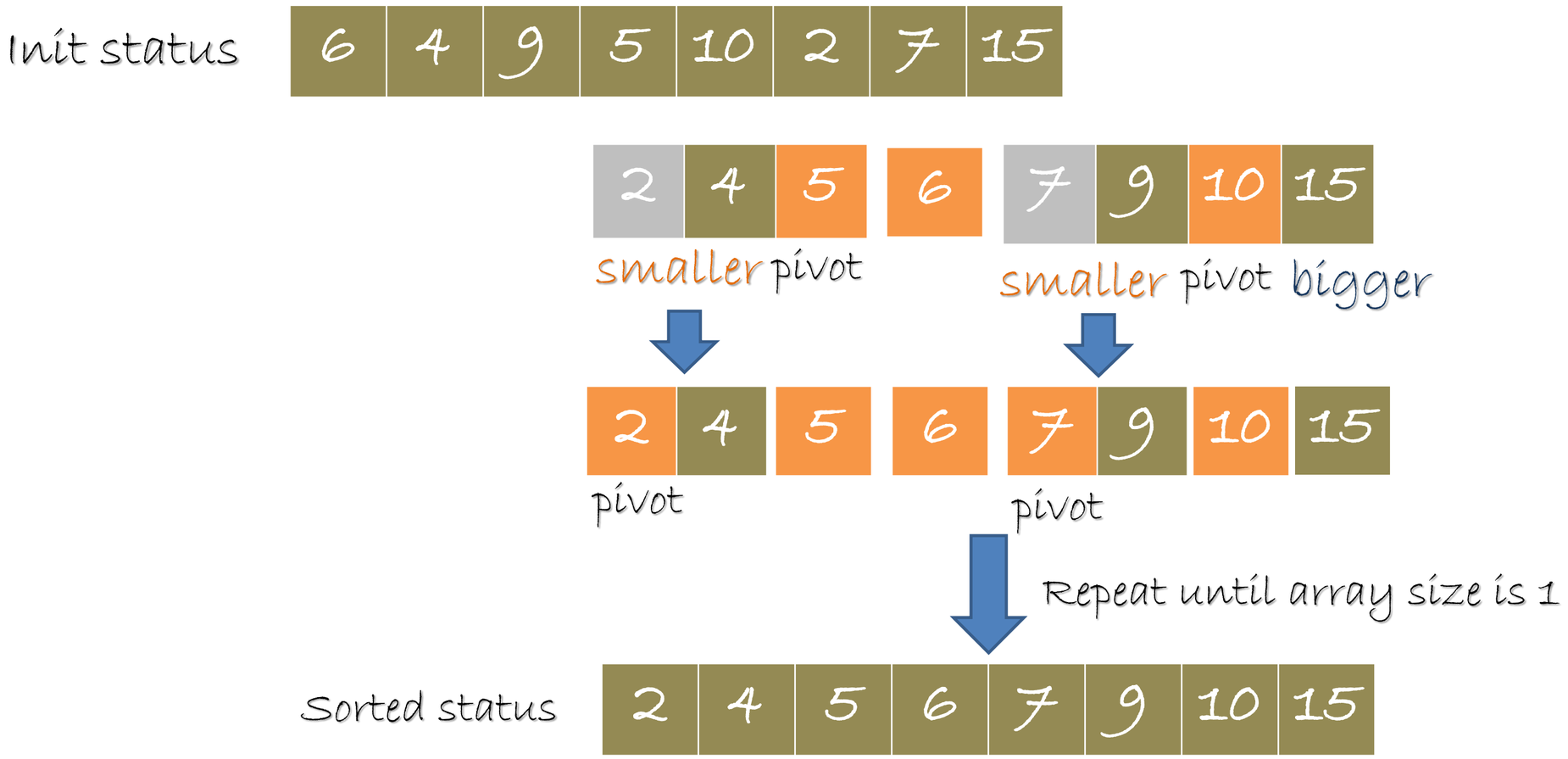
- Divide : Divide the array into two subarrays, based on the pivot
  left - elements smaller than pivot / right - elements larger than pivot
- Conquer : Arranges the sub-arrangement.
- Combine : Merge aligned partial arrays into one array.

Time Complexity : best case - O(nlogn) / worst case - O(n^2)

First. Description & Techniques



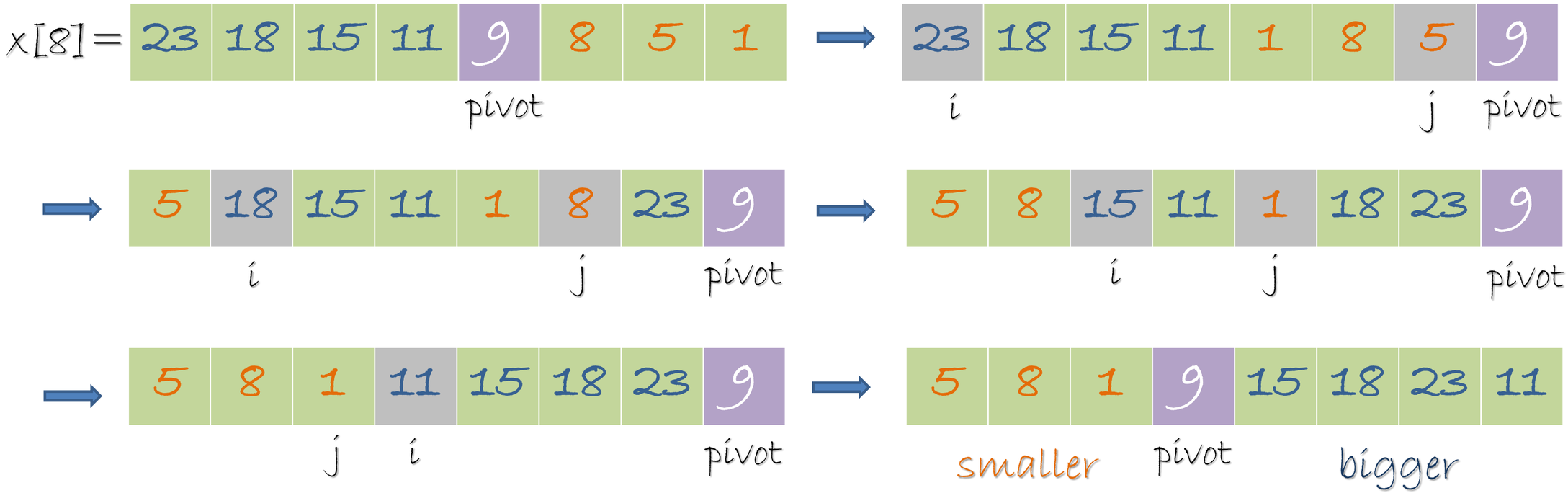
First. Description & Techniques



Second. Example

Example 1. the worst case - Reverse

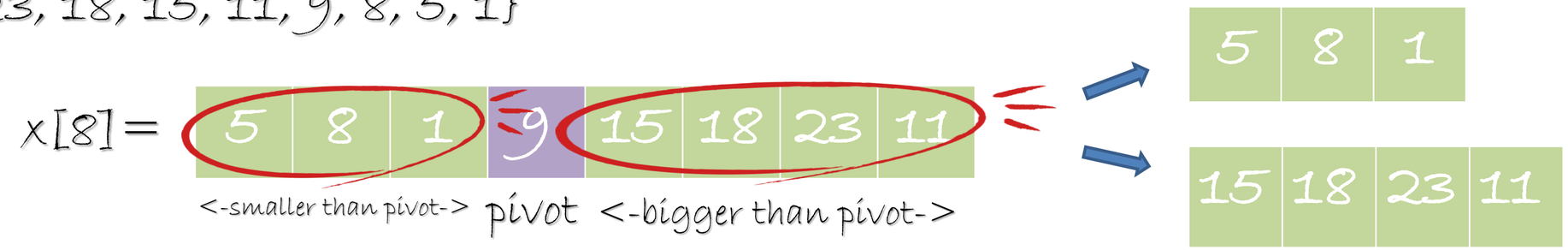
x[8]={23, 18, 15, 11, 9, 8, 5, 1}

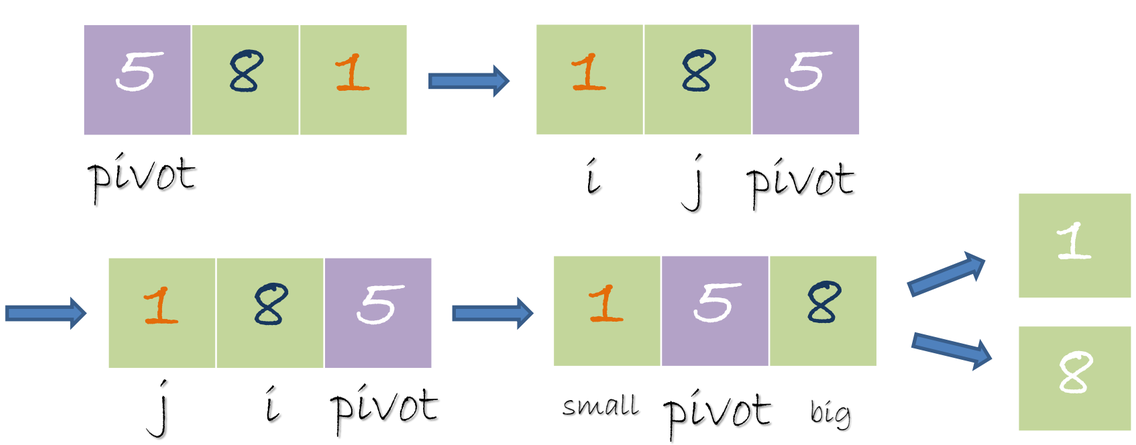


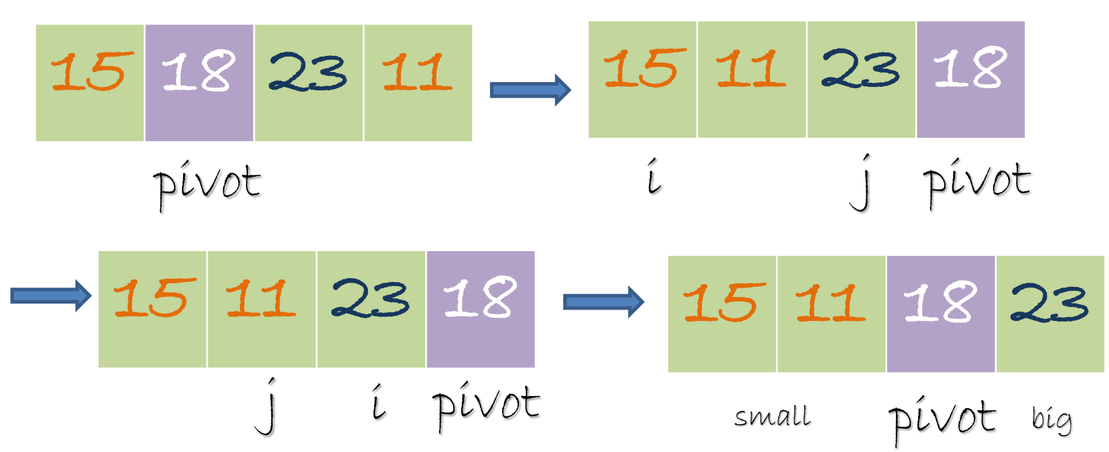
Second. Example

Example 1. the worst case - Reverse

x[8]={23, 18, 15, 11, 9, 8, 5, 1}



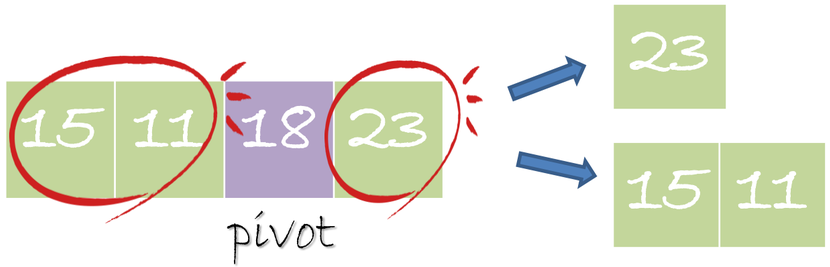




Second. Example

Example 1. the worst case - Reverse

x[8]={23, 18, 15, 11, 9, 8, 5, 1}



Result : 1 5 8 9 11 15 18 23

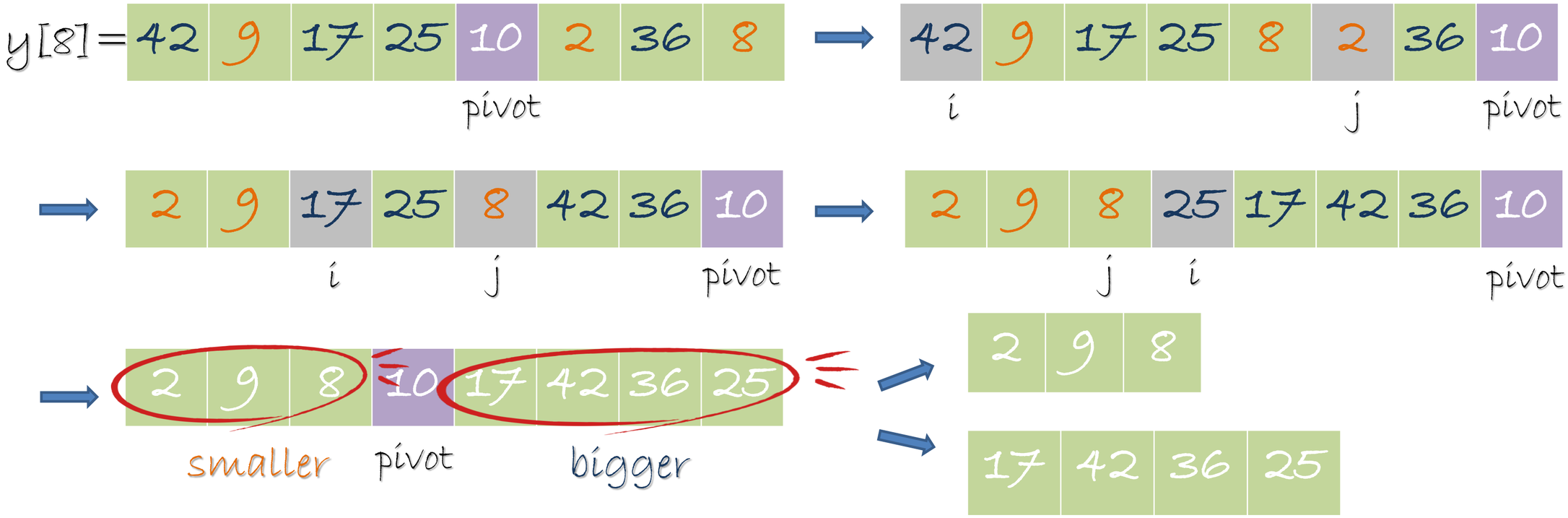
Combine : recursive

1 5 8 9 11 15 18 23 = 1 5 8 9 11 15 18 23

Second. Example

Example 2.

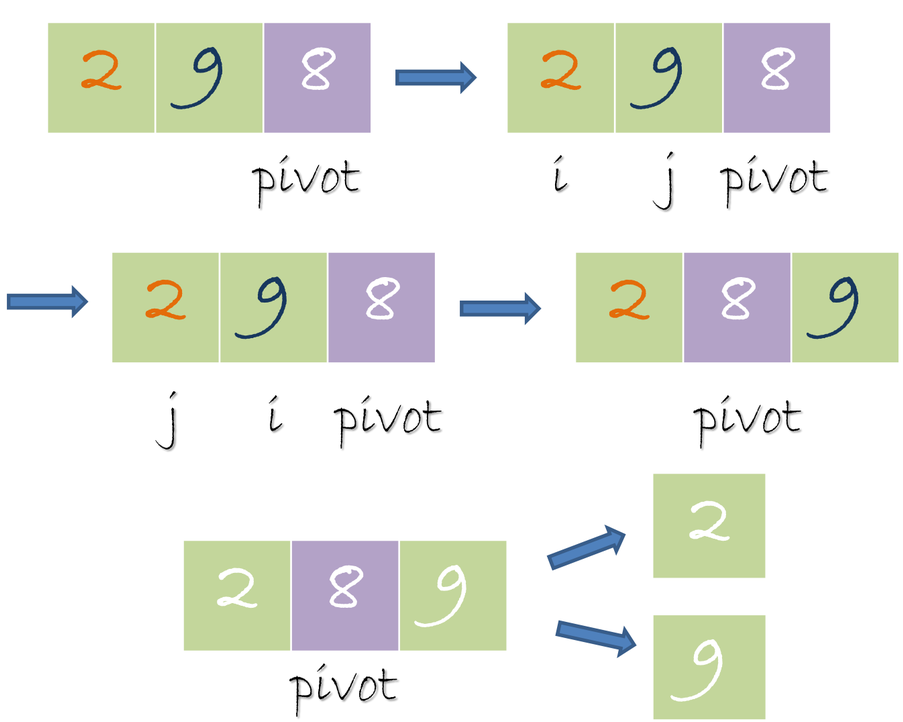
y[8]={42, 9, 17, 25, 10, 2, 36, 8}

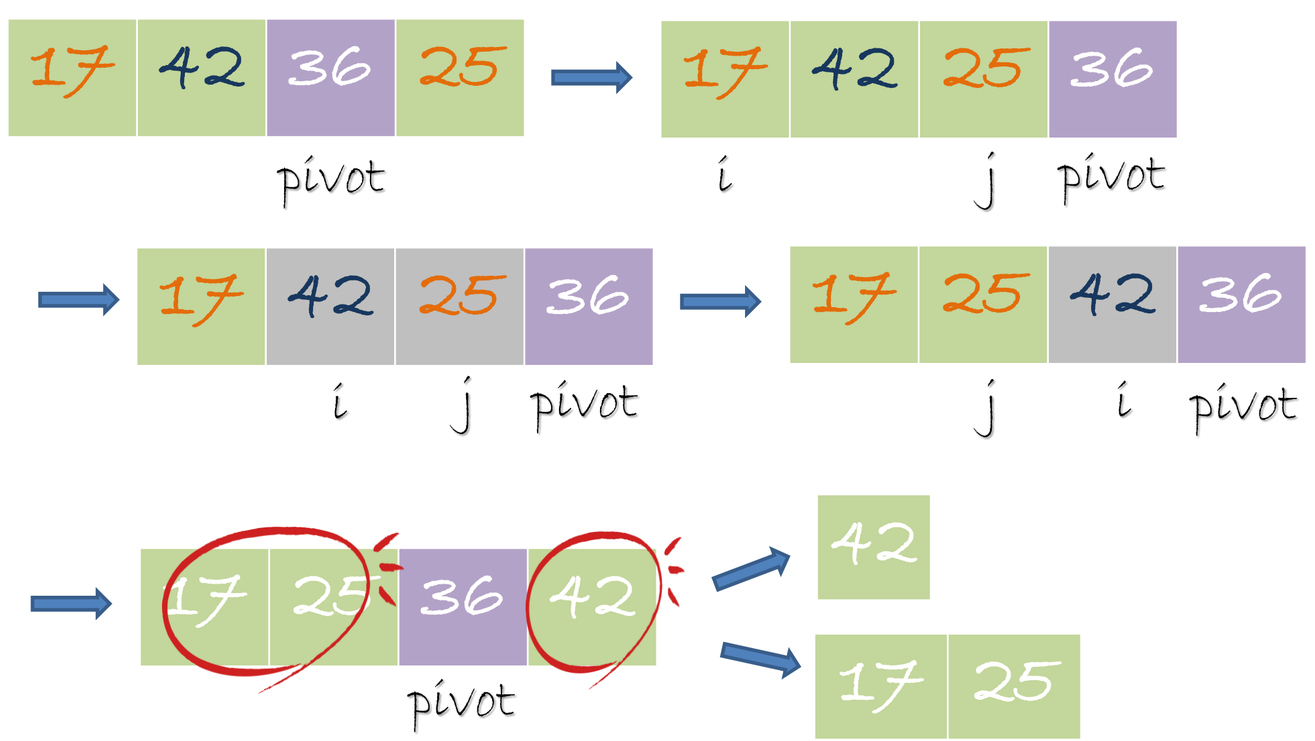


Second. Example

Example 2.

y[8]={42, 9, 17, 25, 10, 2, 36, 8}

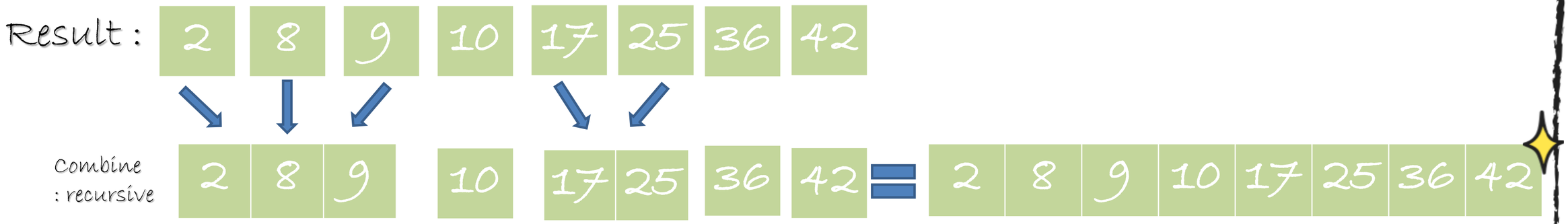




Second. Example

Example 2.

y[8]={42, 9, 17, 25, 10, 2, 36, 8}



Third. Pseudocode

Fuction

- Swap(int Q[], int a, int b)
  : Change ath index element and bth index element
- Median(int Q[], int a, int b, int c)
  : To select a pivot, choose the middle value of the three numbers
- Find_pivot_index(int Q[], int l, int r)
  : choose three random numbers, and choose the middle of the three random numbers
- inPlacePartition(Q, l, r, k)
  : Function to sort by pivot
- inPlaceQuickSort(Q, l, r)
  : Quick sort algorithm
- Main(void)
  : Enters an array of size N and executes a quick sort

Third. Pseudocode

- Swap(int Q[], int a, int b)

  input array Q, index a, b

  output array Q changed ath index element and bth index element

  ```
  tmp <- Q[a]
  Q[a] <- Q[b]
  Q[b] <- tmp
  ```

- Median(int Q[], int a, int b, int c)

  input array Q, index a, b, c

  output index of the middle values of Q[a], Q[b], Q[c]

Third. Pseudocode

- Find_pivot_index(Q, l, r)

  input array of Q, index l, r
  output index of the middle values of p1, p2, p3

  p1 <- random number from l to r
  p2 <- random number from l to r
  p3 <- random number from l to r

  pivot_index <- median(Q, p1, p2, p3)
  return pivot_index

Third. Pseudocode

- inPlacePartition(Q, l, r, k)

    input array Q[l...r] of distinct elements, index l, r, k
    output final index of the pivot resulting from partitioning Q[l...r] into

```
p <- Q[k]
swap(Q, k, r)
i <- l
j <- r-1
while (i<=j)
        while (i<=j & Q[i]<=p) i++
        while(j>=i & Q[j] >=p) j--
swap(Q, i, r)
return i
```

Third. Pseudocode

- inPlaceQuickSort(Q, l, r)

  input array Q, position l, r
  output array Q with elements of position from l to r rearranged in increasing order

```
if (l>=r) return
pivot <- find_pivot_index(Q, l, r)
a, b <- inPlacePartition(Q, l, r, pivot)
inplaceQuickSort(Q, l, a-1)
inplaceQuickSort(Q, b+1, r)
```

- Main(void)

```
N <- input
Q <- malloc
for i=0 to N
        Q[i]<-input
inPlaceQuickSort(Q, 0, N-1)
print(Q, 0, N-1)
```

Fourth. C-code

```c
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int Q[], int a, int b)
{
    int tmp;
    tmp = Q[a];
    Q[a] = Q[b];
    Q[b] = tmp;
}
```

Declare Header File

function swapping with ath element and bth element

```c
int median(int Q[], int a, int b, int c)
{
    int max = Q[a], min = Q[a];
    if (max < Q[b]) max = Q[b];
    if (max < Q[c]) max = Q[c];
    if (min > Q[b]) min = Q[b];
    if (min > Q[c]) min = Q[c];
    if (max == Q[a]) {
        if (min == Q[b]) return c;
        else return b;
    }
    else if (max == Q[b]) {
        if (min == Q[a]) return c;
        else return c;
    }
    else {
        if (min == Q[a]) return b;
        else return a;
    }
}
```

a function that indexes the intermediate elements between the ath element and the bth element and the cth element

Fourth. C-code

a function that randomly selects three pivot indexes and sets the middle value of them as pivot.

```
35  int find_pivot_index(int Q[], int l, int r)
36  {
37      int pivot_index, p1, p2, p3;
38      srand(time(NULL)); //initialize base point with current time
39
40      //random index designation between lth index to rth index
41      p1 = rand() % (r - l) + l;
42      p2 = rand() % (r - l) + l;
43      p3 = rand() % (r - l) + l;
44
45      pivot_index = median(Q, p1, p2, p3); //the middle value of p1, p2, p3
46      return pivot_index;
47  }
48
```

Fourth. C-code

```
49  int inPlacePartition(int Q[], int l, int r, int k)
50  {
51      int i, j, p;
52      p = Q[k];
53      swap(Q, k, r); //Swap pivot(p) to the last element
54      i = l; j = r - 1;
55
56      //Align to the right of the pivot if greater than or equal to the pivot,
57      //or to the left of the pivot if smaller or equal to the pivot.
58      while (i <= j)
59      {
60          while (i <= j && Q[i] <= p) i++;
61          while (j >= i && Q[j] >= p) j--;
62          if (i < j) swap(Q, i, j);
63      }
64      swap(Q, i, r); //If i and j reverse, change the pivot and ith number
65
66      return i;
67  }
68
```

a function that randomly selects three pivot indexes and sets the middle value of them as pivot.

Fourth. C-code

```c
void inPlaceQuickSort(int Q[], int l, int r)
{
    int pivot, a, b, tmp;

    if (l >= r) return; //return if l reverse r

    pivot = find_pivot_index(Q, l, r); //find the pivot index

    //sorting by pivot
    //b is the last index of the same element as pivot
    b = inPlacePartition(Q, l, r, pivot);

    tmp = b;
    while (1)
    {
        if (Q[tmp] != Q[b]) {
            a = tmp + 1;
            break;
        }
        tmp--;
    }

    inPlaceQuickSort(Q, l, a - 1);
    inPlaceQuickSort(Q, b + 1, r);
}
```

The same number of index ranges as pivots are a, b, the process of obtaining a through b

Quick Sort algorithms

```c
void main()
{
    int N, *Q, x, i, a, b;
    scanf("%d", &N); //enter the array size
    //Dynamic assignment of arrays of size N
    Q = (int*)malloc(sizeof(int)*N);
    for (i = 0; i < N; i++)
    {
        scanf("%d", &x);
        Q[i] = x;
    } //save input elements in array

    inPlaceQuickSort(Q, 0, N - 1);

    for (i = 0; i < N; i++)
        printf(" %d", Q[i]); //print a sorted array
}
```

Main function : Enter array size N, and store the entered integer by dynamically assigning an array of size N. Declares a quick-sort algorithm and print an ordered array.

thank you