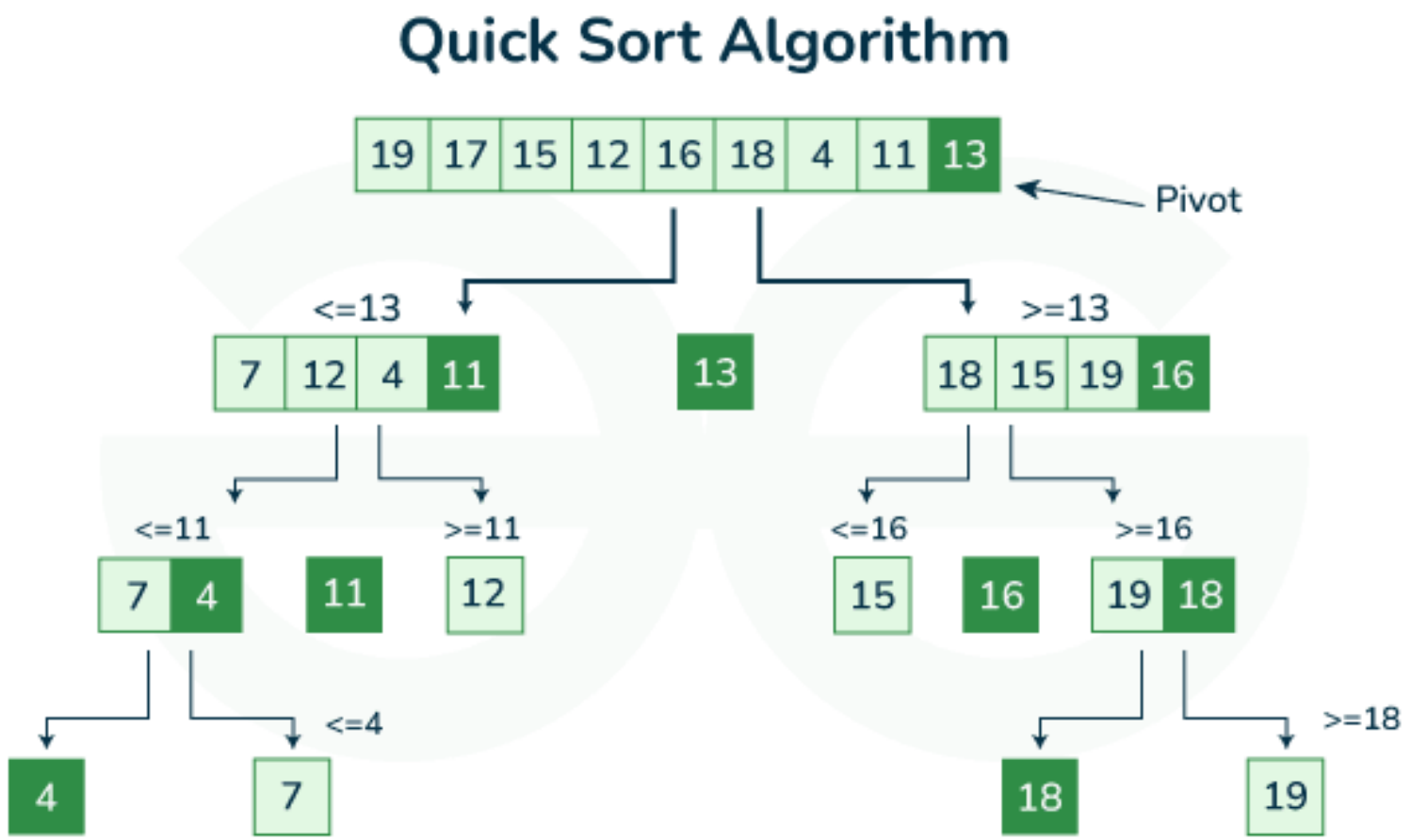
Швидке сортування



Алгоритм швидкого сортування

- 1. Вибрати опорний елемент масиву (середній елемент, кінцевий елемент, випадковий елемент)
- 2. Посортувати елемени відносно опорного елемента: якщо сортуємо у порядку зростання, то зліва від опорного елемента мають бути елементи менші за нього, а справа більші.
- 3. Знайти номер розділюючого елемента і повторити процедуру рекурсивно для елементів підмасиву, що зліва від розділюючого елемента та елементів підмасиву, що справа від розділюючого елемента

Псевдокод рекурсивної реаоізації алгоритму швидкого сортування

quickSort(int[]a, int low, int high){

1. *If (low >= high)* виходимо з методу;
2. Вибираємо опорний елемент (розглянути 3 випадки: опорний елемент є центральним, останнім і випадковим)

pivot=a[low+(high-low)/2]; pivot=a[high]

3. Створюємо змінні (i, j) для перебору елементів зліва від опорного елемента і справа від нього відповідно (індексування елементів масиву зліва і справа)

i=low; j=high;

4. Перебираємо елементи зліва від опорного елемента, починаючи від першого елемента. Якщо елемент менший за опорний то переходимо на наступний елемент, якщо ні то фіксуємо індекс знайденого елемента

```
while(a[i] < pivot) {
        i++;
}
```

5. Перебираємо елементи справа від опорного елемента, починаючи від останнього елемента, якщо елемент більший за опорний переходимо на попередній інакше фіксуємо індекс знайденого елемента

```
while(a[j] > pivot) {
        j--
}
```

6. Кроки 4 і 5 виконуємо поки *i<=j*

Псевдокод рекурсивної реалізації алгоритму швидкого сортування (продовження)

7. Якщо було знайдено елемент зліва від опрного елемента, який більший за нього, а справа від опорного елемента відповідно елемент, який менший за нього, то виконуємо перестановку елементів місцями

```
if(i<=j) {
swap(a, i, j) {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
} ;
i++; j--;
}
```

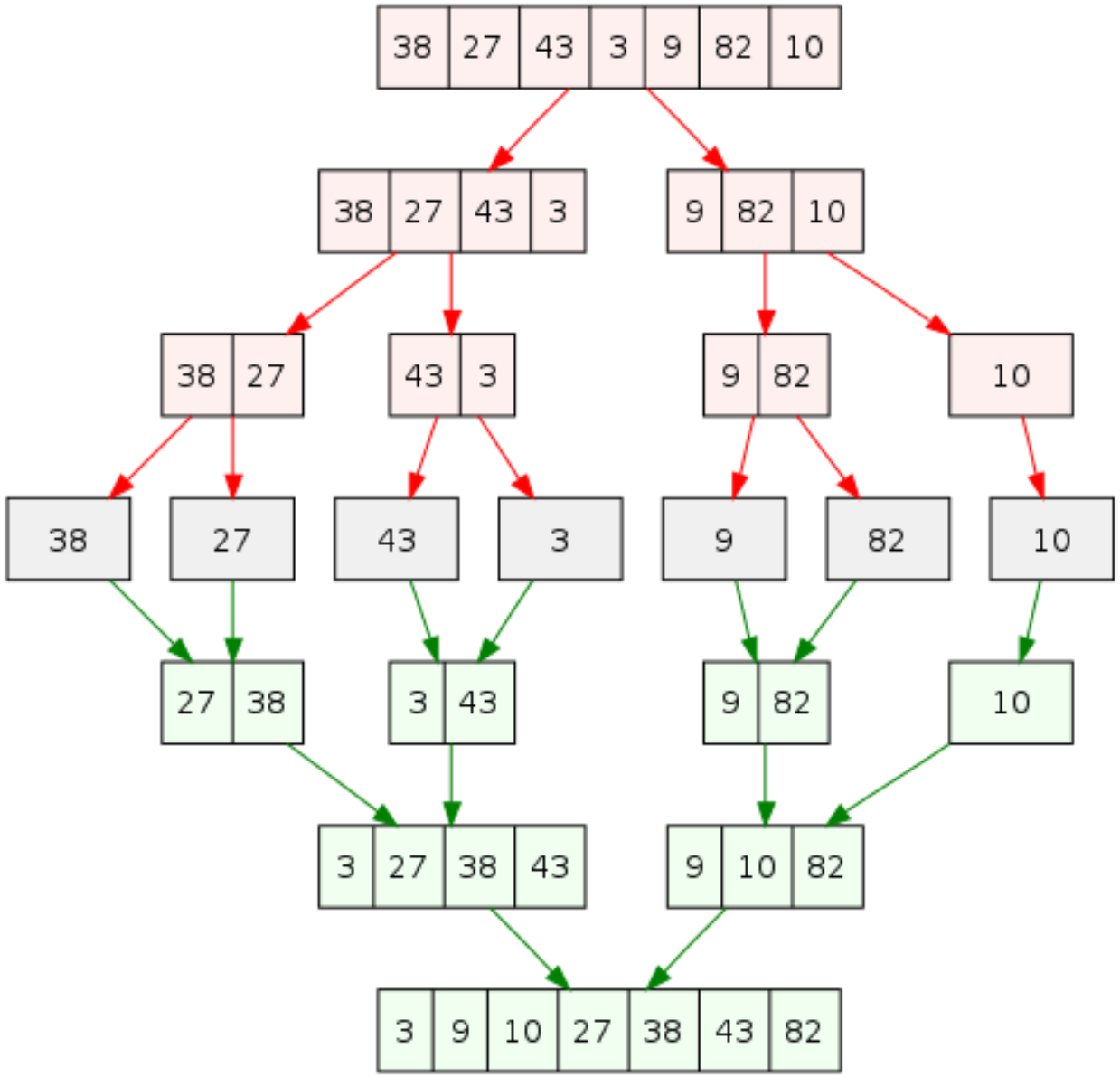
8. Повторити процедуру для елементів підмасиву зліва від розділюючого елемента

```
quickSort(a, low, i-1);
```

9. Повторити процедуру для елементів підмасиву справа від розділюючого елемента

```
quickSort(a, i, high);
```

Сортування злиттям



Алгоритм сортування злиттям

1. Ділимо вхідний масив рекурсивно на два підмасиви доти, поки не отримаємо підмасиви з одного елемента

2. Отримані підмасиви зливаємо в один масив і при цьому виконуємо сортування елементів

    2.1. Перебираємо елементи двох відповідних підмасивів і порівнюємо їх між собою. У результуючий масив спочатку записуємо елемент, який менший, а потім той, що більший (якщо сортуємо за заростанням)

    2.2. Підмасиви можуть бути не однакової довжини, тоді потрібно дописати елементи що залишилися просто в кінець результуючого масиву.

3. Повторити процедуру рекурсивно для всіх підмасивів, отриманих врезультаті поділу вхідного масиву

Псевдокод для сортування злиттям

mergeSort(int [] a) {

1. Якщо розмір вхідного масиву *n=1* закінчуємо поділ

if(a.length==1) вийти з функції;

2. Створюємо два нових підмасиви

mid=a.length/2;
left=new int[mid];
right=new int[a.length-mid];

3. В перший підмасив записуємо першу половину елементів вхідного масиву, а в другий масив – другу половину елементів вхідного масиву

for(int i=0; i<mid; i++){
left[i]=a[i];}
for(int i=mid; i< a.length; i++) {
right[i-mid]=a[i];}

4. Ділимо отримані підмасиви знову на два рекурсивно доки не отримаємо підмасив з одного елемента

mergeSort(left);
mergeSort(*right*);

5.Зливаємо та сортуємо отримані підмасиви в результуючий:

merge(a, left, right);

}

Псевдокод для сортування злиттям(продовження)

merge(int[]a, int[]left, int[]right) {

1. Створюємо дві змінні (i=0, j=0) для індексування по двох відповідних підмасивах (left, right), які утворилися в результаті поділу, а також одну змінну (id=0) для індексування по результуючому масиві (a)

2. Перебираємо елементи двох підмасивів та порівнюємо їх між собою і вставляємо менший з них у результуючий масив та переходимо на наступний елемент

```
while(i<left && j<right) {
        if(left[i]<right[j]) {
                a[idx]=left[i]; i++;
        } else {
                a[idx]=right[j]; j++;
        }
        idx++;
}
```

3. Дописуємо елементи, які залишились або в лівому або в правому підмасивах в кінець результуючого масиву

```
for(tail=i; tail<left.length; tail++) {
        a[idx++]=left[tail]; }
for(tail=j; tail<right.length; tail++) {
        a[idx++]=lrught[tail];}
}
```