Семинар по „Увод в програмирането“

Работа върху масиви.

Сортиращи алгоритми и binary search.

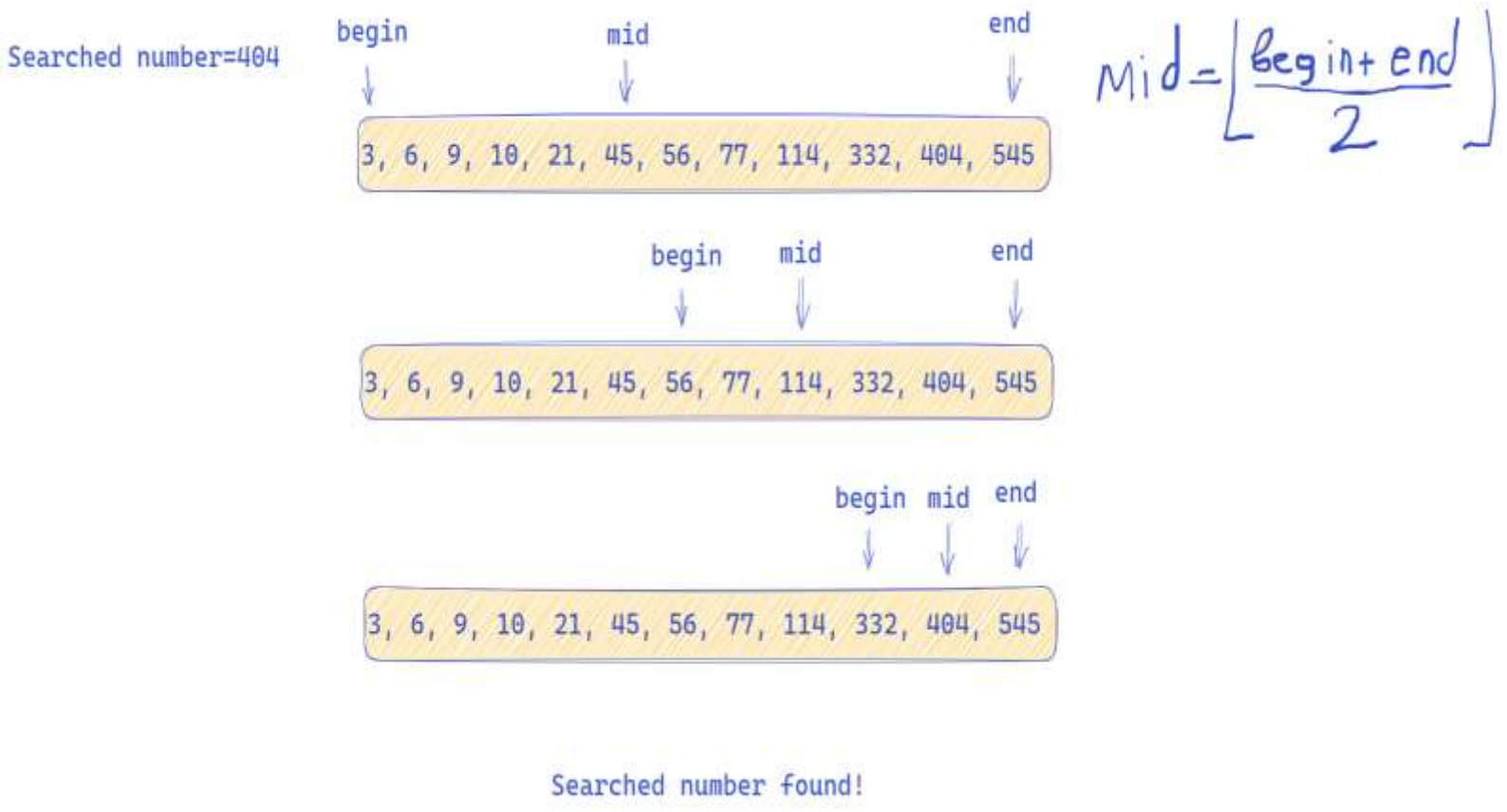
1. Linear search

Linear search е метод за търсене на елемент в масив, който последователно обхожда масива от неговото начало, докато се намери търсения елемент. В най-лошия случай той ще обходи всички елементи от началото до края.

2. Binary search

Binary search представлява алгоритъм за търсене на елемент в дадена масив от елементи, който постепенно намалява обхвата на работа на масива с точно половина, докато намери търсения елемент. Важно услове, за да мжоже **Binary search** алгоритъма да работи е **масивът да е сортиран.**

Схема на алгоритъма **Binary search:**

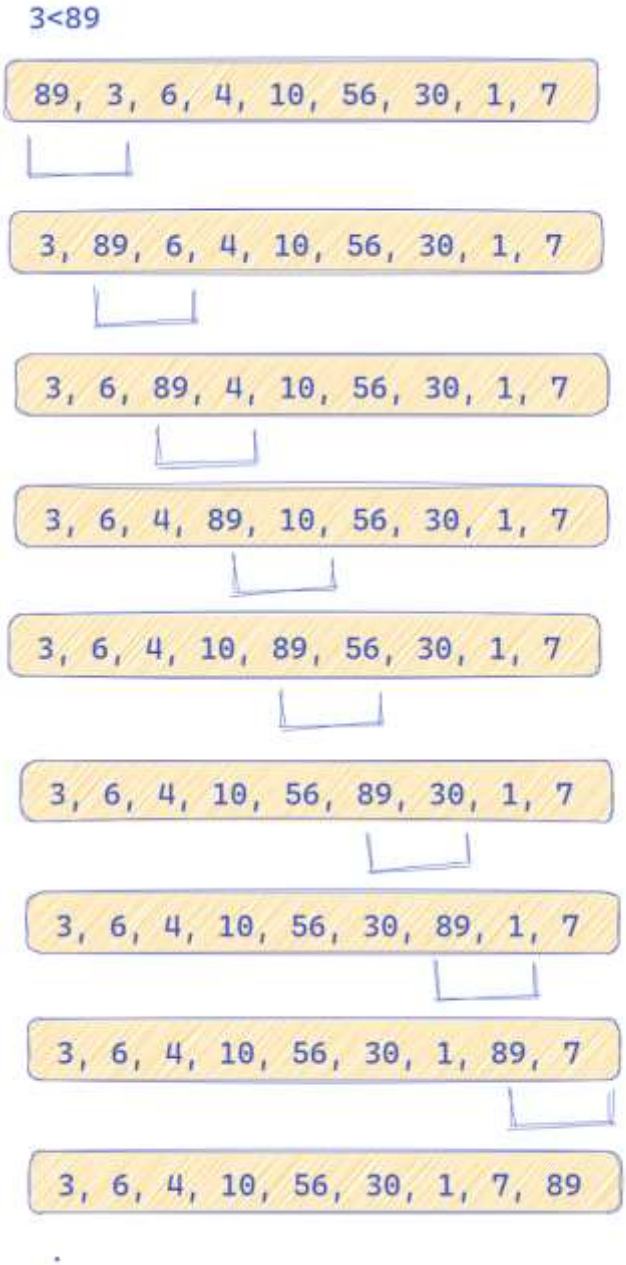


Както се вижда на схемата имаме имаме индекс към началния елемент(**begin=3**), индекс към крайния елемент (**end=545**) и индекс към средата (**mid=45**), който ще наричаме още **ключ**. Сега проверяваме нашия **ключ** дали е равен на търсения от нас елемент. Отговорът е не, следователно намаляме работния капацитет наполовина. На втора стъпка виждаме, че **begin=56 (mid+1)**, **end** си остава същия, а **mid=114.** Отново проверяваме дали текущия ни ключ(**mid=114**) е равен на търсената от нас стойност. Отговорът отново е не, следователно преминаваме към трета стъпка. Сега **begin=332, end** отново си остава същия, а новия ключ е **mid=404.** Правим проверка дали нашия ключ е равен на търсената стойност. Отговорът е **да,** следователно алгоритъма приключва, връщайки позицията на търсената стойност.

Класически сортиращи алгоритми

3. Bubble Sort

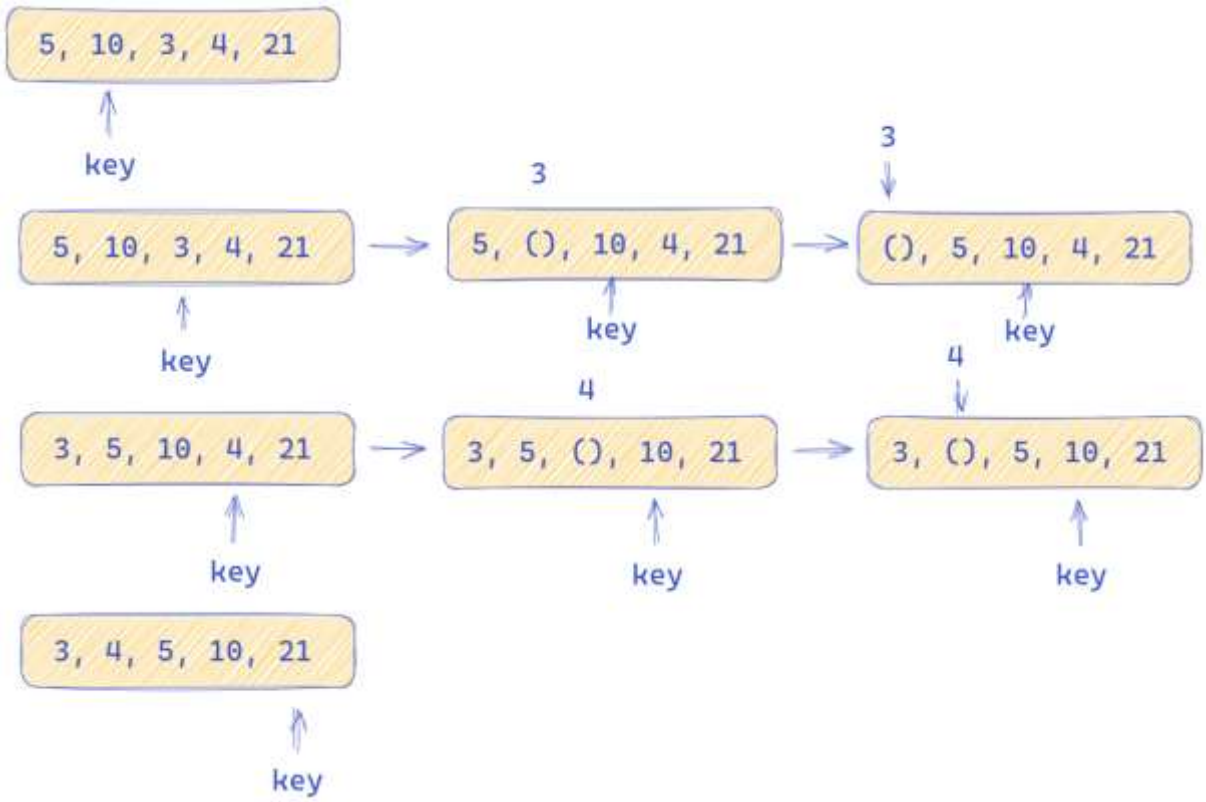
Схема на работа на **Bubble sort:**



Bubble sort работи като взима двойка последователни елементи и ги сравнява. Ако първия е по-голям от втория, ги разменя, ако не запазва наредбата им и продължава със следващата двийка докато стигне до последната двойка елементи. Един такъв блок от сранения и размени съдържа в себе си n - i - 1 операции като **n** е големината на колекцията, а **i**={1…n}. Следователно **Bubble sort**, в най-лошия случай, ще се изпълни за (n-1) + (n-2) + … + 1 = (n*(n-1)) / 2 стъпки.

4. Insertion sort

Схема на **Insertion sort:**

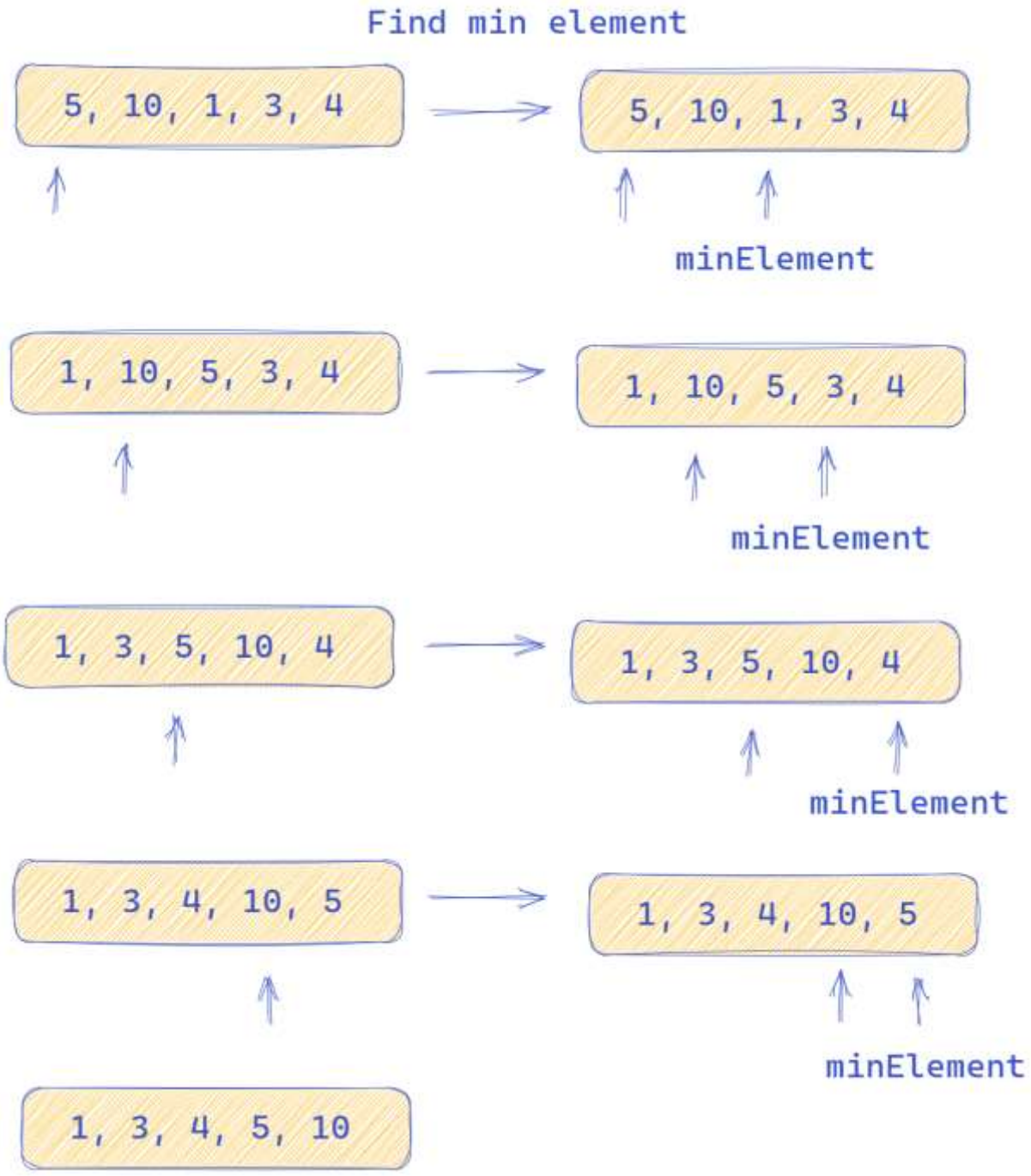


Заадаваме **key**, в началото, да е равен на втория елемент в масива. Сега сравнение със предходия елемент дали е по-голям от него. Отговорът е не, следователно запазваме наредбата им. Задаваме **key** да е равен на следващия елемент(3). Правим сравнение с предходния елемент дали е по-голям от него. Отговорът е да, следователно намираме неговото място като идърпваме предходните елементи една позиция надясно, докато текущия предходем елемент не е по-малък от него или докато не стигнем началото на масива. Така този алгоритъм се повтаря докато **key** не стане последния елемент на масива и не си намери своето място. В най-лошия случай този алгоритъм работи за

1 + 2 + ... + (n-1) = (n*(n-1)) / 2 стъпки.

5. Selection sort

Схема на **Selection sort:**



Започваме от първия елемент на масива. Намираме най-малкия елемент в остатъка от масива. След като го намерим разменяме стойностите им и продължаваме към селдващия елемент. Тези операции в най-лоши случай са:

n (сравнения за намиране на най-малкия елемент на всяка итерация) * **n** (за да вземем всеки елемент и да го подредим на мястото му)