Семинар 10

Введение в программирование на Python

Папулин С.Ю.
papulin_hse@mail.ru

2015

Семинар 9

1. Методы определения min/max
2. Обмен значений переменных
3. Методы сортировки

План семинара 10

1. Поиск элемента
2. Последовательный поиск
3. Бинарный поиск

Поиск элемента

Поиск элемента

Примеры поиска элементов в списке

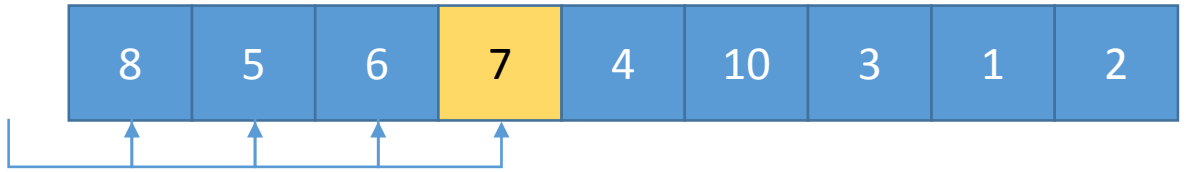
```
myList = [8,5,6,7,4,10,3,1,2]

minEl = min(myList)
maxEl = max(myList)

minIndx = myList.index(minEl)
maxIndx = myList.index(maxEl)

if 5 in myList:
    pass #do something
```

Последовательный поиск

Последовательный поиск

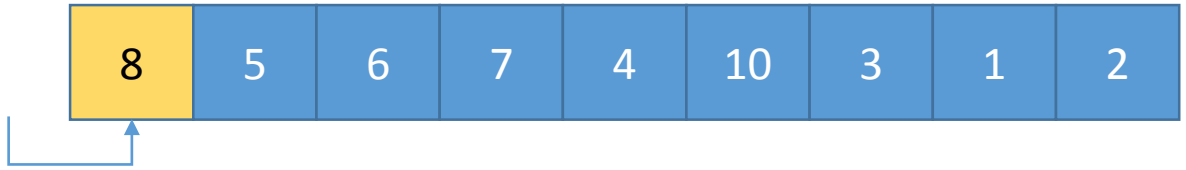
Примеры последовательного поиска элемента в неотсортированном списке

```
myList = [8,5,6,7,4,10,3,1,2]
```



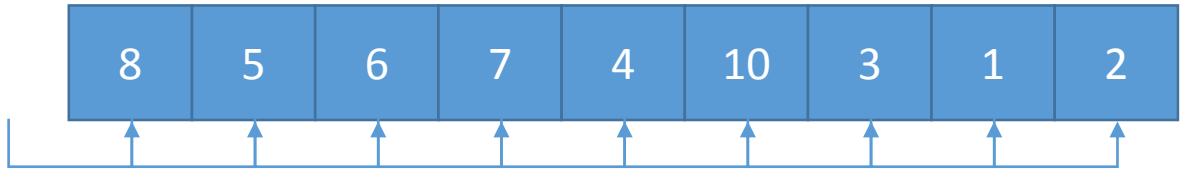
4

Наилучший случай



1

Наихудший случай



n

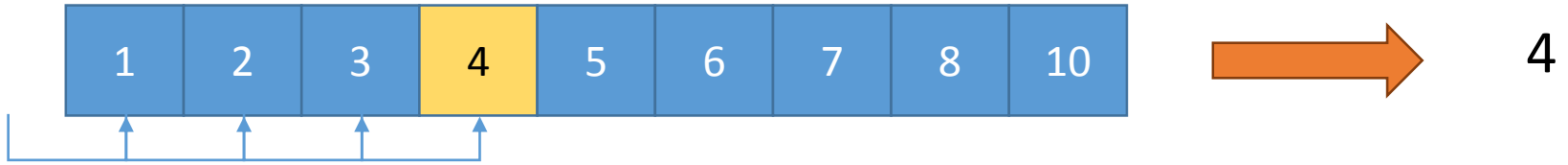
Последовательный поиск

Вариант	Наилучший случай	Наихудший случай	В среднем
Элемент присутствует	1 -> O(1)	n-> O(n)	n/2 -> O(n)
Элемент отсутствует	n -> O(n)	n -> O(n)	n -> O(n)

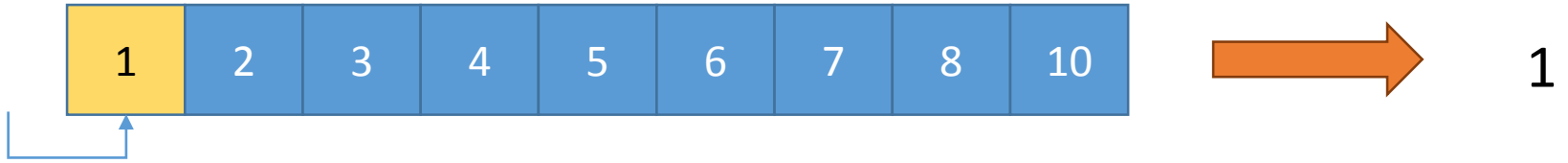
Последовательный поиск

Примеры последовательного поиска элементов в отсортированном списке

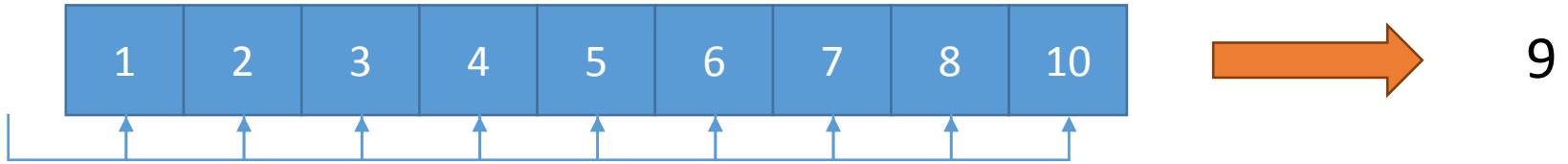
```
mySortedList = [1,2,3,4,5,6,7,8,10]
```



Наилучший случай

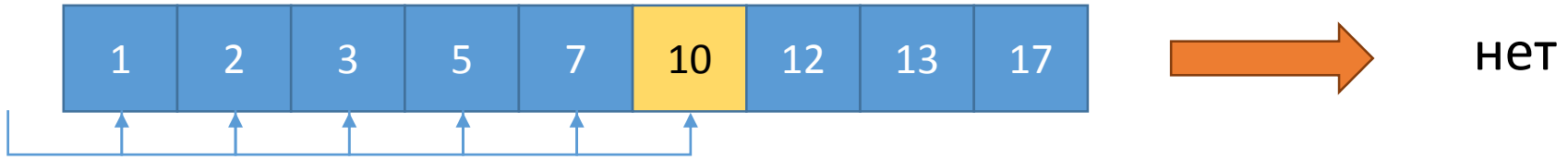


Наихудший случай



Последовательный поиск

Есть ли в списке 8?



Вариант	Наилучший случай	Наихудший случай	В среднем
Элемент присутствует	1 -> O(1)	n -> O(n)	n/2 -> O(n)
Элемент отсутствует	1 -> O(1)	n -> O(n)	n/2 -> O(n)

Бинарный поиск

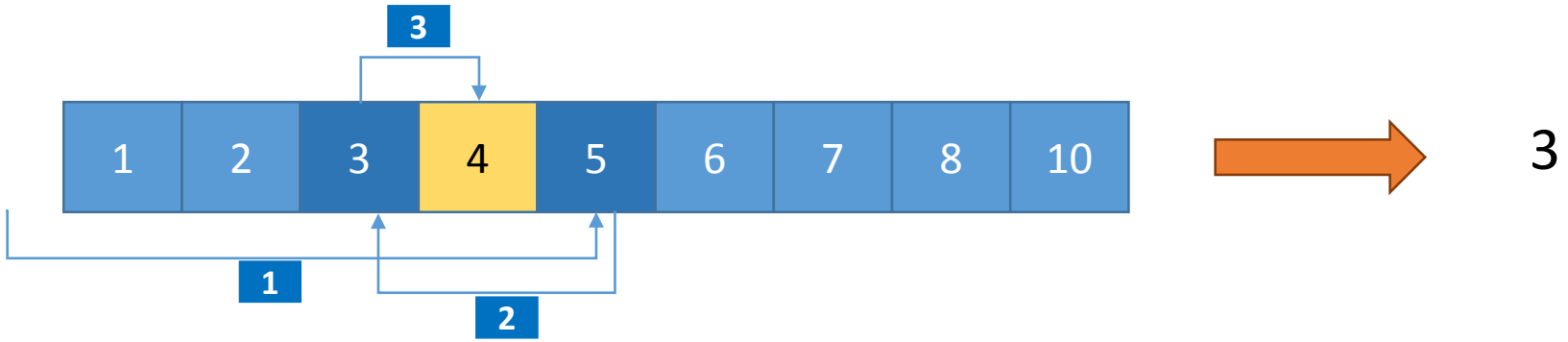
Бинарный поиск

Условие использования: *список должен быть* ***отсортирован***

```
mySortedList = [1,2,3,4,5,6,7,8,10]
```

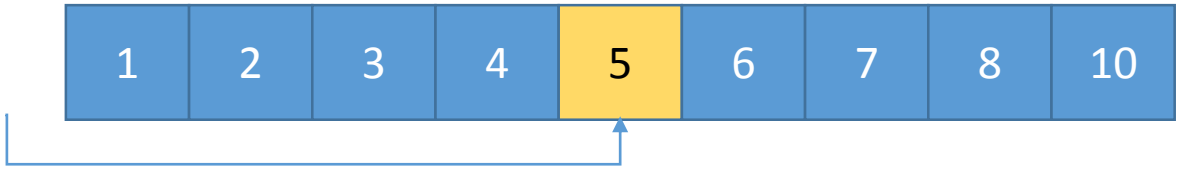
1) Проверяем элемент в середине списка (срединный элемент)
2) Если его значение совпадает с требуемым, останавливаем поиск
3) Если значение больше, то повтор поиска с шага 1 для части списка перед срединным элементом
4) Если значение меньше, то повтор поиска с шага 1 для части списка после срединным элемента

Есть ли в списке 4?



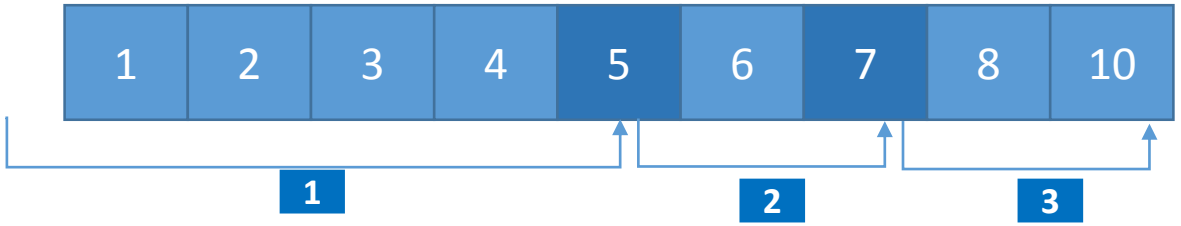
Бинарный поиск

Наилучший случай



1

Наихудший случай



3

Сложность	Наилучший случай	Наихудший случай	В среднем
Вычисление	O(1)	O(log n)	O(log n)

Бинарный поиск рекурсией

```
mySortedList = [1,2,3,4,5,6,7,8,10]

def binarySearch(alist, item):
    if len(alist) == 0:
        return False
    else:
        midpoint = len(alist)//2
        if alist[midpoint]==item:
            return True
        else:
            if item<alist[midpoint]:
                return binarySearch(alist[:midpoint],item)
            else:
                return binarySearch(alist[midpoint+1:],item)

print(binarySearch(mySortedList, 5))
```

True

В чем недостаток данной рекурсивной реализации бинарного поиска?

Последовательный vs Бинарный поиск

Какой способ поиска лучше?

Семинарские задачи

https://official.contest.yandex.ru/contest/1786/enter/

Источники

http://interactivepython.org/runestone/static/pythonds/index.html