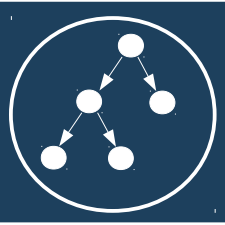


Data Structures and Algorithms

Алгоритмы.
Линейный(последовательный) поиск.

Сведение о алгоритме

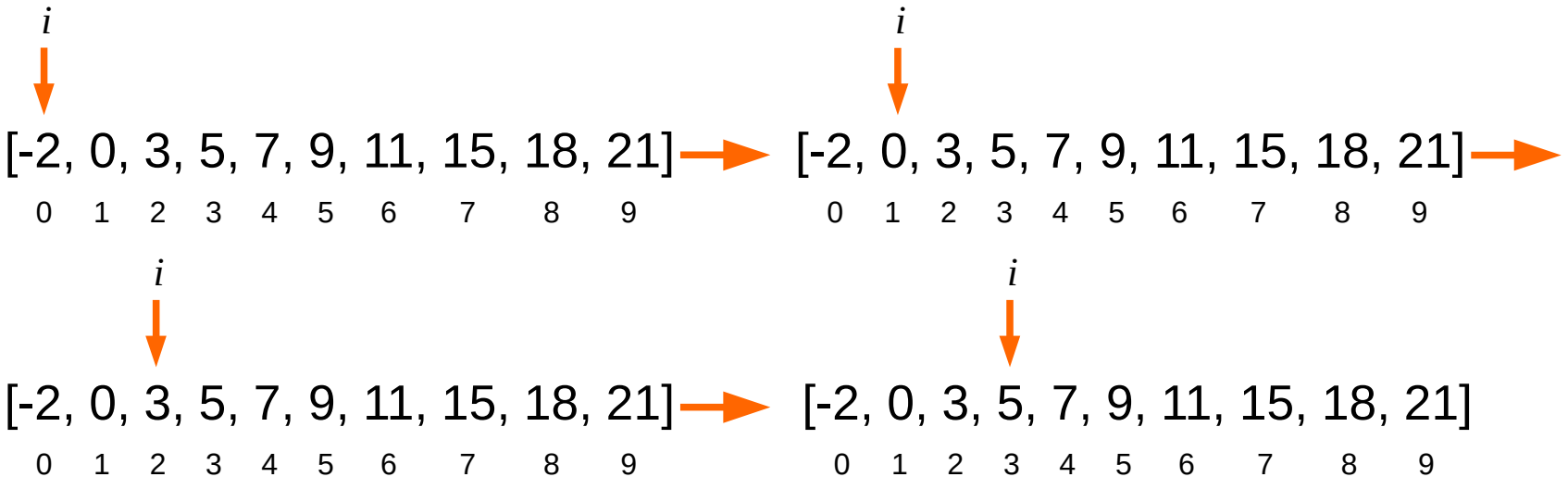
Алгоритм линейного поиска

Сложность по времени в наихудшем случае	O(n)
Затраты памяти	O(n)

Принцип работы алгоритма

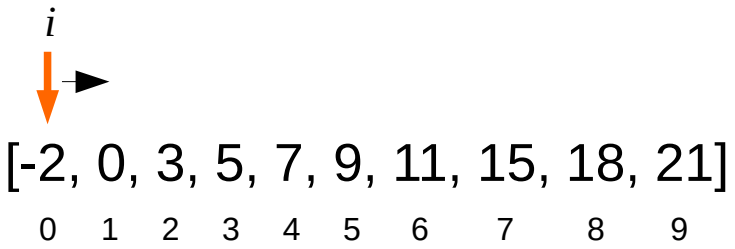
1) Установить начальное значения индекса (обозначим его *i*) равным индексу первого элемента в последовательности.
2) Получить элемент стоящий по индексу n[i]. Возможны два случая:
   a) Ключ элемента равен искомому. Вернуть индекс и завершить алгоритм. Поиск успешен.
   b) Ключ элемента не равен искомому. Увеличить индекс на единицу. *i=i+1*. Перейти к пункту **3**.
3) Проверить, индекс меньше или равен индексу последнего элемента в последовательности. Если да, то перейти к пункту **2**. Если нет, то завершить алгоритм. Поиск неудачен.

Графическая иллюстрация работы алгоритма



Работа алгоритма продемонстрирована в предположении, что искомым элементом является **5**.

Графическая иллюстрация работы алгоритма

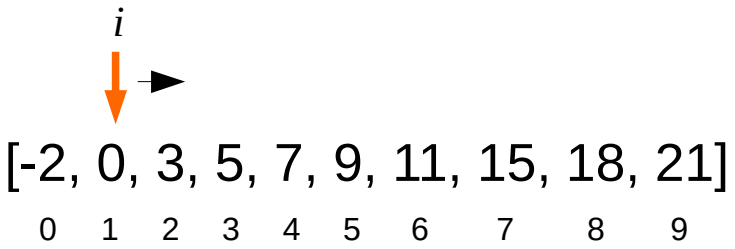


Предположим что индексация в последовательности начинается с 0. На первом шаге устанавливаем значение индекса равный 0. *i=0*. Элемент на этом индексе не равен искомому. Увеличиваем значение индекса на единицу.

i=i+1=0+1=1

Этот индекс меньше индекса последнего элемента. $i \leq 9$. Продолжаем алгоритм.

Графическая иллюстрация работы алгоритма

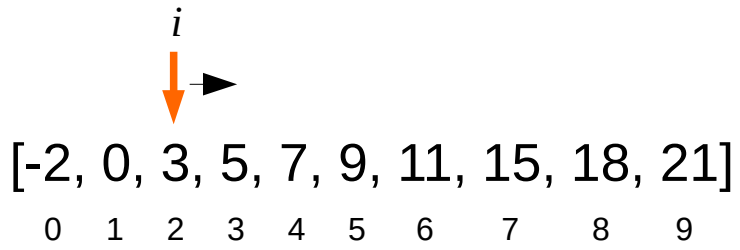


Элемент на этом индексе не равен искомому. Увеличиваем значение индекса на единицу.

i=i+1=1+1=2

Этот индекс меньше индекса последнего элемента. $i \leq 9$. Продолжаем алгоритм.

Графическая иллюстрация работы алгоритма

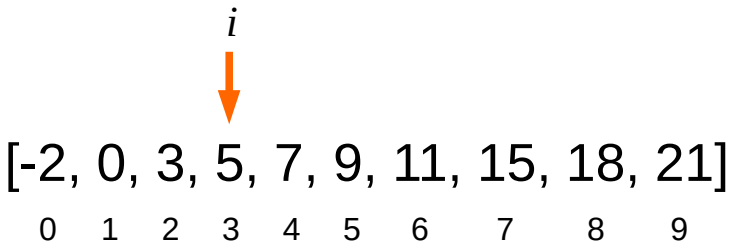


Элемент на этом индексе не равен искомому. Увеличиваем значение индекса на единицу.

i=i+1=2+1=3

Этот индекс меньше индекса последнего элемента. $i \leq 9$. Продолжаем алгоритм.

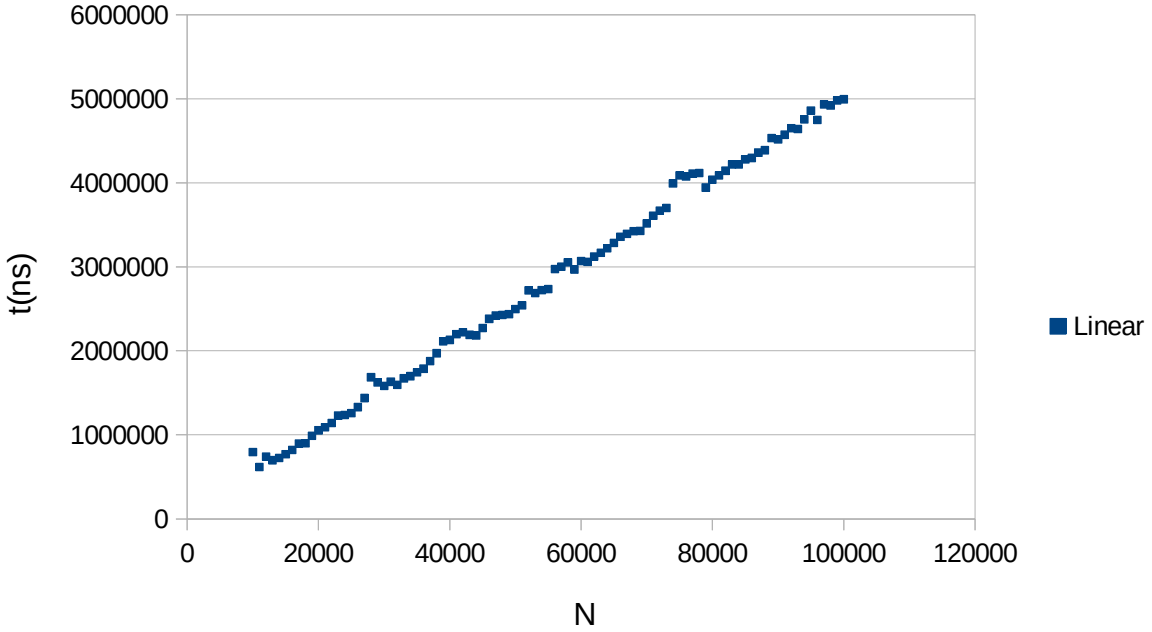
Графическая иллюстрация работы алгоритма



Элемент на этом индексе равен искомому. Заканчиваем алгоритм. Поиск успешен.

Вычислительный эксперимент

С целью проверки асимптотического поведения реализации алгоритма линейного поиска, был проведен вычислительный эксперимент. В последовательностях разных размеров (были взяты массивы) выполнен поиск 1000 элементов. Построен график зависимости времени поиска от размеров последовательностей.



Модификация алгоритма линейного поиска

Алгоритм линейного поиска можно ускорить путем исключения пункта 3, в базовом алгоритме. Т.е. можно исключить проверку принадлежности индекса значения индексам в последовательности. Для этого стоит добавить в конец базовой последовательности элемент равный искомому.

Стоит отметить, что такая модификация имеет смысл только для последовательностей в которых вставка и удаление элемента не связанны с высокими вычислительными затратами.

Принцип работы модифицированного алгоритма линейного поиска

1) Добавить элемент (значение ключа которого равно искомому) в конец последовательности.
2) Установить начальное значения индекса (обозначим его *i*) равным индексу первого элемента в последовательности.
3) Получить элемент стоящий по индексу n[i]. Возможны два случая:
   a) Ключ элемента равен искомому. Удалить последний элемент. Если индекс не равен индексу последнего элемента — **поиск успешен**. В случае равенства индекса индексу последнего элемента — **поиск не удачен**.
   b) Ключ элемента не равен искомому. Увеличить индекс на единицу. *i=i+1*. Перейти к началу пункта **3**.

Python

Реализация алгоритма на Python

Реализация алгоритма на Python

```
def liniar_search(sequince, element):
    for i in range(len(sequince)):
        if sequince[i] == element:
            return i
    return -1
```

Реализация алгоритма линейного поиска

Реализация алгоритма на Python

```
def modified_linear_search(sequince, element):
    sequince.append(element)
    i = 0
    while sequince[i] != element:
        i = i+1
    sequince.pop()
    if i != len(sequince):
        return i
    return -1
```

Реализация модифицированного алгоритма линейного поиска

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

```
public static int linearSearch(int[] sequince, int element) {
    for (int i = 0; i < sequince.length; i++) {
        if (sequince[i] == element) {
            return i;
        }
    }
    return -1;
}
```

Реализация алгоритма линейного поиска

Реализация алгоритма на Java

```
public static int modifiedLinearSearch(List<Integer> sequince, int element) {
    sequince.add(element);
    int lastIndex = sequince.size() - 1;
    int i = 0;
    for (; sequince.get(i) != element;) {
        i++;
    }
    sequince.remove(lastIndex);
    if (i != lastIndex) {
        return i;
    }
    return -1;
}
```

Реализация модифицированного алгоритма линейного поиска

Список литературы

1) Дональд Кнут. «Искусство программирования, том 3. Сортировка и поиск» 2-е изд. М.: «Вильямс», 2007. С. 824. ISBN 0-201-89685-0. [426 -429]
2) Ананий Левитин. Алгоритмы: введение в разработку и анализ. : Пер. с англ. — М. :Издательский дом "Вильямс", 2006. — 576 с. : ил. — Парал. тит. Англ. ISBN 5-8459-0987-2. Стр. [147-148]