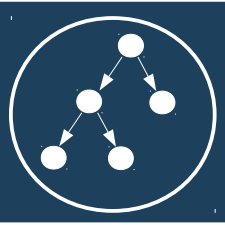


Data Structures and Algorithms

Алгоритмы. Интерполяционный поиск.

Сведение о алгоритме

Алгоритм интерполяционного поиска.

Сложность по времени в наихудшем случае O(n)

Принцип работы алгоритма

В основе алгоритма лежит интерполяция зависимости элемента последовательности от значения его индекса. Полученная интерполяционная зависимость используется для возможности предсказания местоположения элемента. Как и в случае с бинарным поиском область поиска уменьшается с каждым шагом. Бинарный поиск уменьшает область поиска всегда в два раза. В случае хорошо подобранной интерполяционной зависимости область поиска сужается быстрее, в тоже время неудачно подобранная интерполирующая зависимость может привести к падению эффективности до линейной.

В общем случае создание интерполяционной зависимости хотя и возможно но затруднительно. Гораздо более простой и эффективный способ заключается в использовании поиска в отсортированной последовательности.

Интерполяция

Интерполяция, интерполирование — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

Имеющиеся данные вида (x_i, y_i) — точки данных, базовые точки. Где i = 1...N.

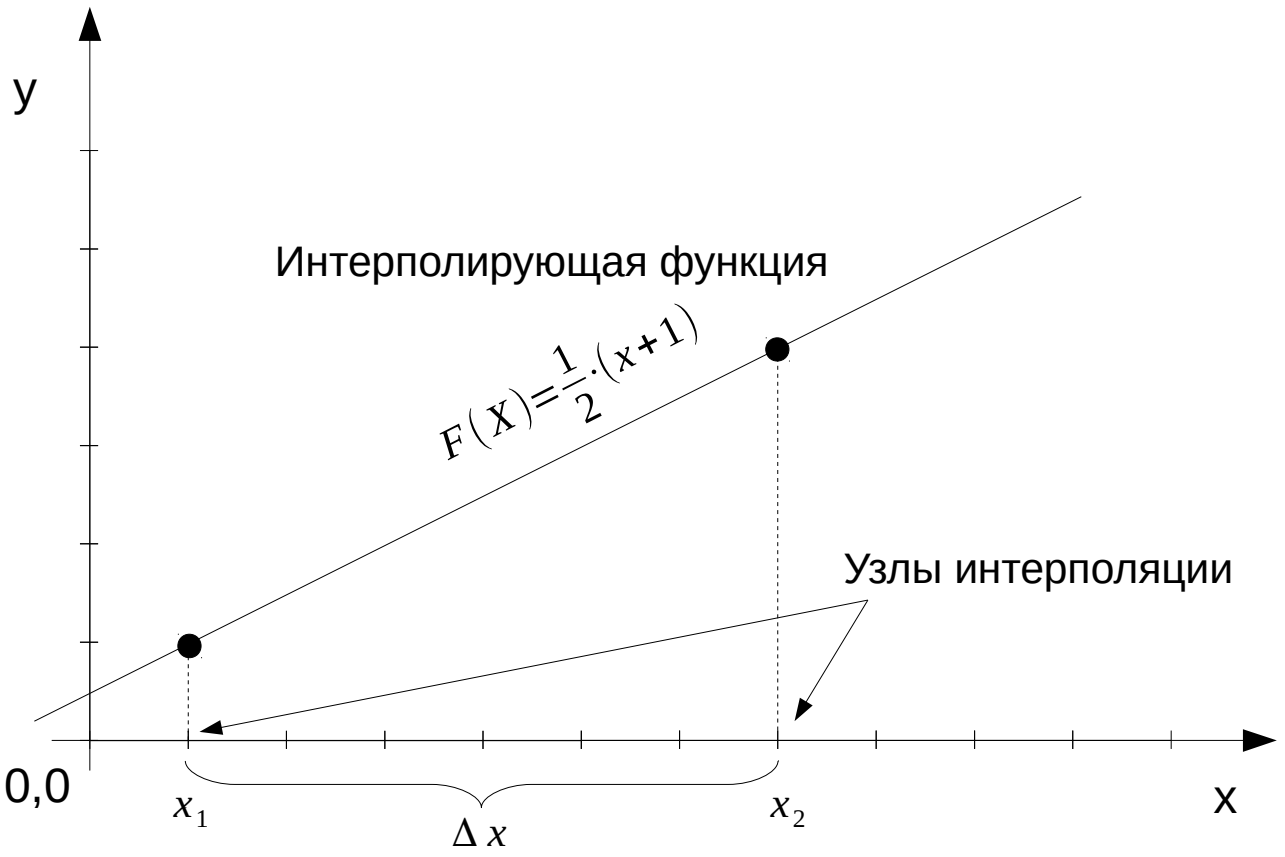
x_i — узлы интерполяции

$\Delta x = x_i - x_{i-1}$ — шаг интерполяционной сетки

Задача интерполяции состоит в нахождении такой функции F, что $F(x_i) = y_i$

F — интерполирующая функция

Интерполяция



Базовые точки

x	y
1	1
7	4

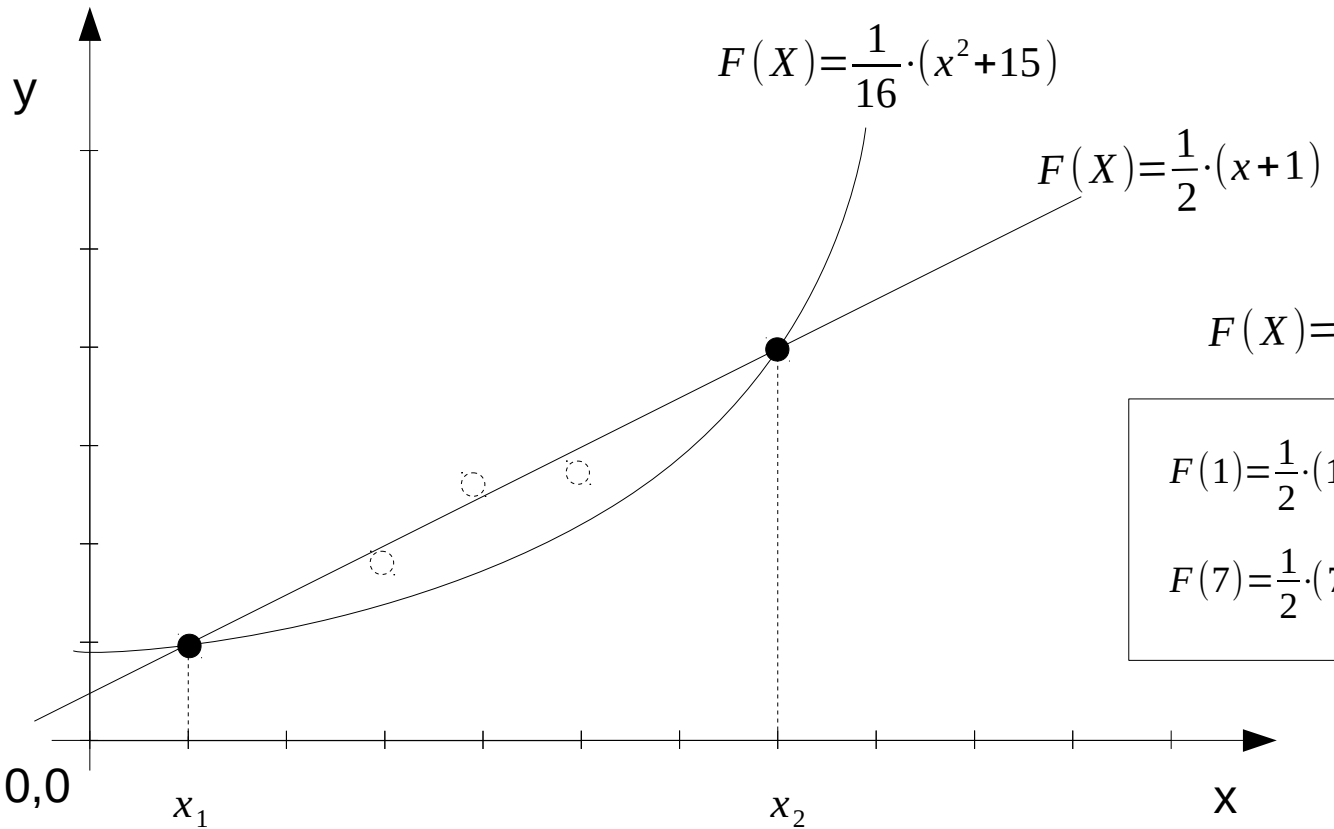
$$F(1)=\frac{1}{2}\cdot(1+1)=1$$

$$F(7)=\frac{1}{2}\cdot(7+1)=4$$

Разные интерполяционные функции

Базовые точки

x	y
1	1
7	4



$F(X)=\frac{1}{2}\cdot(x+1)$

$F(1)=\frac{1}{2}\cdot(1+1)=1$

$F(7)=\frac{1}{2}\cdot(7+1)=4$

$F(X)=\frac{1}{16}\cdot(x^2+15)$

$F(1)=\frac{1}{16}\cdot(1^2+15)=1$

$F(7)=\frac{1}{16}\cdot(7^2+15)=4$

Линейная интерполяция

Пожалуй наиболее простым методом интерполяции по двум точкам (в общем случае интерполяцию можно проводить по N≥2 точкам) является линейная интерполяция. В таком случае в качестве интерполирующей функции используется уравнение прямой по двум точкам. Довольно часто алгоритм интерполяционного поиска использует именно линейную интерполяцию.

Уравнение прямой по двум точкам (2-х мерная система координат):

$$\frac{x-x_1}{x_1-x_2}=\frac{y-y_1}{y_1-y_2}$$

Отсюда можно получить:

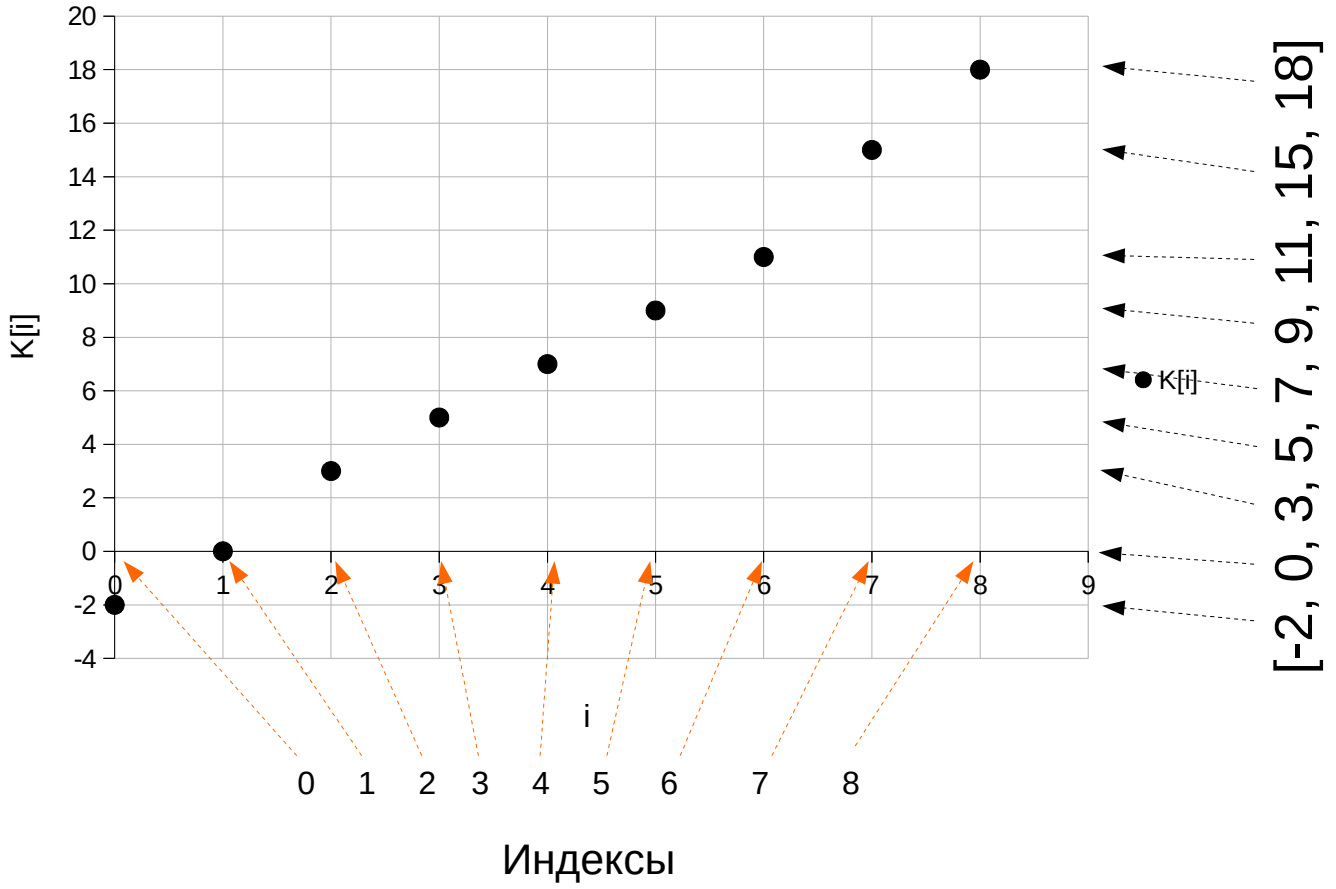
$$\frac{x-x_1}{x_1-x_2}=\frac{y-y_1}{y_1-y_2}\Rightarrow(x-x_1)\cdot(y_1-y_2)=(y-y_1)\cdot(x_1-x_2)$$

Требуемые прямые и обратные зависимости:

$$(x-x_1)\cdot(y_1-y_2)=(y-y_1)\cdot(x_1-x_2)\Rightarrow y(x)=\frac{(x-x_1)\cdot(y_1-y_2)}{x_1-x_2}+y_1$$

$$(x-x_1)\cdot(y_1-y_2)=(y-y_1)\cdot(x_1-x_2)\Rightarrow x(y)=\frac{(y-y_1)\cdot(x_1-x_2)}{y_1-y_2}+x_1$$

Использование линейной интерполяции в алгоритме



Значения и индексы

[-2, 0, 3, 5, 7, 9, 11, 15, 18]

0 1 2 3 4 5 6 7 8

Для применения алгоритма интерполяционного поиска в последовательности будем использовать индексы последовательности в качестве x координат точек, а значения в качестве y координат точек.

Использование линейной интерполяции в алгоритме

Для работы будем использовать отсортированную последовательность. В качестве базовых точек используем пары (i, K[i]) где i — индекс в последовательности, K[i] — значение элемента по этому индексу. Используем линейную интерполяцию по двум точкам. Предсказание индекса на котором стоит искомый элемент (обозначим его K) будем проводить по обратной зависимости. Выберем две точки для построения интерполяционной прямой. Обозначим их как l и r соответственно. В таком случае индекс искомого элемента вычисляется по формуле:

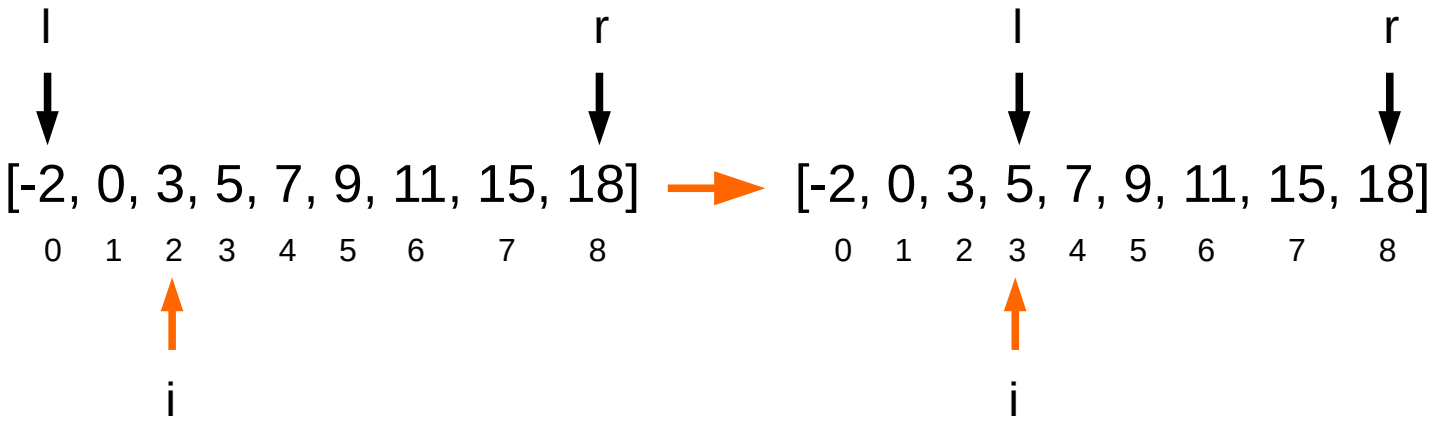
$$i(K) = \frac{(K - K[l]) \cdot (l - r)}{K[l] - K[r]} + l$$

Принцип работы алгоритма

1) Сортируется последовательность в которой будет проводится поиск. Если последовательность уже отсортирована то этот шаг можно пропустить.
2) В качестве начальных точек (левая и правая) выбираем первый и последний элемент последовательности.

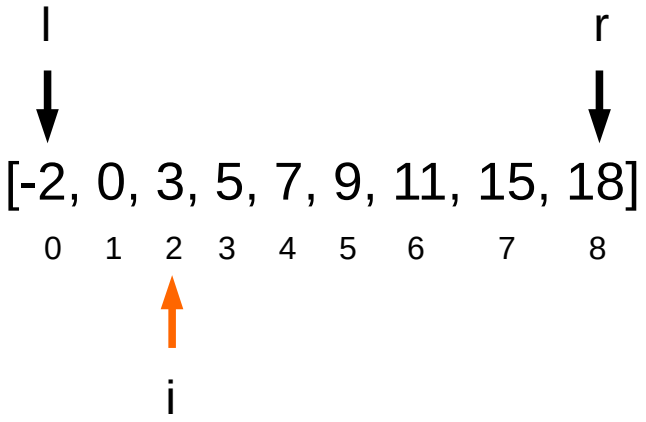
3) Определяется значение индекса по формуле линейной интерполяции. Получаем элемент по этому индексу. Полученный элемент сравнивается с искомым элементом. Различают случаи :
    a) Элемент равен искомому. Заканчиваем алгоритм. Поиск успешен.
    b) Элемент больше искомого. Сдвигаем правую точку. Новое значение (найденный индекс - 1).
    c) Элемент меньше искомого. Сдвигаем левую точку. Новое значение (найденный индекс +1).
4) Повторяем пункт 3 до тех пор, пока не будет найден искомый элемент или не станет пустым интервал для поиска.

Графическая иллюстрация работы алгоритма



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

Графическая иллюстрация работы алгоритма



$l=0,\ K[l]=-2$

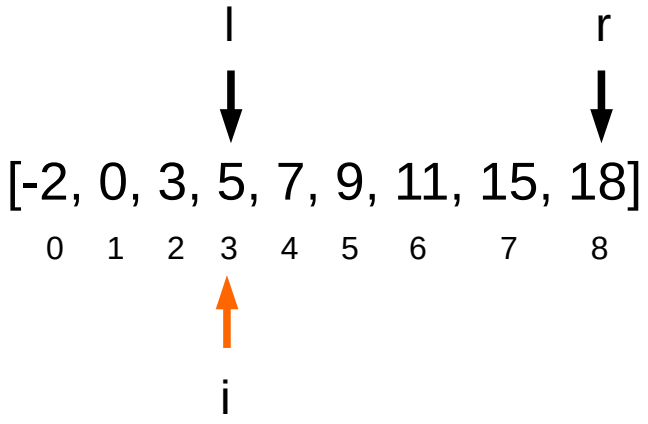
$r=8,\ K(r)=18$

На первом шаге, в качестве левой точки выбирается первый элемент последовательности в качестве правой точки последний элемент последовательности. Индекс вычисленный с помощью интерполяционной функции равен:

$$i(5)=\frac{(5-(-2))\cdot(0-8)}{-2-18}+0=2$$

На втором индексе стоит 3. Это меньше искомого следовательно новое значение $l=2+1=3,\ K[l]=5$

Графическая иллюстрация работы алгоритма



$l=3,\ \ K[l]=5$

$r=8,\ \ K(r)=18$

Вычисляем новый индекс:

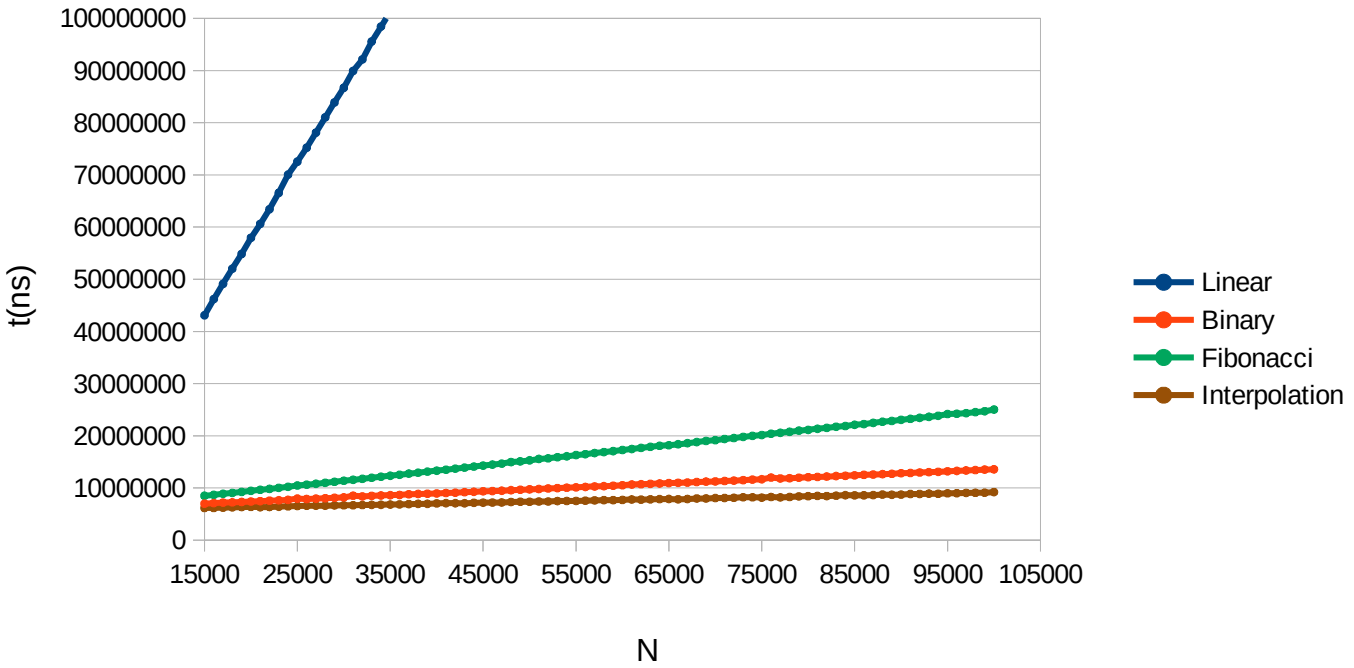
$$i(5)=\frac{(5-(5))\cdot(3-8)}{5-18}+3=3$$

На 3 индексе стоит 5. Это искомое. Поиск окончен.

Вычислительный эксперимент

Для оценки асимптотического поведения реализации этого алгоритма был проведен следующий вычислительный эксперимент.

Для одномерного массива из 100_000 элементов построена зависимость времени поиска от количества искомых элементов. Рассмотрен линейный,бинарный, поиск Фибоначчи, интерполяционный (для трех последних время сортировки массива также учитывалось). Для корректности каждый замер был повторен 100 раз и было взято усредненное время.



Область применения интерполяционного поиска

В случае равномерно распределенных данных в последовательности эффективность алгоритма интерполяционного поиска превышает эффективность бинарного. И только в случае неудачно подобранной интерполяционной функции или неравномерного распределения данных (например данные распределены экспоненциально, а интерполяционная функция выбрана линейной) его эффективность может снизится до линейной.

Как показывает практика чем больше размер последовательности, тем равномернее в ней распределены данные. На больших последовательностях интерполяционный поиск эффективнее бинарного. Это и определяет область его применения.

Python

Реализация алгоритма на Python

Python

Реализация алгоритма на Python

```
def interpolation_search(sequince, element):
    l = 0
    r = len(sequince)-1
    while sequince[l] < element and sequince[r] > element:
        if sequince[l] == sequince[r]:
            break
        index = (element - sequince[l]) * (l-r)//(sequince[l]-sequince[r]) + l
        if sequince[index] > element:
            r = index - 1
        elif sequince[index] < element:
            l = index + 1
        else:
            return index
    if sequince[l] == element:
        return l
    if sequince[r] == element:
        return r
    return -1
```

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

```
public static int interpolationSearch(int[] sequince, int element) {
     int l = 0;
     int r = sequince.length - 1;
          for (; sequince[l] < element && element < sequince[r];) {
               if (sequince[l] == sequince[r]) {
                    break;
               }
               int index = (element - sequince[l]) * (l - r) / (sequince[l] - sequince[r]) + l;
               if (sequince[index] > element) {
                    r = index - 1;
               } else if (sequince[index] < element) {
                    l = index + 1;
               } else {
                    return index;
               }
          }
          if (sequince[l] == element) {
               return l;
          }
          if (sequince[r] == element) {
               return r;
          }
     return -1;
}
```

Список литературы

1) Ананий Левитин. Алгоритмы: введение в разработку и анализ. : Пер. с англ. — М. : Издательский дом "Вильяме", 2006. — 576 с. : ил. — Парал. тит. Англ. ISBN 5-8459-0987-2. Стр. [240-241]
2) Дональд Кнут. «Искусство программирования, том 3. Сортировка и поиск» 2-е изд. М.: «Вильямс», 2007. С. 824. ISBN 0-201-89685-0. Стр. [449 -450]