Алгоритмы поиска

Поиск в линейных структурах

Суть метода- множество элементов просматривается последовательно в некотором порядке. Если в ходе просмотра множества найден искомый элемент, просмотр прекращается с положительным результатом, иначе алгоритм выдаст отрицательный результат.

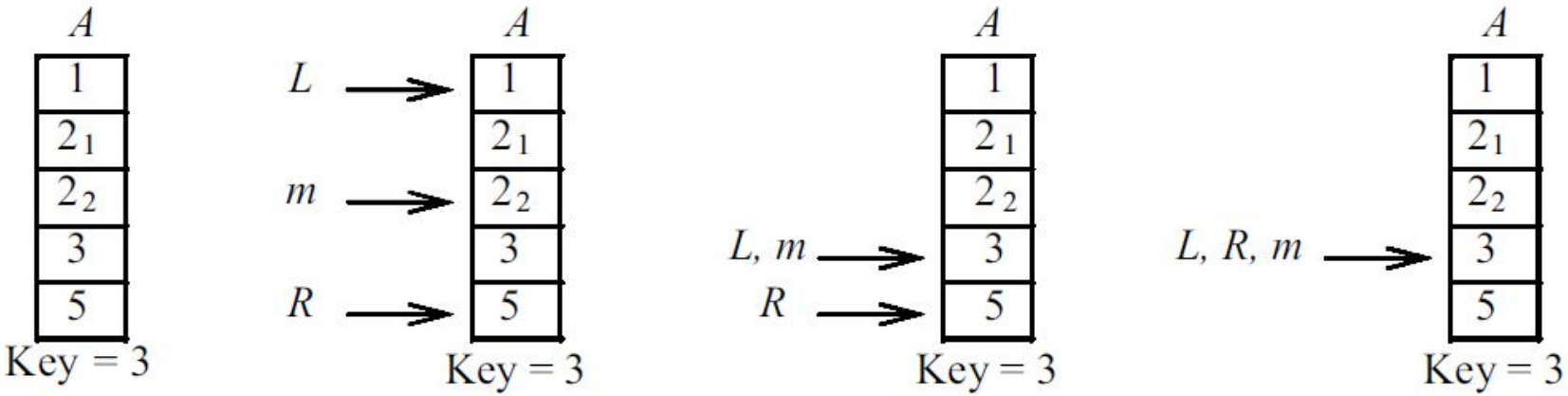
```
static bool LinearSearch(int [] Mas, int Key)
    {
            int i = 0;
            while (i < Mas.Length && Mas[i] != Key)
                i++;
            if (Mas[i] == Key) return true;
            else return false;
    }
 static bool LinearSearch(int [] Mas, int Key)
     {
             int i = 0;
             Array.Resize(ref Mas, Mas.Length + 1);
             Mas[Mas.Length-1] = Key;
             while (Mas[i] != Key)
                 i++;
             Array.Resize(ref Mas, Mas.Length - 1);
             if (i < Mas.Length ) return true;
             else return false;
     }
```

Сложность алгоритма пропорциональна O(n).

Бинарный поиск

Алгоритм предполагает, что множество хранится, как некоторая упорядоченная последовательность элементов, к которым можно получить доступ посредством индекса.

Пусть область поиска имеет границы L и R (L=0 и R=Mas.Lenght-1).
Находят индекс среднего элемента m=(L+R)/2.
Если Key>A[m], тогда искомая область сокращается – L=m+1 до R, иначе – R=m.
Пока L<>R область поиска сокращается вдвое.
Как только L=R, можно сделать вывод о результатах поиска.

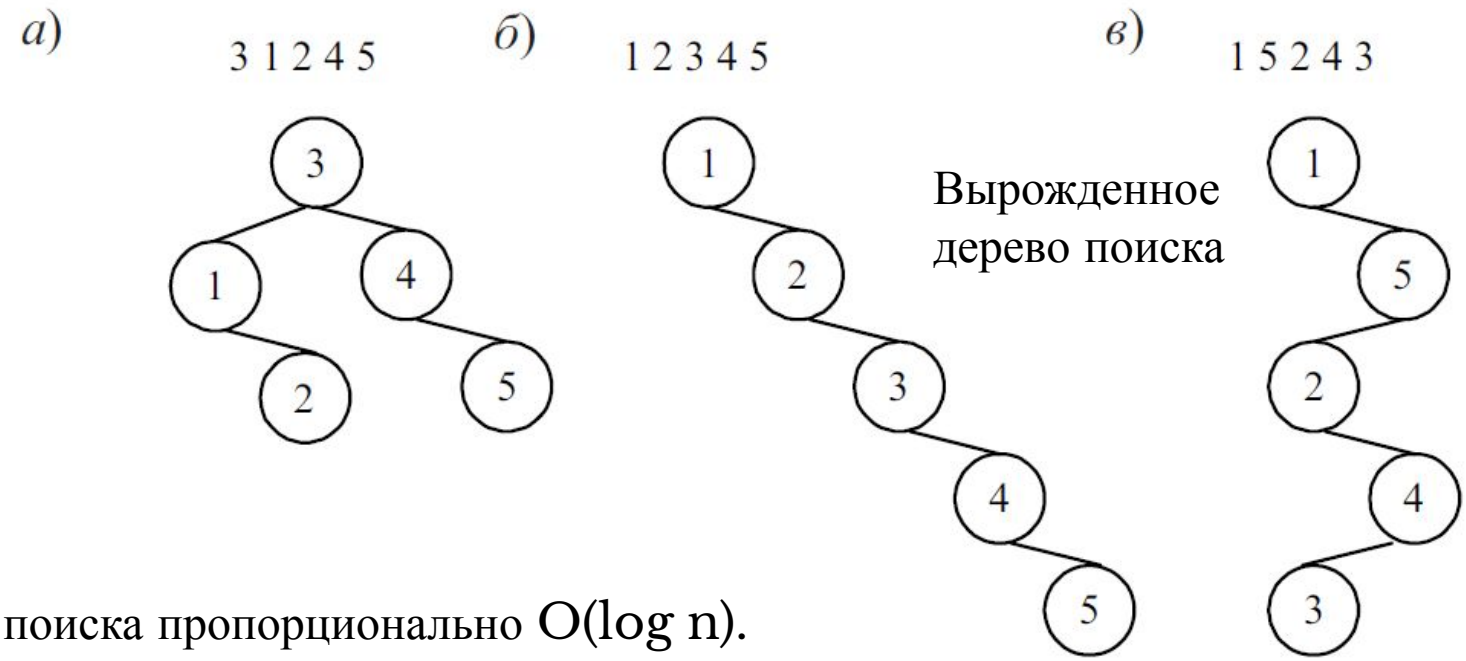


Сложность алгоритма пропорциональна O(log n).

Использование деревьев в задачах поиска

Двоичное дерево **упорядочено**, если для любой вершины x справедливо такое свойство: все элементы хранимые в левом поддереве, меньше элемента x, а все элементы, хранимые в правом поддереве, больше элемента x.

Если в дереве встречаются одинаковые элементы, то оно **частично упорядочено.**



Время поиска пропорционально O(log n).

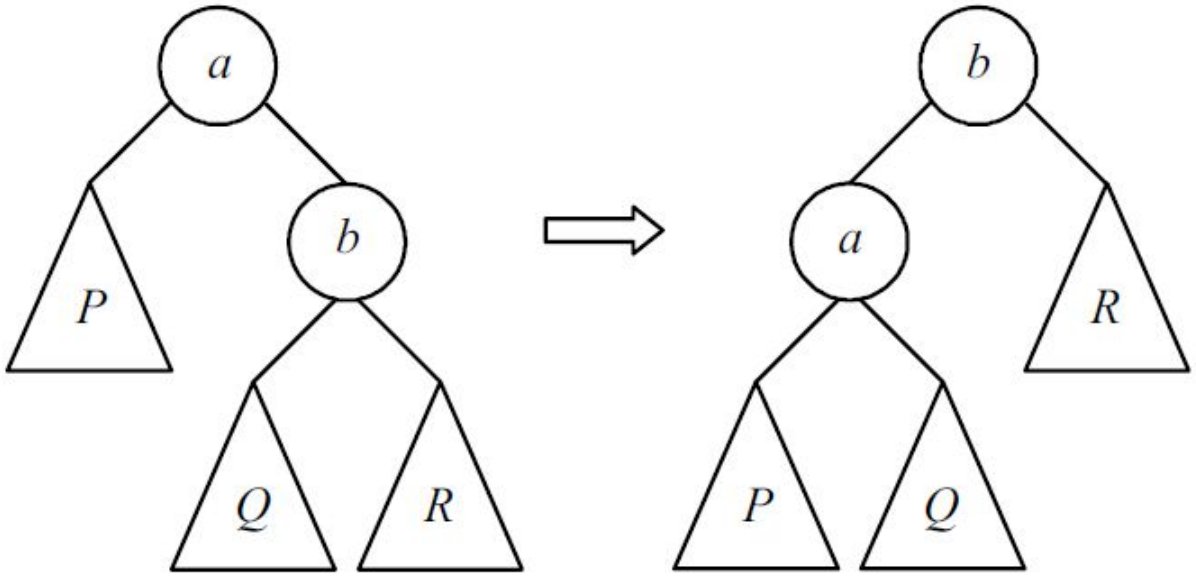
Время поиска пропорционально O(n).

Сбалансированные деревья

Идеально сбалансированным называется дерево, у которого для каждой вершины выполняется требование: число вершин в левом и правом поддеревьях различаются не более чем на 1.

Дерево считается **сбалансированным** по АВЛ (Г.М. Адельсон-Вельский и Е.М. Ландис) , если для каждой вершины выполняется требование: высота левого и правого поддеревьев различаются не более чем на 1.

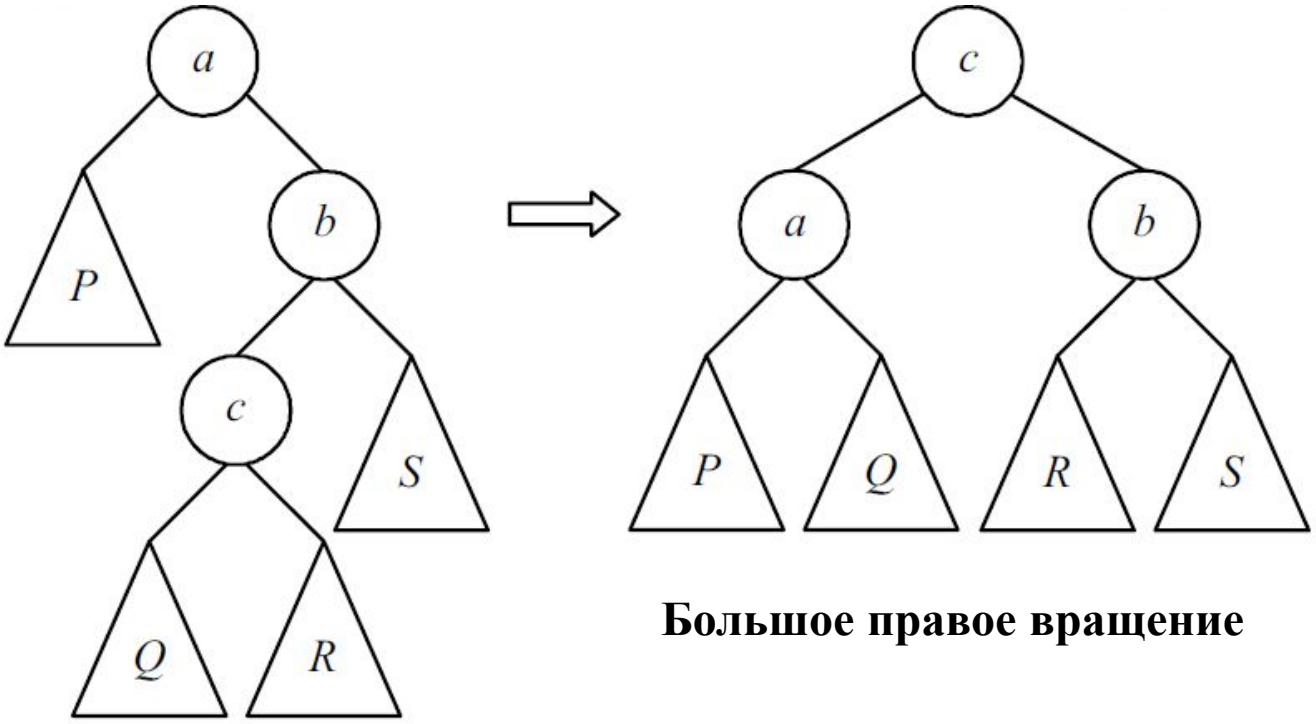
P<a<Q<b<R



Малое правое вращение

Сбалансированные деревья поиска

P<a<Q<c<R<b<S

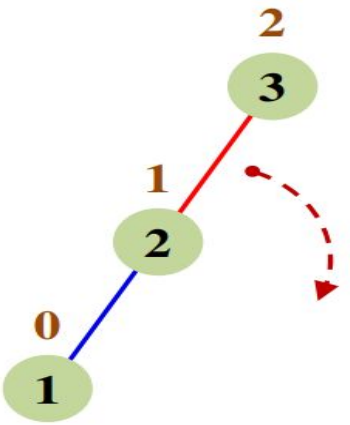


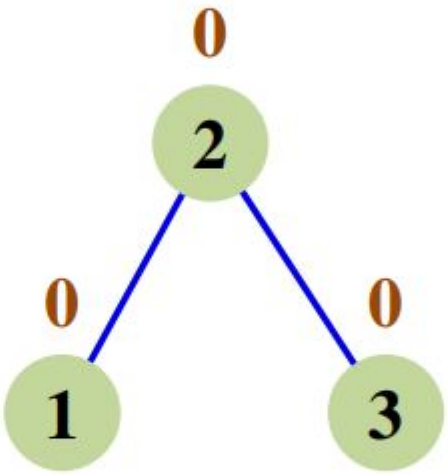
Большое правое вращение

Добавление нового элемента в сбалансированное дерево заключается в следующем:

1) Поиск по дереву.
2) Вставка элемента в место, где закончился поиск.
3) Восстановление сбалансированности дерева.

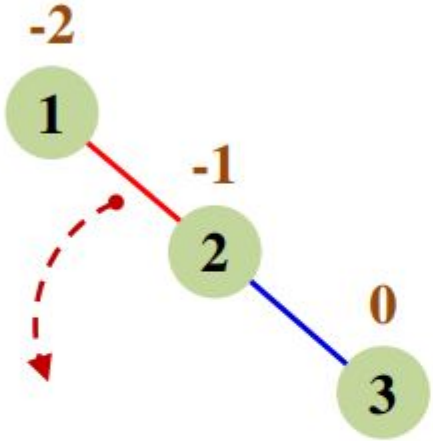
Правый поворот

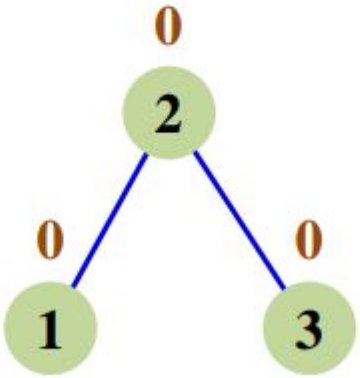




Дерево сбалансированно

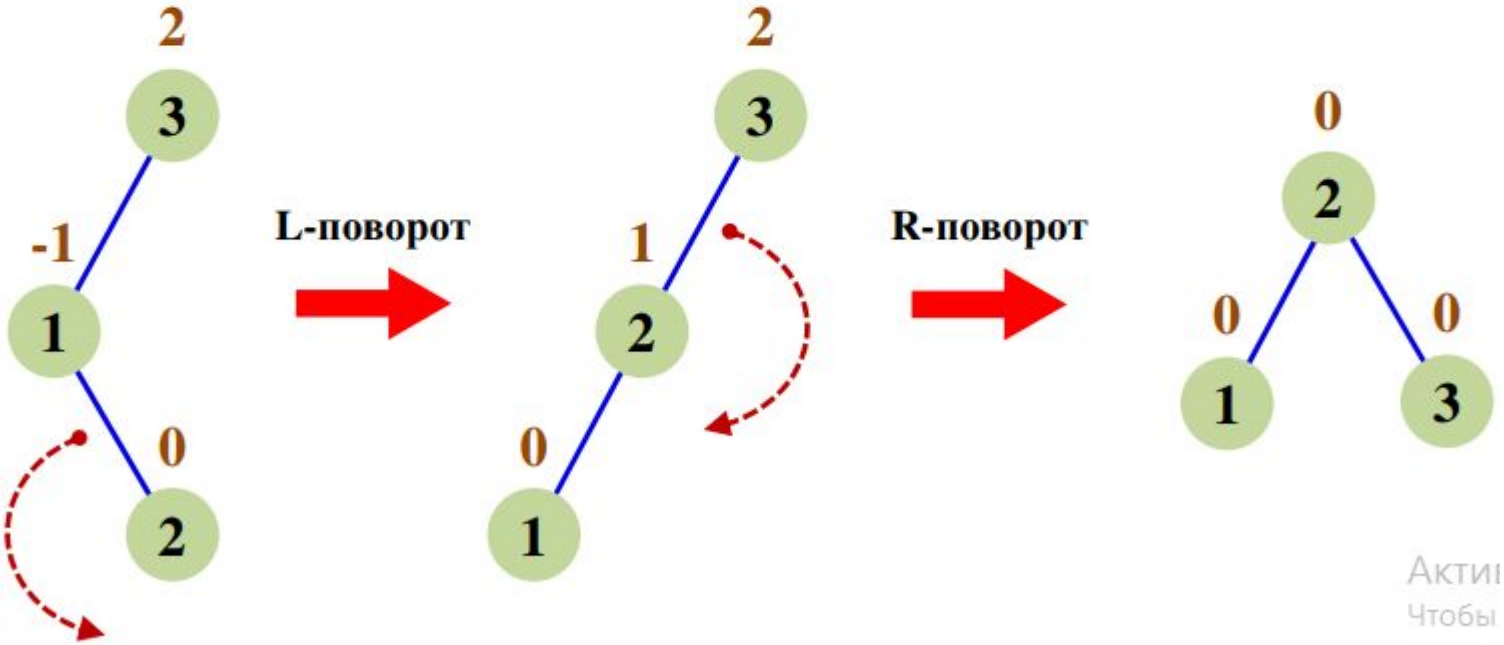
Левый поворот



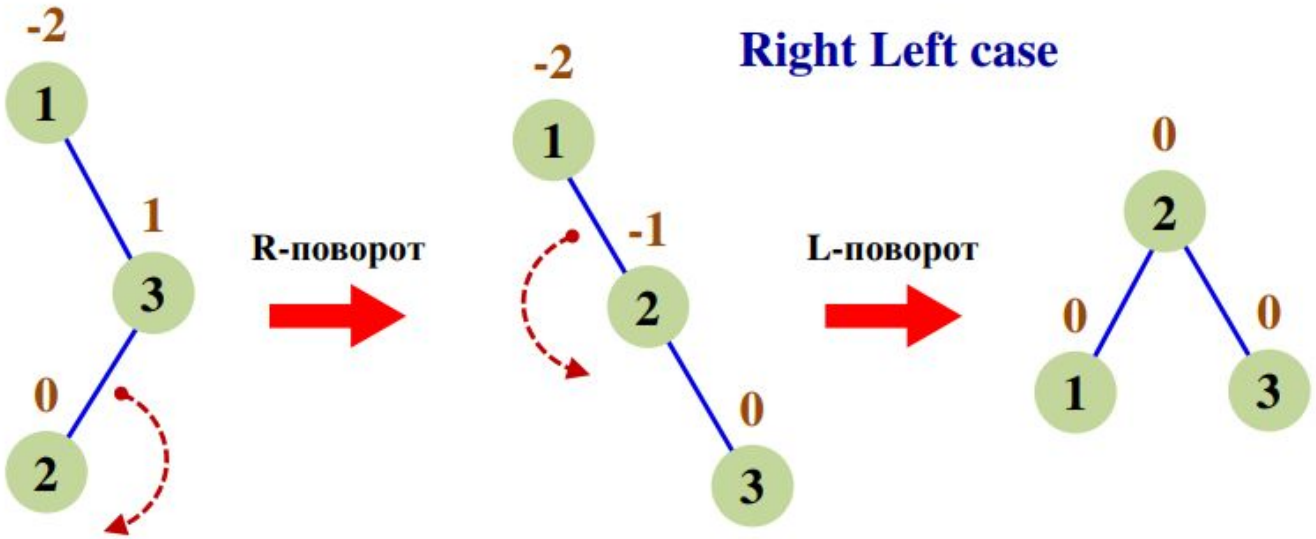


Дерево сбалансированно

Двойной левый -правый поворот

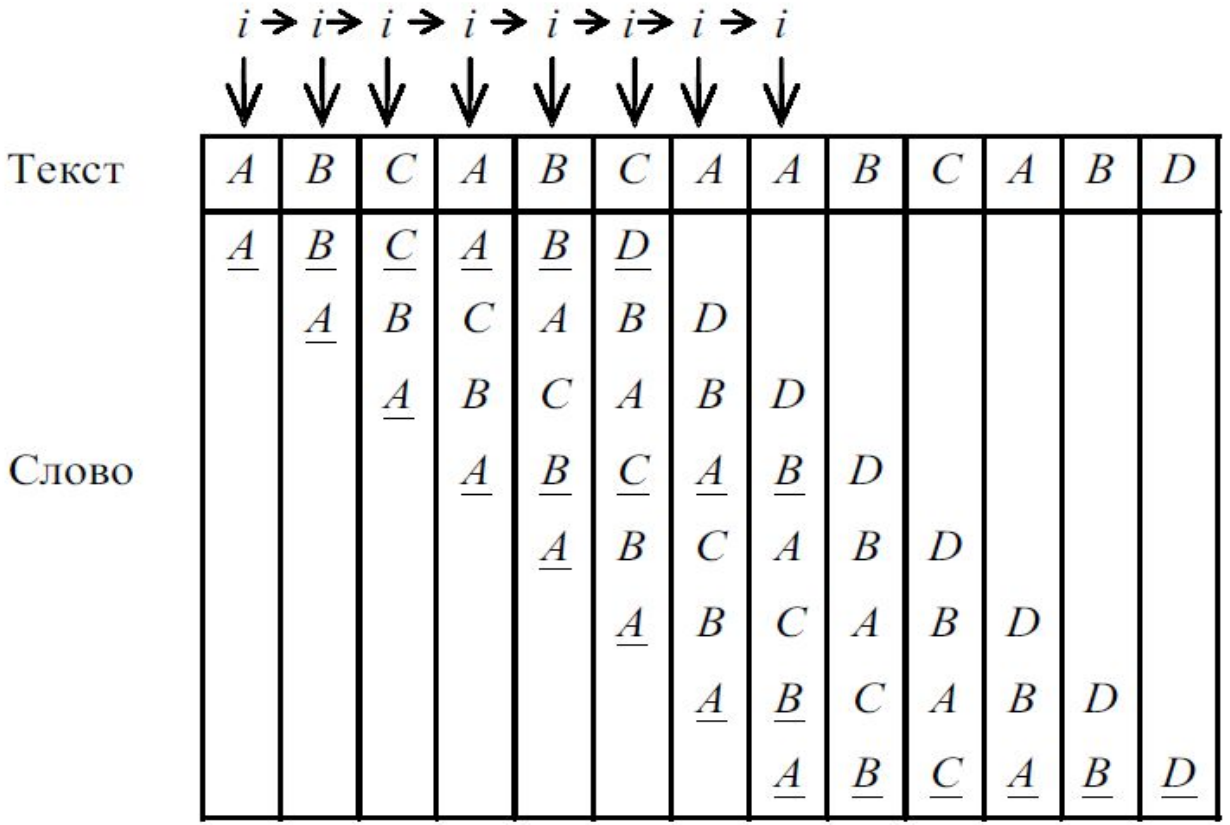


Двойной правый- левый поворот



Поиск в тексте. Прямой поиск

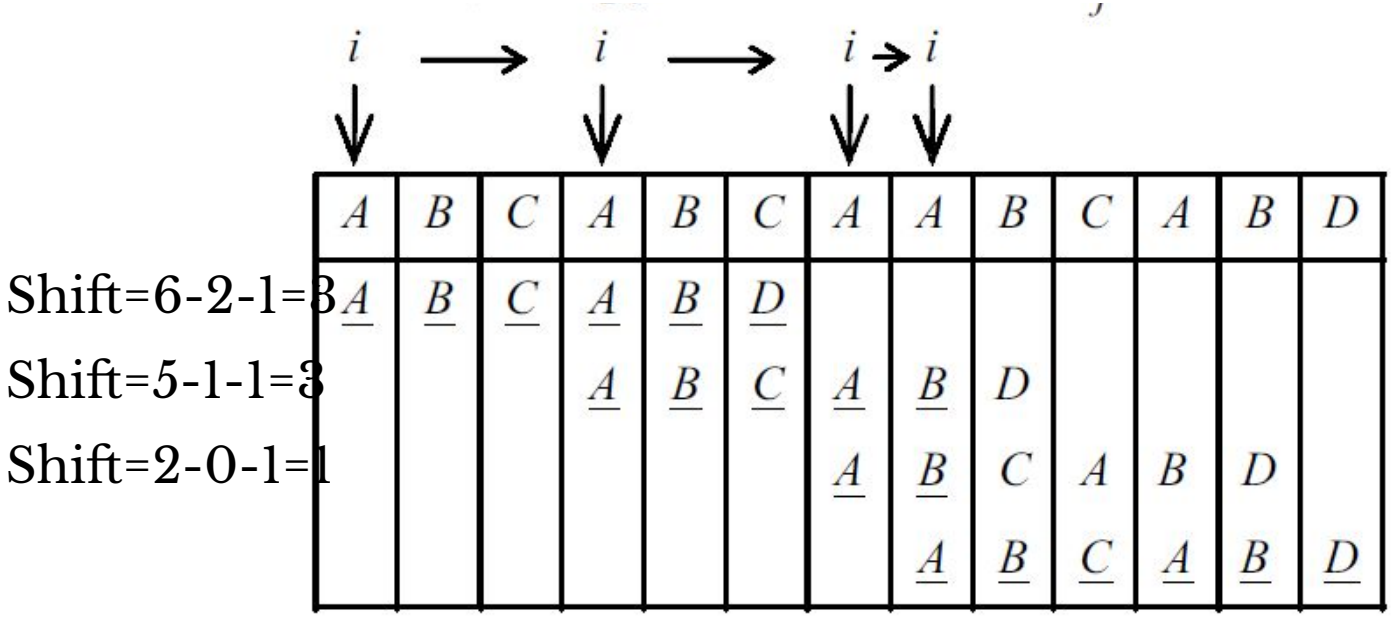
Суть метода- в начальный момент происходит сравнение первого символа теста с первым символом слова, второго символа текста со вторым символом слова и т.д. Если произошло совпадение всех символом, то фиксируется факт нахождения слова. В противном случае происходит сдвиг слова на одну позицию. И повторяется посимвольное сравнение.



Сложность алгоритма пропорциональна O((N-M)*M),
где N -длина текста , M-длина слова

Алгоритм Кнута, Мориса и Пратта

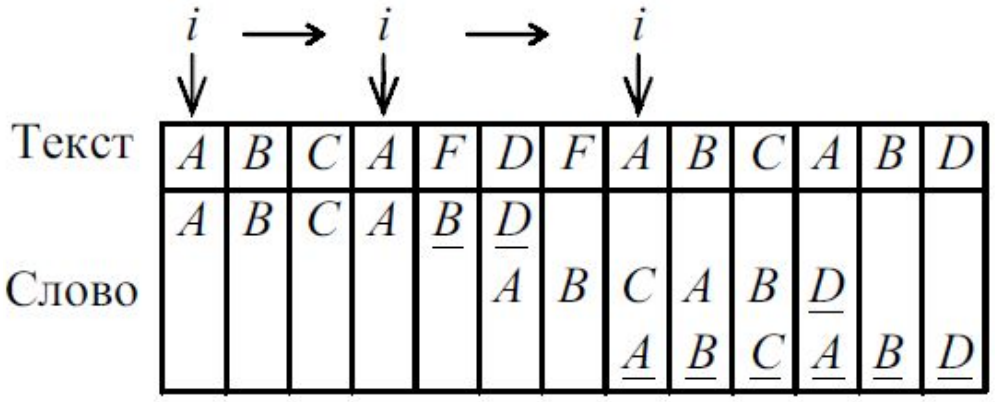
- Пусть j позиция в слове, содержащая первый несовпадающий символ.
- Величина сдвига Shift=j-LenSuff-1.
- LenSuff (суффикс)- размер самой длинной последовательности символов слова, предшествующих j, которая полностью совпадает с началом слова.



Алгоритм требует порядка O(N+M) сравнений символов ,
где N -длина текста , M-длина слова

Алгоритм Боуера и Мура

- Сравнение символов начинается с конца слова.
- Обнаружено расхождение между символом слова и текста.
- Символ в тексте который не совпал x.
- Слово сдвигаем вправо так, чтобы самый правый символ слова, равный x, оказался в той же позиции, что и символ текста x.
- Если несовпадающий символ текста x в слове вообще не встречается, то сдвигаем вправо так, чтобы ни один символ слова не накладывался на символ x.



Не совпало с "*F*", "*F*" нет в слове
Не совпало с "*A*", Shift ["*A*"] = 2

Алгоритм требует меньше O(N) сравнений символов , в самых благоприятных обстоятельствах число сравнений пропорционально O(N/M)
где N -длина текста , M-длина слова