**Алгоритм Бойера-Мура**

1.void BojerMoor(string str, int size, int Elem[256])//Функция бойера мура

2.{

3. int i;//переменная

4. for (i = 0; i < 256; i++)// бегаем по нашим элементам

5. Elem[i] = -1;//начинаем с - 1 все наши элементы

6. for (i = 0; i < size; i++)// функция для заполнения

7. Elem[(int)str[i]] = i;//заполняем значение крайнего символа

8. }

9.

10. vector <int> search(string txt, string pat)//функция для поиска подстроки в строке

11. {

12. int m = pat.size();//переменная для размера нашей строки

13. int n = txt.size();//переменная для размера нашего текста

14. vector<int> OUT;//вектор

15. int Elem[256]; //состоящий из таких элементов

16. BojerMoor(pat, m, Elem);//заполняем левыми элементами наш вектор

17. int s = 0; //будет для смещения такста

18. while (s <= (n - m))//цикл пока не дошли до конца

19. {

20. int j = m - 1;//пихаем в переменную предыдущее значение и так по кругу

21.

22. while (j >= 0 && pat[j] == txt[s + j]) // Продолжаем умеменьшать индекс 23. шаблона j, пока симвлоы текста и шаблона совпадают

24. j--;//уменьшаем

25. if (j < 0)//если меньше нуля то

26. {

27. OUT.push\_back(s);//Смещаем так что следующий символ в тексте совпадал с 28. последним

29. s += (s + m < n) ? m - Elem[txt[s + m]] : 1;//если встретили в конце

30. текста

31. }

32. else

33. s += max(1, j - Elem[txt[s + j]]);// положительный сдвиг где левый символ 34. в тексте совпадает с последним

35. }

36. return OUT;//возвращаем аут

Для начала рассмотрим вспомогательную функцию BojerMoor. Цикл на 4 строке проходится по 256 символам и имеет константное время. Далее цикл 6 проходится по всем элементам строки и имеет время m, т.к он идет по строке паттерна. Перейдем к основной функции на строке 16. Строки 12-14 нас не интересуют, т.к они выполняются за константное время. Как мы выяснили раньше, функция на 16 строке работает за O(m). Перейдем к циклу на 18 строке . Весь цикл на выполняется (n-m) раз, далее в нем используется вложенный цикл на 22 строке, который идет по шаблону. Остальные операторы имеют константное время, интереса они не представляют. Из теории следует, что общее время выполнения цикла на 16 строке составляет О(n). Как известно ранее, функция на 16 строке выполняется за O(m), значит суммарное время выполнения всей функции search составляет O(n+m).

Достоинства:

1. Алгоритм Бойера-Мура на хороших данных очень быстр, а вероятность появления плохих данных крайне мала. Поэтому он оптимален в большинстве случаев, когда нет возможности провести предварительную обработку текста, в котором проводится поиск.
2. На больших алфавитах (относительно длины шаблона) алгоритм чрезвычайно быстрый и требует намного меньше памяти, чем алгоритм Ахо-Корасик.

Недостатки:

1. Алгоритмы семейства Бойера-Мура не расширяются до приблизительного поиска, поиска любой строки из нескольких.
2. На искусственно подобранных неудачных текстах скорость алгоритма Бойера-Мура серьёзно снижается.