Boyer Moore Algorithm

Intelop View

Boyer Moore String Searching Function

1- Description:

This function makes use of the boyer moore string searching algorithm for searching for a substring inside a text string.
Help on arguments to be supplied: The first parameter is a text string of type char * or char[] The second parameter is the pattern to be searched of type char * or char[]
The third parameter is of type int * and here the number of occurrences of the pattern in the text string will be stored.
The fourth parameter is of type int * or int[] which is an array of a minimum size of the number of occurrences of the pattern inside the text string. The array supplied as the fourth parameter serves as a place to store values of the positions of occurrence of the pattern string in the text string for example if the third argument is c and the fourth is d *c will hold the number of occurrences of the pattern string in the text string d[0],d[1],....d[(*c)-1] will store the places where the pattern occurred in the main string Hope this will be enough.
This has been written for C++ and contains dynamic allocation of memory.
The int value that the function returns gives the no of comparisons performed

```
# include <iostream.h>
# include <string.h>
//These two include files are mandatory.



int bmsearch(char *text,char *pat,int *no,int *pos)
{
        int i,table[256];
        int len=strlen(pat);
        *no=0;
        int compcount=0;
        for(i=0;i<256;i++)table[i]=len;
        for(i=0;i<len;i++)table[pat[i]]=len-(i+1);
        int ptct=len-1;
        while(ptct<strlen(text))
        {
            int count=0;
            while(count<len)
            {
                if(text[ptct-count]!=pat[len-1-count]){compcount++;break;}
                else count++;
            }
            if(count==len)
            {
                (*no)++;
                *(pos+(*no)-1)=(ptct-count+1);
                ptct+=len;
            }
            else
            {ptct+=(table[text[ptct-count]]-count);}
        }
        return compcount;
```

BM TEST Program

This is a test program for the file bm.h just for you to see the working of the function

```
# include "bm.h"
# include <conio.h>
# include <stdio.h>

main()
{
        char text[300],pat[100];
        int no,count,pos[20],i;
        clrscr();
        cout<<"\n\nEnter the text string:";
        gets(text);
        cout<<"Enter the pattern string:";
        gets(pat);
        count=bmsearch(text,pat,&no,pos);
        cout<<"\n\nPattern string occoured "<<no<<" times.";
        cout<<"\n"<<count<<" comparisons required";
        cout<<"\nPositions of occourence:\n";
        for(i=0;i<no;i++)
                cout<<pos[i]<<"\n";
        cout<<"Thank you.";
        return 0;
}
```