hw1_0316051 林祐同

1.實作方式:

演算法:

利用 failureFunction 算出下一個需要比對的 key，逐一與各字串做比對。

```
//************************************************
//failure function
void failureFunction(string& p)
{
    f[0]=-1;
    int size=p.length();
    for(int j=1;j<size;j++)
    {
        int i=f[j-1];
        while(p[j]!=p[i+1] && i>=0)i=f[i];
        if(p[j]==p[i+1])f[j]=i+1;
        else f[j]=-1;
    }
}
```

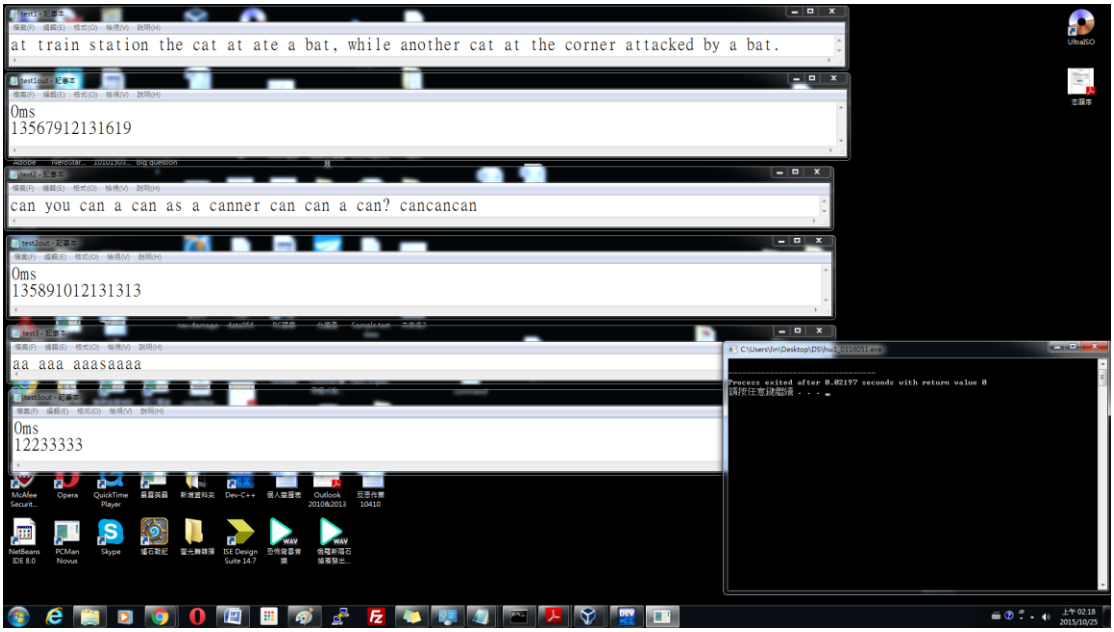
```
failureFunction(pat); // ->f[1000000]
while(fin>>in)
{
    time++;
    //fin>>c;
    int i=0;
    for(int a=0;a<in.length();)
    {
        while(in[a]==pat[i] && pat.length() != i)
        {
            a++;
            i++;
        }
        if(pat.length() == i)
        {
            i=f[i-1]+1;
            ss<<time;
            magic_number = magic_number + ss.str();
            ss.str("");
            ss.clear();
        }else if(i==0)
        {
            a++;
        }else
        {
            i=f[i-1]+1;
        }
    }
}
```

實作 failureFunction 是參考老師上課的講義，function f(j)，f(j)=-1 且 f(j)=與前面最長相同的數目(從 0 開始算)，若前面沒有符合的或相同中斷則 f(j)=-1，透過這想法即可先算出 pattern 的 failureFunction，可以少掉多餘的比較，從 f()裡找出下一個需要比對的 key。

執行過程:

從.txt 逐一讀入字串(開始計算時間)，然後與 pattern 一個一個字元逐一比對，友符合的字元就往後比對，若全部符合就將數字加入 magic_number 也就是要 output 到.txt 的答案，並且去 failureFunction 找到下一個要比對的 key 字元，若沒有完全符合則去 failureFunction 找到下一個要比對的 key 字元。全部算完後(計算時間結束)，再把執行時間和 magic_number output 到.txt。

2. Sample 測試結果:



3.心得:

failureFunction 和 matching 的想法都是參考老師講義來寫的，理解 failureFunction 後再想辦法用 failureFunction 來加快比對的速度。最後遇到一個沒預料到的問題，因為 output 的時間要先印出來，本來我是一有比對成功就 fout，但似乎不符合結果，所以就利用 `stringstream` 把 **`time`** 轉成 **`string`** 與 magic_number 作串接，到最後再一起 output 到.txt 。