**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：11022: String Factoring**

**解題者：徐睿鍾**

**解題日期：2022年6月2日**

**題意：**

給好幾組字串(每組字串不超過80個字元)，輸入\*則程式停止。將字串縮寫後，輸出最少需要幾個字元才能表示該字串。

**題意範例：**

ABB => A(B)2 => 2

ABB可以變成A(B)2，只剩A跟B，因此輸出2

DOODOO => (D(O)2)2 => 2

ABBABABAB => A(B)2(AB)3 => 4

ABBABABBABABAB

這個例子可以有很多種分法

例如:A(B)2(AB)2B(AB)3 => 7

(A(B)2AB)2(AB)2 => 6

要輸出最小，因此輸出6

**解法：**

使用dp解決這問題，dp公式為:

dp[a][b] = min( dp[a][b] , dp[a][i]+dp[i+1][b] ) ，a ≤ i < b

dp[a][b]代表第a個字元到第b個字元縮寫後的最小值

dp[a][a] = 1

dp[a][b]的求法分為兩種

(1.) dp[a][i]+dp[i+1][b]

//第1種的解釋以ABBABABBABABAB為例

dp[1][14] = dp[1][10] + dp[11][14] = 4+2 = 6

(2.) 字元a到字元b可以化簡成比第1種更少的字元數

//第2種的解釋以ABBABABBAB為例

利用第1種求法可以得出很多種可能，但(A(B)2AB)2 = 4

才是最小值，因此我們可以得出當這個字串切割成可以平分的情況時，且平分後的字串也相同時，可能會有更小的值(例如ABBABBABB也可分成ABB ABB ABB)

**解法範例：**

無

**討論：**

(1)這題實作上從大的開始計算，但小範圍的會先計算完。

(2)建立一個2維陣列，記錄已算過的值(例如memory[12][15]代表第12個字元到第15個字元的值)，因 為小範圍的值會先計算完，且每次計算完的值必是最小的(無需更新)。再呼叫dp(a,b)時只要memory[a][b]與初始化的值不同時，直接回傳memory[a][b]即可。//初始化的 值可設為2147483647

**程式：**

#include <iostream>

#include <string>

#define MAX 2147483647

using namespace std;

string input;

string blank=" ";

int memory[81][81];

int dp(int l,int r){

if(memory[l][r]!=MAX)

return memory[l][r];

if(l==r)

return 1;

for(int i=l;i<r;i++){

if( (dp(l,i) + dp(i+1,r) ) < memory[l][r] )

memory[l][r]=(dp(l,i) + dp(i+1,r));

}

int sublen=r-l+1;

int k,j;

for(int i=1;i<sublen;i++) {

if(sublen%i==0) {

for(k=l,j=0;k<=r;k++) {

if(input[k]!=input[l+j])

break;

j++;

if(j>=i)

j=0;

}

if( k==r+1 && (dp(l,l+i-1) < memory[l][r]) )

memory[l][r]=dp(l,l+i-1);

}

}

return memory[l][r];

}

int main()

{

int n;

while(cin>>input){

if(input[0]=='\*')

break;

n=input.length();

input=blank+input; //input[0] useless

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

memory[i][j]=MAX; //initial dp

cout<<dp(1,n)<<endl;

}

return 0;

}