**[拓展KMP求回文串](https://www.cnblogs.com/linhaitai/p/9921763.html)**

****HDU 3613 Best Reward(扩展KMP:回文判断)****

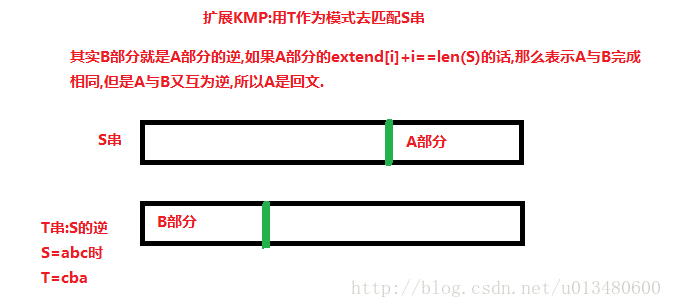
**[http://acm.hdu.edu.cn/showproblem.php?pid=3613](http://acm.hdu.edu.cn/showproblem.php?pid=3613" \t "https://blog.csdn.net/u013480600/article/details/_blank)**

题意:给你一个字符串,要你把这个字符串分成两段,并使得被分开的两段价值和最大.一个串如果是回文,那么它的价值就是所有字符的价值和,否则价值为0.

分析:

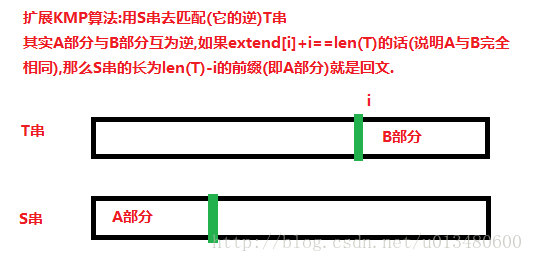
       首先原始串为S,将S逆转得到串T.(S=abcaaa,那么T=aaacba).

       S串的后缀回文:即S串中区间[i,n-1]的串是不是回文?



将S作为主串,T串用扩展KMP算法去匹配S,extend1[n]数组保存匹配结果.****如果extend1[i]+i==n时(n为S的长),那么以S[i]为首字符一直到底n-1位置的串是回文串,否则不是.(自己举个例子验证一下)****

       S串的前缀回文:即S串中区间[0,i-1]的串是不是回文?



将T作为主串,S串用扩展KMP算法去匹配T,extend2[n]数组保存匹配结果.如果extend2[len-i]+len-i==n时(n为S的长),那么以S[i-1]为尾字符一直到0位置的串是回文串,否则不是****.(自己举个例子验证一下)****

****仔细思考下上面的模型.****

AC代码:

#include <iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

using namespace std;

const int MAXN=500000+1000;

char S[MAXN],T[MAXN];

int n;*//串S与串T的长都为n*

int next[MAXN],extend1[MAXN],extend2[MAXN];

int val[30],sum[MAXN];

void EKMP(char \*S,char \*T,int \*next,int \*extend)

{

int j=0;

int k=1;

while(j+1<n && T[0+j]==T[1+j])

j++;

next[1]=j;

for(int i=2;i<n;i++)

{

int len=k+next[k]-1,L=next[i-k];

if(L<len-i+1)

next[i]=L;

else

{

j=max(0,len-i+1);

while(i+j<n && T[i+j] == T[0+j])

j++;

next[i]=j;

k=i;

}

}

j=0;

while(j<n && S[0+j]==T[0+j])

j++;

extend[0]=j;

k=1;

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

{

int len=k+extend[k]-1,L=next[i-k];

if(L<len-i+1)

extend[i]=L;

else

{

j=max(0,len-i+1);

while(i+j<n && j<n && S[i+j]==T[j])

j++;

extend[i]=j;

k=i;

}

}

}

int main()

{

int kase;

scanf("%d",&kase);

while(kase--)

{

for(int i=0;i<26;i++)

scanf("%d",&val[i]);

scanf("%s",S);

n=strlen(S);

for(int i=0;i<n;i++)

{

T[i]=S[n-1-i];

if(i==0)sum[i]=val[S[i]-'a'];

else sum[i]=sum[i-1]+val[S[i]-'a'];

}

T[n]=0;

EKMP(T,S,next,extend1);

EKMP(S,T,next,extend2);

int max\_s=-1e8;

for(int i=1;i<n;i++)*//i表示前半段的长度*

{

int sc=0;*//分数*

*//求前缀[0,i-1]分数*

if(extend1[n-i]+n-i==n)

{

sc+= sum[i-1];

}

*//求后缀[i,n-1]的分数*

if(extend2[i]+i==n)

{

sc+= sum[n-1]-sum[i-1];

}

max\_s=max(max\_s,sc);

}

printf("%d\n",max\_s);

}

return 0;

}