经典DP。影响决策的是字符串的内容。而添加和删除字符本质上是一样的，我们不管选择哪一种都可以，所以只需要取两者中费用最小的。

如果你在开头删一个字母可以变成回文那么完全可以在结尾添加这个字母使其变成回文

状态转移方程就是：

if(s[i]==s[j]) dp[i][j] = dp[i+1][j-1];  
        else dp[i][j] = min(dp[i+1][j]+w[s[i]-'a'],dp[i][j-1]+w[s[j]-'a']);

其中dp[i][j]表示从字符串的i-j段为回文串时的最小值。

值得注意的是i需要反向枚举，j需要正向枚举，这是由状态转移的顺序决定的。因为dp[i][j]需要借用i+1时的结果和j-1时的结果。

#include <cstdio>

#include <algorithm>

#include<iostream>

#include<cstring>

using namespace std;

int w[100];

int dp[2010][2010];

int main()

{

// freopen("input.txt","r",stdin);

int n,m,a,b;

char c;

string s;

cin>>n>>m;

cin>>s;

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

{

cin>>c>>a>>b;

a=min(a,b);

w[c-'a']=a;

}

for(int i=m-1;i>=0;i--)

for(int j=i+1;j<m;j++)

if(s[i]==s[j])

dp[i][j]=dp[i+1][j-1];

else

dp[i][j]=min(dp[i+1][j]+w[s[i]-'a'],dp[i][j-1]+w[s[j]-'a']);

cout<<dp[0][m-1]<<"\n";

return 0;

}