USACO OPEN07 Problem 'cheappal' Analysis

by Richard Peng

This problem can be done using dynamic programming (DP). First observe that the cost of inserting/removing characters can be combined into one cost since inserting another of of the same character at the mirrored poisition would have the same effect as deleting the character. So we can left cost[ch] be the minimum of the two costs for inserting/deleting character ch.

Now consider the dp state of [l,r] which represents a substring. For each state, we try to minimize the cost of changing that substring into a palindrome. Then the state transition function would be:

DP[l,r]=min{DP[l+1,r]+cost[s[l]]. ('pair up' left most character)
DP[l,r+1]+cost[s[r]], ('pair up' right most character)
DP[l+1,r-1] (only if the two end characters are equal, aka. s[l]=s[r]}

The state transition takes O(1) time while there are a total of O(M^2) states. So the algorithm runs in O(M^2) time.

Test Data

All data were generated randomly while specifying N and M.
It's intended that a brute-force search would get cases 1-4, anything O(M^3) or faster would get cases 1-8 while the O(M^2) solution is required to get full points.

Below is the solution of China's Yang Yi:

```
#include <stdio.h>
#define MAXN 2001

FILE *in = fopen("cheappal.in","r");
FILE *out = fopen("cheappal.out","w");

int n, m, is[26], de[26], dp[MAXN][MAXN];
char str[MAXN + 1];

int main () {
    int i, j, a, b;
    char ch;

    fscanf (in, "%d%d%s", &n, &m, str);
    for (i = 0; i < n; i ++) {
        do fscanf(in,"%c",&ch);
            while (ch <= ' ');
        fscanf(in,"%d%d",&a,&b);
        is[ch - 'a'] = a;
        de[ch - 'a'] = b;
        }
    for (a = 0; a < m; a ++)
        for (i = 0; i + a < m; i ++) {
            j = i + a;
            if (i == j || i + 1 == j && str[i] == str[j])
                dp[i][j] = 0;
            else {
                dp[i][j] = (dp[i+1][j] + (is[str[i]-'a'] <? de[str[i]-'a']))
<? (dp[i][j-1] + (is[str[j]-'a'] <? de[str[j]-'a']));
                if (str[i] == str[j])
                    dp[i][j] <?= dp[i+1][j-1];
                }
            }
    fprintf(out,"%d\n",dp[0][m - 1]);
    fclose(in);
    fclose(out);
    return 0;
}
```