USACO Traingate 'twofive' Analysis

by Artur Dmowski

Consider a partially filled grid with the first K letters of the alphabet. We can only place the (K+1)th letter in a cell with all the cells above it and to its left already filled. Otherwise, we will eventually fill the missing cells with the letter that follows which would make the ordering wrong. For example, for K=6:

```
A B D * _
C E * _ _
F * _ _ _
* _ _ _ _
_ _ _ _ _
```

We can place the 7th letter (denoted G) on any of the cells denoted by '*'. Using both this information and memoizsation, we can quickly calculate number of ways to fill rest of the grid for a partially filled grid. We can easily adjust this to accept a grid containing any K letters (not necessarily first K), by checking if the letter we wish to place is bigger than the adjacent cells above it and to its left.

To find the Nth grid, we "fix" all letters in the output string. We need to keep total number of grids we skipped (say T). We "fix" every letter, by setting it to the next unused letter and see if number of grids we skipped (call it C + T exceeds N. If it does, go to the next letter, otherwise add C to T and continue fixing. Notice that we can calculate C by calculating number of ways to fill the rest of the grid.

Determining number of a grid is analogous. Instead of fixing letter until we exceed N, we just fix every letter until we get to the corresponding letter in the input string. T is the number we're looking for.

Here is the C++ implementation:

```
#include <stdio.h>
#include <string.h>
#define MAX 5
#define ALL 25
using namespace std;

int N;
char mode; /* mode, we're looking for string or for a number */
char str[ALL+1]; /* output string */
int nways[MAX+1][MAX+1][MAX+1][MAX+1][MAX+1]; /* number of ways to fill the rest of the
grid */
int used[ALL]; /* is the letter used */
int maxr[MAX]; /* the biggest letter in a row */
int maxc[MAX]; /* the biggest letter in a column */

/* calculate number of ways to fill the grid with 'a' letters in the 1st row, 'b' in the
2nd etc. */
/* 'l' is the first letter we try to place */
int calc(int a, int b, int c, int d, int e,int l) {
    int cur = 0;
    if( nways[a][b][c][d][e] )
        return nways[a][b][c][d][e];

    if( used[l] )
        return nways[a][b][c][d][e] = calc(a,b,c,d,e,l+1);

    if( a < 5 && l > maxr[0] && l > maxc[a] ) cur += calc(a+1,b,c,d,e,l+1);
    if( a > b && l > maxr[1] && l > maxc[b] ) cur += calc(a,b+1,c,d,e,l+1);
    if( b > c && l > maxr[2] && l > maxc[c] ) cur += calc(a,b,c+1,d,e,l+1);
    if( c > d && l > maxr[3] && l > maxc[d] ) cur += calc(a,b,c,d+1,e,l+1);
    if( d > e && l > maxr[4] && l > maxc[e] ) cur += calc(a,b,c,d,e+1,l+1);

    return nways[a][b][c][d][e] = cur;
}
/* place new letter on the grid and prepare arrays for running calc() */
void prepare( int letter, int r, int c ) {
    memset( nways, 0, sizeof(nways) );
    nways[5][5][5][5][5] = 1;
    maxr[r] = maxc[c] = letter;
    used[letter] = 1;
}
/* find the word from a number */
void find_word( int N ) {
    int l,a,cur=0,in[MAX]={0},tmp;
    char str[ALL+1] = {0};
    memset( maxr, -1, sizeof(maxr) );
    memset( maxc, -1, sizeof(maxc) );
    memset( used,  0, sizeof(used) );
    for( a = 0; a < ALL; a++) {
        in[a/5]++;
        /* go to the next letter */
        for( l = 0; l < ALL; l++ )
            if( !used[l] && l > maxr[a/5] && l > maxc[a%5] ) {
                prepare( l, a/5, a%5 );

                tmp = calc(in[0],in[1],in[2],in[3],in[4],0);

                /* stop when too much was skipped */
                if( cur + tmp >= N ) break;
                cur += tmp;
                used[l] = 0;
            }
        str[a] = char(l+'A');
    }
    printf("%s\n", str);
}
/* find number of a grid */
void find_num(char *str) {
    int l,a,num=1,in[MAX]={0};
    for( a = 0; a < ALL; a++ ) {
        in[a/5]++;
        for( l = 0; l < int(str[a]-'A'); l++ )
            if( !used[l] && l > maxr[a/5] && l > maxc[a%5] ) {
                prepare( l, a/5, a%5 );
                num += calc(in[0],in[1],in[2],in[3],in[4],0);
                used[l] = 0;
            }
        used[l] = 1; maxr[a/5] = maxc[a%5] = l;
    }
    printf("%d\n", num);
}
int main() {
    scanf("%c", &mode);
    if( mode == 'N' ) {
        scanf("%d", &N);
        find_word(N);
    }
    else {
        scanf("%s", &str);
        find_num(str);
    }
}
```