dictionary (原题bzoj1819)

建好trie树,打好 ,然后寻找题目所说的三种情况即可.

Install (原题bzoj2427)

由于是n个点n条边,所以是基环树森林,我们通过tarjan缩环后(环是捆绑选择的),就是森林

建立一个虚拟父亲后,就是一棵树了,然后就是经典的树型01背包问题了:f[i][j]表示i的子树花费j的体积产生的最大收益(如果选了i点就有值,不然就是0);

转移,对于以i为根的子树:

首先先不选i,然后和儿子的子树的收益进行合并:f[x][j]=max(f[x][j],f[x][k]+f[y][j-k]);

然后跟所有儿子搞完后,就来考虑选自身:

如果选不了自己的话,dp值为0,否则就直接加上(因为如果选不了i,那么i的子树中所有贡献都失效)

Paint(原题bzoj1296)

dp[i][j][k][0/1/2]表示第i行第j列一共粉刷了k次，0/1/2分别表示当前格子没有涂色/涂了错的颜色/涂了对的颜色， 然后我们考虑逐格转移：

当j=1也就是出于每行的第一个位置时，我们要考虑上一行的最后一个位置， 即

dp[i][j][k][0]=max(dp[i-1][m][k][1],max(dp[i-1][m][k][2],dp[i-1][m][k][0]));

dp[i][j][k][1]=max(dp[i-1][m][k-1][2],max(dp[i-1][m][k-1][1],dp[i-1][m][k-1][0]));

dp[i][j][k][2]=max(dp[i-1][m][k-1][2],max(dp[i-1][m][k-1][1],dp[i-1][m][k-1][0]))+1;

其余位置要考虑这个格子颜色是否和前一个格子的颜色相等，如果相等，就有

dp[i][j][k][2]=dp[i][j-1][k][2]+1;

可以直接接上

dp[i][j][k][1]=max(dp[i][j-1][k][1],dp[i][j-1][k-1][0]);

前面涂错或不涂

dp[i][j][k][0]=max(dp[i][j-1][k][0],dp[i][j-1][k][1]);

前面涂错或不涂 如果不相等,

dp[i][j][k][2]=max(dp[i][j-1][k-1][2],max(dp[i][j-1][k][1],dp[i][j-1][k-1][0]))+1;

前面可能有三种情况

dp[i][j][k][1]=max(dp[i][j-1][k][2],dp[i][j-1][k-1][0]);

涂对或不涂

dp[i][j][k][0]=max(dp[i][j-1][k][0],dp[i][j-1][k][2]);

涂对或不涂

可以用滚动数组压掉第一维，这样空间复杂度是O(nT),时间复杂度是O(nmT)，还是可以过的