H: Beans on the Grid

各マスについて、そのマスに豆があるときの grundy 数を求めたいです。
行ごとに考えます。

[見ている行に皿のないマスが存在する、すなわち # が存在する場合]

$G[i][j]$ をマス (i, j) に豆があるときの grundy 数とし、$L[i][j]$ を「豆がマス (i, j) にあり、かつ豆が左隣のマスを通ったことがあるときの grundy 数」とし、$R[i][j]$ を「豆がマス (i, j) にあり、かつ豆が右隣のマスを通ったことがあるときの grundy 数」と定義します。

このとき、$G[i][j]$ は $G[i+1][j], L[i][j+1], R[i][j-1]$ から計算でき、$L[i][j]$ は $L[i][j+1], G[i+1][j]$ から計算でき、$R[i][j]$ は $R[i][j-1], G[i+1][j]$ から計算できます。状態数も $3*H*W$ 個なので十分高速に計算できます。

[見ている行に皿のないマスが存在しない、すなわち # が存在しない場合]

この場合は上記の場合と同様に状態を持とうとすると、1 週して同じマスに戻ってきてしまうかもしれません。そこで、$l[i][j][k]$ を「豆がマス i, j にあり、かつ左側を k マス通ったときの grundy 数」と定義し、$r[i][j][k]$ を「豆がマス i, j にあり、かつ右側を k マス通ったときの grundy 数」と定義します。k は 1 以上 W 以下である必要があります。

このとき、 $G[i][j]$ は $G[i+1][j], l[i][j+1][1], r[i][j-1][1]$ から計算でき、$l[i][j][k]$ は $l[i][j+1][k+1], G[i+1][j]$ から計算でき、 $r[i][j][k]$ は $r[i][j-1][k+1]$、$G[i+1][j]$ から計算できます。

ただし、このままではまだ状態数が $O(H*W*W)$ 個と非常に多く、制限時間に間に合いません。そこで、$l[i][j][k]$ および $r[i][j][k]$ の計算が高速化できないか考えます。どちらも同様の方法で考えられるので、特に $l[i][j][k]$ について注目します。上述した通り、$l[i][j][k]$ は $G[i+1][j], l[i][j+1][k+1]$ の mex によって求められます。これは言い換えると、$l[i][j+1][k+1]$ の値を $G[i+1][j]$ の値によって変換したことになります。

例えば、$G[i+1][j] = 0$ であったとき、$l[i][j+1][k+1]$ の値が $0, 1, 2, 3$ のそれぞれの場合について、$l[i][j][k]$ の値はそれぞれ $1, 2, 1, 1$ となります。

さて、このように $l[i][j][k]$ の計算の仕方を数字への操作に落とし込んだところで、操作同士の積を考えます。例えば、上の例で $G[i+1][j] = 0$ かつ $G[i+1][j+1] = 0$ のときは、数字に上記の操作を 2 回することになり、結果の列はそれぞれ $2, 1, 2, 2$ となります。

こうなると、例えば操作を $2 \times x$ 回する場合の数字の変換は、古い x 回の操作による数字の変換と新しい x 回の操作による数字の変換を合成することで求めることができます。

このようにダブリングの要領を用いることで $O(logW)$ で各 i, j についての $l[i][j][1]$ の値を求めることができます。(実際にはセグ木に操作を載せるのが一番楽かもしれません。)