Q 2.1)

Subproblems: for each $1 \leq i \leq m$ and $1 \leq j \leq n$, Let $P(i,j)$ be the problem of determining $opt(i,j)$, the maximum number of possible unique paths to reach $B[i][j]$ from $B[1][1]$ with only two kinds of moves (down one cell or right one cell).

Recurrence: for $i > 0$ and $1 \leq j \leq n$,

$$opt(i,j) = \begin{cases} 0, & if\ B[i][j] = TRUE \\ opt(i-1,j) + opt(i,j-1), & otherwise. \end{cases}$$

At any point that $opt(i,j) = 0$ interpreting no possible paths through that cell. There are only two possible ways to reach any cell which is either from its top cell or its left cell since only moving down one cell or right one cell are allowed. So, the number of possible unique paths to reach any cell is the sum of possible unique paths of its top cell and its left cell.

Since $opt(i,j)$ depends on $opt(i-1,j)$ (its top cell) and $opt(i,j-1)$ (its left cell), we solve subproblems in increasing order of j then i.

Base cases:

If $i = 0$ or $j = 0$ then $opt(i,j) = 0$
Interpreting no possible paths to outside the whole cells.

$$opt(1,1) = \begin{cases} 0, & if\ B[1][1] = TRUE \\ 1, & otherwise. \end{cases}$$

Interpreting only one possible path to the start which is staying at the start unless there is a box.

If $opt(m,n) > 0$, it means the warehouse layout meets the requirement for having that many paths to reach the exit $B[m][n]$ with only two kinds of moves. The warehouse layout doesn't meet the requirement otherwise.

Overall time complexity is $O(mn)$ for iterating through all elements in a 2D array of size $m * n$ (all elements from one column then move to another column). And each of $O(mn)$ subproblems is solved in constant time.

Q 2.2)

Subproblems: for each $1 \le i \le m$ and $1 \le j \le n$, Let $P(i,j)$ be the problem of determining $opt(i,j)$, the minimum number of boxes that must be removed in order to reach $B[i][j]$ from $B[1][1]$ with only two kinds of moves (down one cell or right one cell).

Recurrence: for $i > 0$ and $1 \le j \le n$,

$$opt(i,j) = \min\big(opt(i-1,j), opt(i,j-1)\big) + \begin{cases} 1, & if\ B[i][j] = TRUE \\ 0, & otherwise. \end{cases}$$

At any point that $opt(i,j) = 0$ interpreting no boxes need to be removed to make a path to that cell.
There are only two possible ways to reach any cell which is either from its top cell or its left cell since only moving down one cell or right one cell are allowed.

The number of boxes that must be removed in order to reach any cell is either from its top or left cell then increase it by 1 if the cell has a box on it (since that box must be removed in order to reach the cell).
So, the smallest number could be found by choosing the minimum number of boxes that must be removed in order to reach its top and its left cell (among two of them).

Since $opt(i,j)$ depends on $opt(i-1,j)$ (its top cell) and $opt(i,j-1)$ (its left cell), we solve subproblems in increasing order of j then i.

Base cases:

If $i = 0$ or $j = 0$ then $opt(i,j) = \infty$
Interpreting no possible paths to outside the whole cells no matters how many boxes are removed.

$$opt(1,1) = \begin{cases} 1, & if\ B[1][1] = TRUE \\ 0, & otherwise. \end{cases}.$$

Interpreting a box must be removed if it happens to be at the starting cell.

$opt(m,n)$ is the smallest number of boxes that must be removed to meet the requirement to reach the exit $B[m][n]$.

Overall time complexity is $O(mn)$ for iterating through all elements in a 2D array of size $m * n$ (all elements from one column then move to another column). And each of $O(mn)$ subproblems is solved in constant time.

Q 2.3)

Since no matter what unique paths we take from $B[i][j]$ to $B[m][n]$, we must move down one cell ($B[i][j]$ to $B[i][j+1]$) then move to the right one cell ($B[i][j]$ to $B[i+1][j]$) at least once (not in a strict order) because $m, n \geq i, j$. Creating a corner (a cell connecting $B[i][j]$ and $B[i+1][j+1]$) in a path.

A shortcut is when we move down one cell and to the right one cell at the same time ($B[i][j]$ to $B[i+1][j+1]$). That means whenever we have a corner, we would be able to use a shortcut instead because we would end up in the same destination location (and as mentioned above that there's always at least one corner in a path).

So, if there exists a path to the exit, there would have at least one cell that could be using a shortcut.

Q 2.4)

Subproblems: for each $1 \leq i \leq m$, $1 \leq j \leq n$ and $k = \{0,1\}$, let $P(i,j,k)$ be the problem of determining $opt(i,j,k)$, the minimum sum of hazard rating of a path from $B[1][1]$ to $B[i][j]$ taking exactly k shortcuts with only two kinds of moves.

Recurrence: for $i > 0$, $1 \leq j \leq n$ and $k = 0,1$,

$$opt(i,j,0) = \begin{cases} \infty, & if\ B[i][j] = TRUE \\ \min\big(opt(i-1,j,0), opt(i,j-1,0)\big) + \mathrm{H[i][j]}, & otherwise. \end{cases}$$

There are only two possible ways to reach any cell which is either from its top cell or its left cell since only moving down one cell or right one cell are allowed. So, taking the path with least minimum sum of hazard rating between those two possible paths would make the cell becoming the minimum one.

$$opt(i,j,1) = \begin{cases} \infty, & if\ B[i][j] = TRUE \\ \min\begin{pmatrix} \mathrm{opt(i-1,j-1,0)}, \\ opt(i-1,j,1), \\ opt(i,j-1,1) \end{pmatrix} + H[i][j], & otherwise. \end{cases}$$

There are only three possible ways to reach a cell if a shortcut could be taken, via its top-left cell, top cell and left cell. If $opt(i-1,j-1,0) = \infty$, meaning either the path is impossible (there's a box), or a necessary shortcut had been taken (when it is impossible for the destiny cell to be reached from its top or left cell, unless taking a shortcut from its top-left cell), then no more shortcuts could be taken. This is to make sure that no more than one shortcut could be taken in a valid path.

Then the path with minimum sum of hazard rating among three possible ways to reach a cell is taken. If $\mathrm{opt(i-1,j-1,0)}$ is chosen, meaning taking a new shortcut from its top-left cell make the sum of hazard rating be less than taking a shortcut somewhere else along the path. The other two options ($opt(i-1,j,1), opt(i,j-1,1)$) interpreting the (minimum sum) paths from its top and left cells that had already taken a shortcut along the paths.

Since $opt(i,j,0)$ depends on $opt(i-1,j,0)$ (its top cell) and $opt(i,j-1,0)$ (its left cell). Same goes for $opt(i,j,1)$ but it also depends $opt(i-1,j-1,0)$. So, we solve subproblems in increasing order of j then i then k.

Base cases: if $i=0$ or $j=0$ then $opt(i,j,k)=\infty$. $opt(1,1,0)=H[i][j]$.
Interpreting no possible paths from outside the grid and the hazard rating at the starting cell is the only option.

If $i=1$ of $j=1$ then $opt(i,j,1)=\infty$.
Interpreting shortcuts couldn't be taken since there's no corners appear from only moving right one cell or only moving left one cell.

Whenever there's a box in a cell, no paths are possible to reach that cell, so opt are ∞.

$opt(m,n,1)$ is the minimum sum of hazard rating of a path from start to exit $B[m][n]$ taking exactly one shortcut with only two kinds of moves. There would always be at least one valid path using a shortcut as explained in Q2.3.

Overall time complexity is $O(mn)$ for iterating through each element in a 2D array of size $m*n$ twice (all elements from one column then move to another column). And each of $O(mn)$ subproblems is solved in constant time. Costing $O(m*n*2)=O(mn)$ in total.