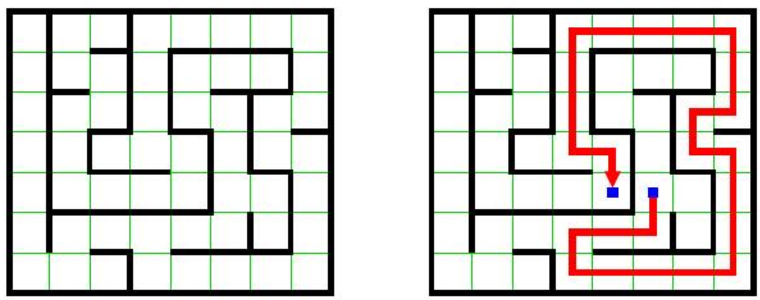
Graph Traversal

You are given a two-dimensional $m \times n$ grid of square cells. Two cells are adjacent if they share a common horizontal or vertical edge (not just a corner). A wall exists between every adjacent pair of cells. You generate a bhulbhulaiya out of the grid by randomly removing some of the walls. Then, you are given two distinct cells in the bhulbhulaiya. Your task is to find out a path to join the given cells. Your bhulbhulaiya should be such that between every pair of (distinct) cells, there exists a unique path, that is, the path-finding problem in the bhulbhulaiya is always uniquely solvable. The following figure gives an example of a 7×8 bhulbhulaiya and the path from the source cell $S = (4,5)$ to the destination (target) cell $T = (4,4)$.



Both these problems (bhulbhulaiya generation and path finding) can be solved using suitable graph-traversal algorithms. Consider an undirected graph $G = (V,E)$, where V is the set of mn cells of the grid. A vertex (i,j) is adjacent only to the four cells $(i \pm 1, j \pm 1)$ (a cell at an edge or at a corner has less than four neighbors). The edge set E of the graph contains $< 2mn$ cells (the exact size is $|E| = (m-1)n + m(n-1) = 2mn - m - n$, because the horizontal walls form an $(m-1) \times n$ array, whereas the vertical walls form an $m \times (n-1)$ array). A BFS/DFS traversal on G runs in $O(|V| + |E|) = O(mn)$ time. For solving the problems stated above, you do not need the graph G explicitly (that is, in the adjacency-matrix or in the adjacency-list format). Given any vertex $(i,j) \in V$, its neighbors (there are at most four of them) can be easily calculated.

Part 1: Initialize a bhulbhulaiya

A bhulbhulaiya consists of the following items.

- The row-dimension m.
- The column-dimension n.
- An $(m-1) \times n$ array H of horizontal walls, where $H[i][j]$ stores the information whether the wall between the cells (i,j) and $(i+1,j)$ is present in the bhulbhulaiya or removed from the bhulbhulaiya.
- An $m \times (n-1)$ array V with $V[i][j]$ storing the information (presence/absence) of the vertical wall between the cells (i,j) and $(i,j+1)$.
- An $m \times n$ array P of parent pointers, where $P[u][v]$ is meant to store the pair (i,j) of indices if (u,v) is a child of (i,j) in the DFS tree to be created in Part 3.

Define a suitable structure to store the above fields pertaining to a bhulbhulaiya. Given m,n, one can initialize a bhulbhulaiya by creating the three arrays H,V,P. To start with, all walls are kept, and all parents are stored as an invalid index like $(-1,-1)$. Write a function *initbhul*(m,n) to create and return an initialized grid.

Part 2: Print a bhulbhulaiya (and a path in it)

Write a function *prnbhul*(M) to print a bhulbhulaiya in the format illustrated in the sample output. An existent wall is shown as a horizontal/vertical line, and a removed wall as blank. In Parts 1 and 3, each cell is shown as empty. In Part 4, you need to show a path between the source and target cells S and T. So this print function would take an "optional" second argument which specifies a path in the bhulbhulaiya.

Part 3: Generate a random bhulbhulaiya

Make a random DFS traversal in G in order to create a uniquely connected bhulbhulaiya from an initialized $m \times n$ grid. Notice that G is never explicitly supplied to you. It will remain only in your head. You need it only for finding the neighbors of a cell.

Write a recursive *DFS*( ) function to traverse G in the depth-first fashion. If you are at vertex (i, j), locate its unvisited neighbors in a random sequence. For each unvisited neighbor (u, v) of (i, j), a recursive call is made. You should remove the wall between (i, j) and (u, v) before this recursive call. Moreover, (u, v) becomes a child of (i, j) in the DFS tree, so set the parent pointer $P[u][v]$ to the pair (i, j).

Write a function $genbhul(B)$ to create a random bhulbhulaiya in B (assumed initialized). This function chooses a random cell $R = (r, s)$ to start the DFS traversal. It makes the outermost call of the recursive DFS function on this vertex R (this becomes the root of the DFS tree).

Notice that for a proper running of the DFS traversal, you need to appropriately manage a *visited* array which in this case is two-dimensional of size $m \times n$.

Part 4: Find paths in the bhulbhulaiya

Assume that you have generated a uniquely connected $m \times n$ bhulbhulaiya using the DFS traversal of Part 3. Randomly choose two different cells $S = (u, v)$ (the king starts at the source) and $T = (x, y)$ (the queen hides at the target). There is a unique path in the bhulbhulaiya, that connects S and T. This path can be obtained by running a second traversal (DFS/BFS) from S on the DFS tree consisting of the removed walls. You instead follow a different approach.

Following the parent pointers stored in Part 3, generate the unique path from S to the root R of the DFS tree. Likewise, obtain the unique path from T to R. Do some cut and paste on these two paths in order to obtain the desired S-T path in the bhulbhulaiya. Write a function *findrani* to implement this idea.

The *main*() function

- Read m and n from the user.
- Initialize (Part 1) and print (Part 2) an $m \times n$ bhulbhulaiya B (showing all the walls).
- Call *genbhul* on B (Part 3) to generate a random uniquely connected bhulbhulaiya in B. Print the bhulbhulaiya (with some of the walls removed).
- Generate two distinct cells S and T. Find the unique S-T path in B (Part 4). Print the bhulbhulaiya along with the discovered path.

Note: By adding the line

```
srand((unsigned int)time(NULL));
```

at the beginning of your *main* function, you can generate different bhulbhulaiyas in different runs of your program.

Sample output

```
m = 10
n = 20

+++ Initial bhulbhulaiya
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

+++ Random bhulbhulaiya generated
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    |                   |                           |                               |
    +   +---+---+---+   +   +---+---+---+---+---+   +   +---+---+---+---+---+---+   +
    |   |           |       |       |       |       |   |   |                   |   |
    +   +---+   +   +---+---+---+   +   +   +   +---+   +   +   +---+---+   +   +   +
    |       |   |               |       |   |           |   |       |       |   |   |
    +---+   +   +---+---+---+   +---+---+   +---+---+---+   +---+   +   +---+---+   +
    |       |   |       |               |                       |   |               |
    +   +---+   +   +   +---+---+---+   +   +---+---+---+---+   +   +---+---+---+   +
    |   |       |   |           |   |       |   |               |   |           |   |
    +   +   +   +   +---+---+   +   +---+---+   +   +---+---+---+   +---+---+   +   +
    |   |   |           |       |   |           |       |           |   |       |   |
    +   +   +---+---+---+   +---+   +   +   +---+   +---+   +---+---+   +   +---+   +
    |   |       |       |   |           |           |                   |   |       |
    +   +   +---+   +   +   +---+---+---+---+---+---+   +---+---+---+   +   +---+   +
    |   |   |       |   |       |       |   |               |       |   |       |   |
    +   +   +   +---+   +---+   +   +   +---+   +---+---+---+   +   +---+---+   +   +
    |   |   |   |   |       |       |               |       |   |               |   |
    +   +---+   +   +---+   +---+---+---+---+---+---+   +   +   +---+---+---+---+   +
    |                   |                               |       |                   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

+++ Path from S = (1,9) to T = (9,14)
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    | x   x   x   x   x | x   x   x   x   x   x   x | x   x   x   x   x   x   x   x |
    +   +---+---+---+   +   +---+---+---+---+---+   +   +---+---+---+---+---+---+   +
    | x |     x   x | x   x |       |     S | x   x | x |   | x   x   x   x     | x |
    +   +---+   +   +---+---+---+   +   +   +   +---+   +   +   +---+---+   +   +   +
    | x   x | x | x   x   x   x |       | x | x   x   x |   | x   x | x   x |   | x |
    +---+   +   +---+---+---+   +---+---+   +---+---+---+   +---+   +   +---+---+   +
    | x   x | x | x   x |     x   x   x | x                    | x | x   x   x   x |
    +   +---+   +   +   +---+---+---+   +   +---+---+---+---+   +   +---+---+---+   +
    | x |     x | x | x   x   x |   | x   x |   |               | x |           |   |
    +   +   +   +   +---+---+   +   +---+---+   +   +---+---+---+   +---+---+   +   +
    | x |   | x   x     | x   x |   |           |       | x   x   x |   |       |   |
    +   +   +---+---+---+   +---+   +   +   +---+   +---+   +---+---+   +   +---+   +
    | x |       | x   x | x |           |           | x   x             |   |       |
    +   +   +---+   +   +   +---+---+---+---+---+---+   +---+---+---+   +   +---+   +
    | x |   | x   x | x | x   x | x   x |   | x   x   x     |       |   |       |   |
    +   +   +   +---+   +---+   +   +   +---+   +---+---+---+   +   +---+---+   +   +
    | x |   | x |   | x   x | x   x | x   x   x     | x   x |   |               |   |
    +   +---+   +   +---+   +---+---+---+---+---+---+   +   +   +---+---+---+---+   +
    | x   x   x         | x   x   x   x   x   x   x   x | x   T |                   |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```