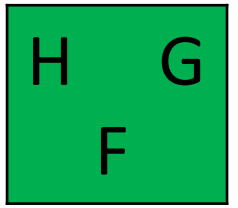
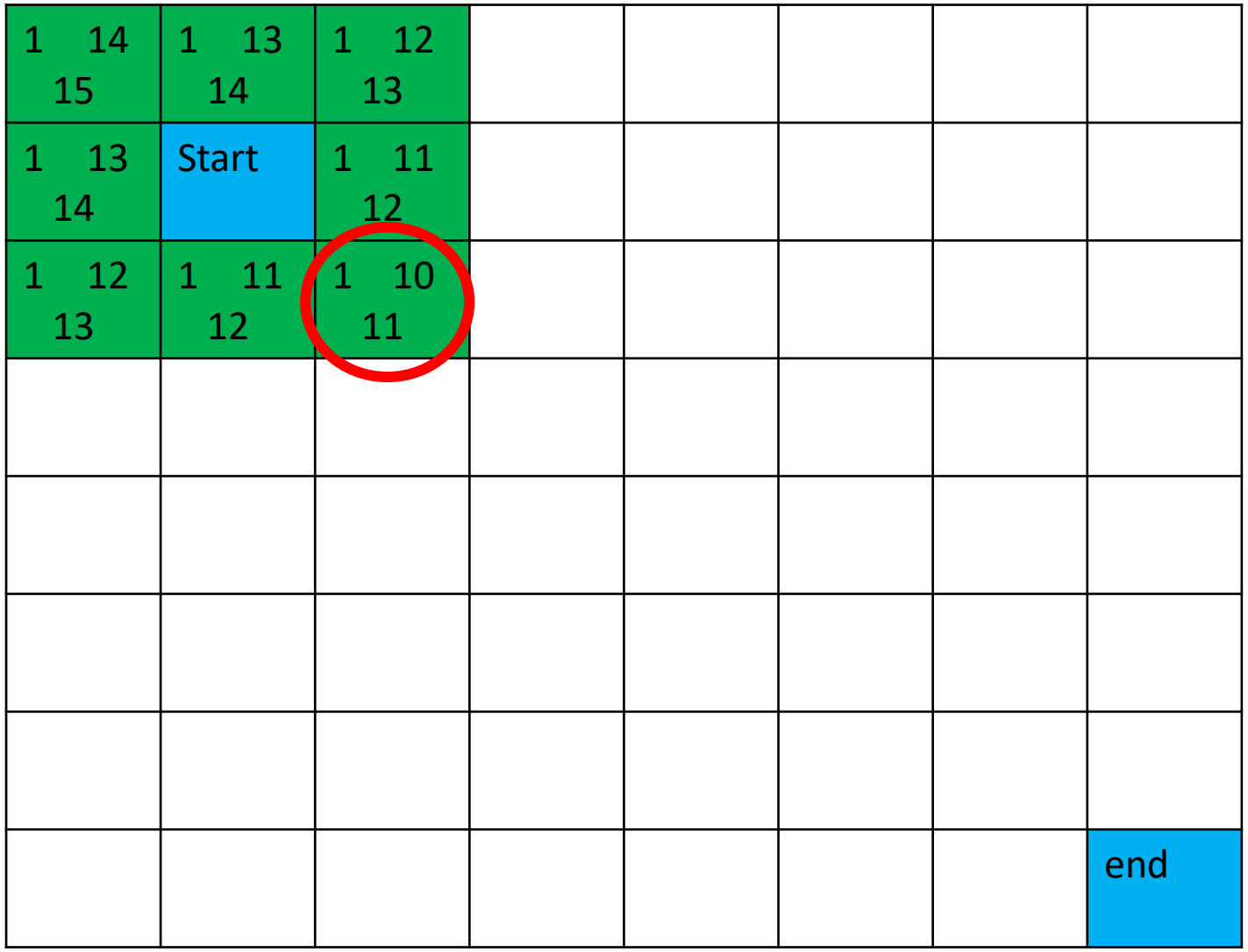
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 18<br>19 | 1 17<br>18 | 1 16<br>17 | | | | | |
| 1 17<br>18 | Start | 1 15<br>16 | | | | | |
| 1 16<br>17 | 1 15<br>16 | 1 14<br>15 | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | end |




G cost = distance from starting node
H cost (heuristic) = distance from end node
F cost = G cost + H cost

Manhattan Distance (One of the heuristic)
= abs(current_cell.x - goal.x) +
abs(current_cell.y - goal.y)




Open list: green cell
Close list: blue cell

- Find the lowest F cost cell in open list
- Choose it as the father node in next iteration
- So F cost 11 cell will be next father node
- Remove current cell from open list
- Add current cell to close list

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 14 15 | 1 13 14 | 1 12 13 | | | | | |
| 1 13 14 | 0 0 0 | 1 11 12 | 2 10 12 | | | | |
| 1 12 13 | 1 11 12 | 1 10 11 | 2 9 11 | | | | |
| | 2 10 12 | 2 9 11 | 2 8 10 | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | end |

Open list: green cell
Close list: blue cell

- Check whether the adjacent cell is over the maze limit
- Add adjacent cell from father node to open list if there are not exist in it and not over the limit
- Also, calculate those node cost

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 14 15 | 1 13 14 | 1 12 13 | | | | | |
| 1 13 14 | 0 0 0 | 1 11 12 | 2 10 12 | | | | |
| 1 12 13 | 1 11 12 | 1 10 11 | 2 9 11 | 3 8 11 | | | |
| | 2 10 12 | 2 9 11 | 2 8 10 | 3 7 10 | 4 6 10 | | |
| | | 3 8 11 | 3 7 10 | 3 6 9 | 4 5 9 | 5 4 9 | |
| | | | 4 6 10 | 4 5 9 | 4 4 8 | 5 3 8 | 6 2 8 |
| | | | | 5 4 9 | 5 3 8 | 5 2 7 | 6 1 7 |
| | | | | | 6 2 8 | 6 1 7 | end |

Open list: green cell
Close list: blue cell

- Repeat the process, until the adjacent cell include the end node
- Once found the end node, keep tracing back the parent node, so we can get the shortest path