```python
import heapq

def heuristic(a, b):
    return abs(a[0]-b[0]) + abs(a[1]-b[1])

def a_star(start, goal):
    grid_size = (4, 4)
    moves = [(-1,0), (1,0), (0,-1), (0,1)]
    open_set = [(heuristic(start, goal), 0, start, [start])]
    visited = set()

    while open_set:
        _, cost, current, path = heapq.heappop(open_set)
        if current == goal:
            return path
        if current in visited:
            continue
        visited.add(current)
        for dx, dy in moves:
            nx, ny = current[0] + dx, current[1] + dy
            neighbor = (nx, ny)
            if 0 <= nx < grid_size[0] and 0 <= ny < grid_size[1]:
                heapq.heappush(open_set, (cost+1+heuristic(neighbor, goal), cost+1, neighbor,
                    path + [neighbor]))

start = (0, 0)
goal = (3, 3)
path = a_star(start, goal)
print(f"start={start} and goal={goal}:\npath from start to goal:{path}")
```

```
start=(0, 0) and goal=(3, 3):
path from start to goal:[(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (2, 3), (3, 3)]

=== Code Execution Successful ===
```