Astar pathfinding with closed array instead of closed list, pseudocode

Add starting point to OPENLIST, give it an f value of anything (0 is fine);

CLOSEDARRAY stores the calculated g values: initialize it to unfeasibly high values, like 99999999

Set CLOSEDARRAY[startx][starty] g cost to 0;

While(!done)

If OPENLIST is empty, we failed to find a path. Return failure.

Pull top node off OPENLIST, make it CURRENTNODE

IS CURRENTNODE the goal x,y? Then we are done the while loop, have found a path.

Else Expand in all directions from CURRENTNODE: for each direction NEWDIR:

Can we move to this set of coords?

If so, calculate a new g cost by using CLOSEDARRAY[CURRENTNODE.x]

[ CURRENTNODE.y] g cost and adding cost of new direction step

Is the new cost LESS than using CLOSEDARRAY[NEWDIR.x][NEWDIR.y] g cost?

If so, update CLOSEDARRAY[NEWDIR.x][NEWDIR.y] g cost to the new cost

Make a new node for NEWDIR, calculate it’s fcost from CLOSEDARRAY[NEWDIR.x][NEWDIR.y] g cost + h cost (heuristic \* distance to goal from here)

Search the OPENLIST, if any nodes on OPENLIST have the same x,y cords as NEWDIR, delete them.

Add NEWDIR to OPENLIST.

When done expanding all directions, sort OPENLIST by fcost;

If we exited the while loop, we found our path. Now rebuild the path by starting at the goal and working backwards:

Check all direction we can step to from the goal, find the one with the cheapest g cost on CLOSEDARRAY: that is where we must have come from to get here.

Repeat the above until you have traced your path back to the start, making a list of the points you pass through. You now have the path in reverse order.