You are overestimating when the heuristic's estimate is higher than the actual final path cost. You're underestimating when it's lower (you don't have to underestimate, you just have to not overestimate; correct estimates are fine). If your graph's edge costs are all 1, then the examples you give would provide overestimates and underestimates, though the plain coordinate distance also works peachy in Cartesian space.

Assuming F(N) = distance from node N to a goal node, G(N) = known the smallest distance from the start to node N, H(N) = estimate of distance from N to the goal node.

If H(N) is an overestimate then if the algorithm reaches the goal node by some other path with distance < F(N) then it will never look at node N, because H(N) is then seemingly worse than an already established path.  And so even if the \_actual\_ distance via N is optimal this path will then be overlooked.  Which is incorrect.

If, on the other hand, H(N) is an underestimate then if the algorithm reaches the goal node by some path with distance D, and the actual distance via N is less than D, then it is guaranteed that F(N) will also, be less than D, so the algorithm will continue to look at N (and perhaps find a path shorter than D).

Overestimating doesn't exactly make the algorithm "incorrect"; what it means is that you no longer have an admissible heuristic, which is a condition for A\* to be guaranteed to produce optimal behavior. With an inadmissible heuristic, the algorithm can wind up doing tons of superfluous work examining paths that it should be ignoring, and possibly finding suboptimal paths because of exploring those. Whether that actually occurs depends on your problem space. It happens because the path cost is 'out of joint' with the estimated cost, which essentially gives the algorithm messed up ideas about which paths are better than others.