# A* algorithm pseudocode

```
closedset := the empty set          // The set of nodes already evaluated
openset := {start}                  // The set of tentative nodes to be evaluated
parent := the empty map             // The map of navigated nodes
g[start] := 0                       // Cost from start along best known path
f[start] := g[start] + h(start)     // Estimated total cost from start to nearest goal through y
while openset is not empty
        current := the node in openset having the lowest f[] value
        if is_goal(current)
                return reconstruct_path(parent, current)
        remove current from openset
        add current to closedset
        for each neighbor in neighbor_nodes(current)
                if neighbor in closedset continue
                tentative_g := g[current] + dist_between(current,neighbor)
                if neighbor not in openset or tentative_g < g[neighbor]
                                parent[neighbor] := current
                                g[neighbor] := tentative_g
                                f[neighbor] := g[neighbor] + h(neighbor)
                                if neighbor not in openset
                                                add neighbor to openset
return failure
```

# Genetic algorithm pseudocode

```
// population is an ordered list of individuals
// weights is a list of corresponding fitness values for each individual
// fitness is a function to compute these values

function GENETIC-ALGORITHM(population,fitness) returns an individual
        repeat
                weights := WEIGHTED-BY(population,fitness)
                population2 := empty list
                for i=1 to SIZE(population) do
                                parent1, parent2 := WEIGHTED-RANDOM-CHOICES(population,weights,2)
                                child := REPRODUCE(parent1, parent2)
                                if (small random probability) then child := MUTATE(child)
                                add child to population2
                 population := population2
        until some individual is fit enough, or enough time has elapsed
        return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
        n := LENGTH(parent1)
        c := random number from 1 to n
        return APPEND(SUBSTRING(parent1,1,c), SUBSTRING(parent2,c+1,n))
```