A* algorithm

Friday, November 13, 2020 11:06 AM

dicts
G(n) = { }
F(n) = { }

E = [ ] edges (nodes reached)
Open = [n_s]
Closed = [ ] (Visited nodes having some path through it)

$f(n_s) = g(n_s) + h(n_s)$
$\downarrow 0 \quad + h(n_s) = h(n_s)$

$G[n_s] = g(n_s) = 0$
$F[n_s] = h(n_s)$

if open is not empty.

Step ① Select node n in Open with smallest f(n) → call it n.
if $n = n_f$; break
(*) remove n from Open and place it in closed.

Step ② : For each adjacent/reachable node of n ⇒ Call it $\{n_1, n_2, n_3 \ldots\}$
list of adjacent nodes.

for n_i in $(n_1, n_2 \ldots n_3)$:
if $n_i == n_F$:
update G[n_i], F[n_i], P[n_i]
break.

$g(n_i)$ = G[n] + cost of going from n to n_i

if $g(n_i)$ (new cost) $< G[n_i]$ (old cost) ← if n_i not in G then $G[n_i] = \infty$

$G[n_i] = g(n_i)$
$P[n_i] = n$ { set parent of n_i }
$F[n_i] = g(n_i) + h(n_i)$

if n_i not in open and not in closed
add n_i to open

if n_i is in closed :
remove n_i from closed and add it to open

if n_i is in open leave it there and we can visit it later