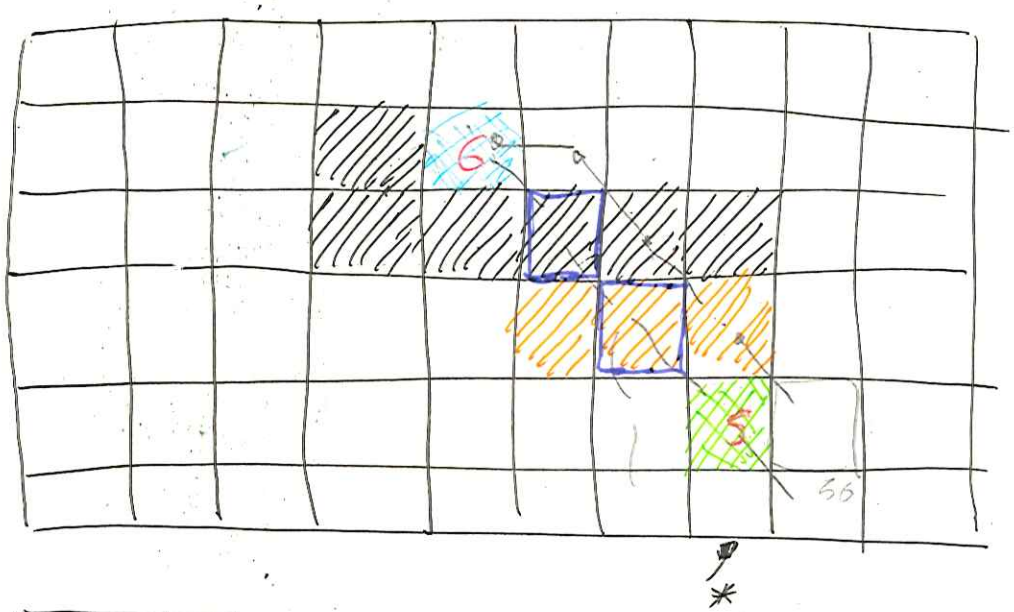
Entendiendo A*

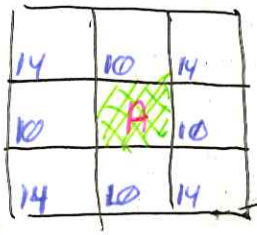
lo primero son las distancias Entre cada □ es 1, entonces en la diagonal es

$h = \sqrt{1^2 + 1^2}$

$h = \sqrt{2} = 1.41$

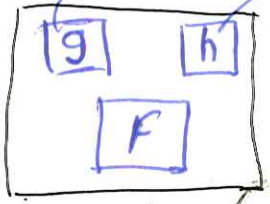
Se multiplica todo por 10 para que sea mas simple hacer los calculos.





Asi quedarian el cálculo para caminar hasta cada uno de los vecinos que tiene la "celda" ("tile" en ingles).

↳ Costo de caminar desde A.



g cost

h cost (heuristic)

Para una celda genera Cji

g: Que tan lejos esta de el punto inicio

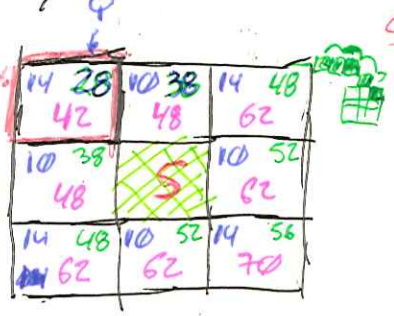
h: Distancia a el punto final

El algoritmo comienza desde la casilla inicial y calcula algunos valores relacionados a sus vecinos

↳ que tan lejos esta del pto G.

$F_{cost} = G_{cost} + H_{cost}$

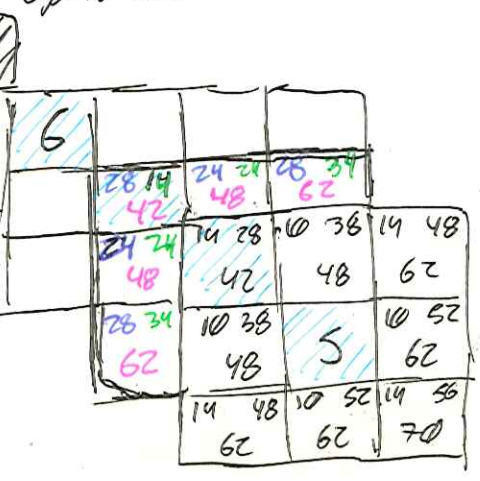
S: start → Posición de inicio.



El siguiente paso del algoritmo es mirar cada uno de los F_{cost} y seleccionar el mas pequeño.

Se repite la operación

En este caso se observa que el Hcost esta decreciendo en la diagonal 5 → 42 → 42-6



Este analisis hace referencia a la posición inicial mostrada en *

Q: Esta distancia es 28 porque son 2 diagonales para llegar a G desde ese cuadro

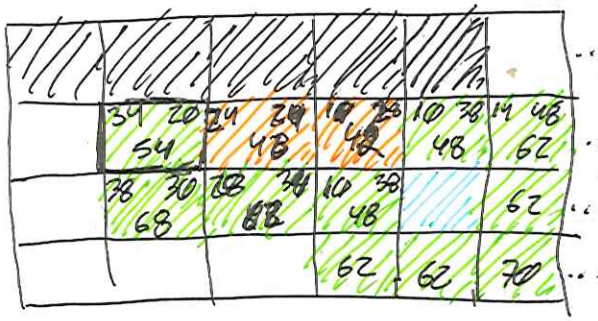
Continuando con A*

Ahora, en el primer ejemplo los obstaculos no fueron tenidos en cuenta, se hizo el ejemplo como una línea diagonal directa.

Observemos ahora que sucede cuando agregamos los obstaculos.

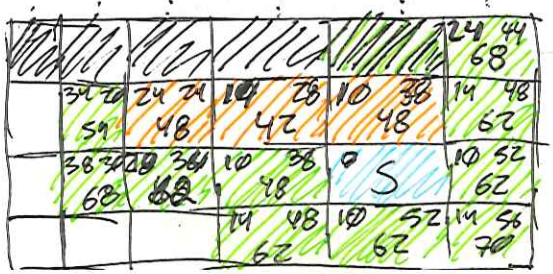
En este segundo paso se genera un escenario donde tenemos tres nodos con el mismo valor de f (48) decidir como ~~actuar~~ por lo cual se debe analizar los 3 nodos ~~restantes~~.

El criterio para decidir con cual nodo continuar es el valor de Hcost (que indica que tan cerca se está del objetivo). Por lo cual se analiza el nodo [24 24 48] de la izquierda del nodo actual.



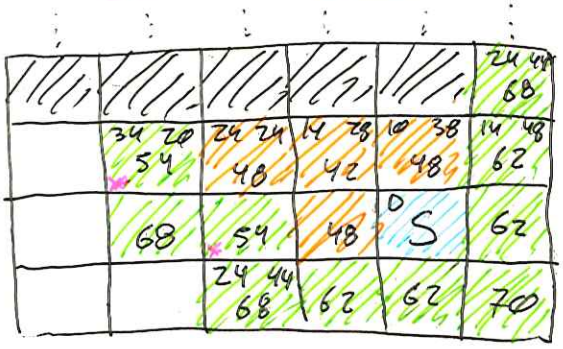
En este paso Observamos que el Hcost de la casilla siguiente aumentó a 54, lo cual nos indica que de alguna forma no nos acercamos al nodo G.

Nos quedan otros dos nodos ahora por analizar que estan con una Hcost de 48, como sus parametros son iguales se elige uno de los dos — el de la derecha



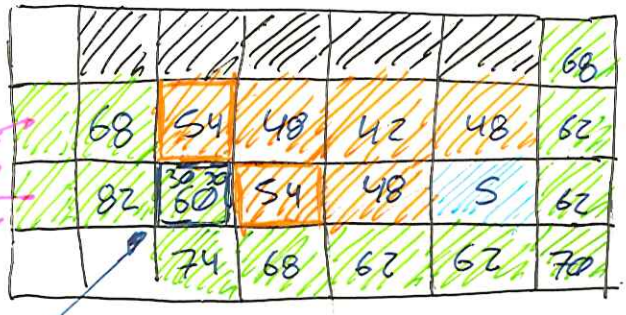
Y se debe evaluar el tercer elemento con Hcost en 48

Entendiendo A*



En este momento los elementos mas interesantes son los dos nodos con f_{cost} de 54

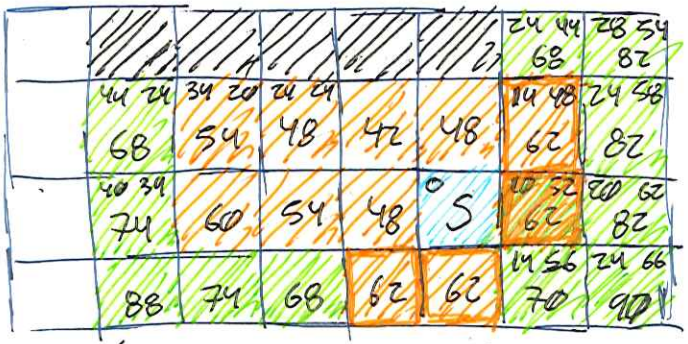
Es necesario analizar ambos: y verificar como se afectan los nodos vecinos:



Al analizar se actualizan los nodos y esto hace que el nuevo que tiene una funcion de costo menor es $f_{cost} = 60$

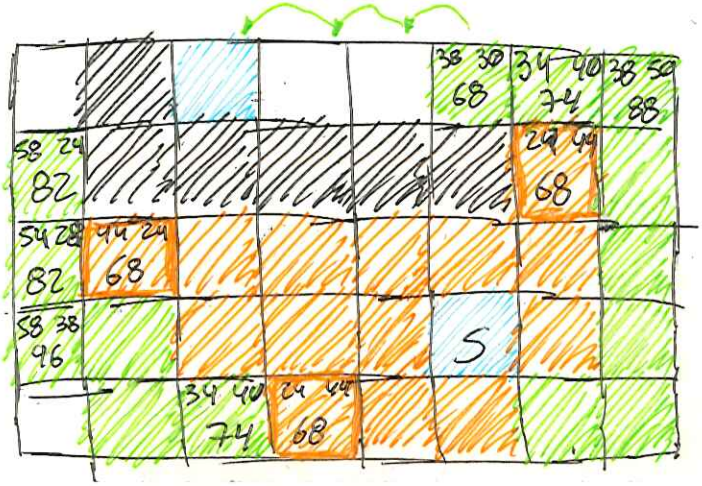
Observar que este fue actualizado debido a que hay un camino mas rápido, entonces su f_{cost} bajo'.

Analizar este elemento solo "abre" un nuevo lugar con un $f_{cost} = 88$, haciendo que los nuevos nodos a explorar sean los de $f_{cost} = 62$. (4 nodos)



Despues de actualizar, los nodos que pasan a ser los de interes son aquellos con el $f_{cost} = 68$. (3)

Continuamos con este analisis



Desde aca el sistema ya encuentra el camino, que es basicamente ir andando en linea recta.

Entendiendo A*

Seudo código del Algoritmo:

OPEN // Conjunto de nodos a ser evaluados.

CLOSED // Conjunto de nodos ya evaluados.

Agregar el nodo START al conjunto OPEN.

loop:

current = nodo en OPEN con el menor Fcost.

remover current del conjunto OPEN

agregar current al conjunto CLOSED.

if (current es el nodo GOAL) // El camino ha sido encontrado.

retornar

para cada neighbour de current node

if (neighbour no es atravesable (obstaculo) || neighbour esta en CLOSED)

brincar al siguiente neighbour

if (nuevo camino hacia neighbour es mas corto || neighbour no esta en OPEN)

actualizar Fcost de neighbour

actualizar parent de neighbour a current.

if (neighbour no esta en OPEN)

agregar neighbour en OPEN