ALGORTIMO A*

A* es un algoritmo de búsqueda inteligente o informada que busca el camino más corto desde un estado inicial al estado meta a través de un espacio de problema, usando una heurística óptima. Como ignora los pasos más cortos (más "chatos") en algunos casos rinde una solución subóptima.

Un algoritmo de búsqueda admisible es el que garantice el hallazgo de una ruta óptima entre el nodo de inicio y el nodo meta, si es que ella existe. En la búsqueda A* una heurística admisible es una que no sobreestima la distancia remanente entre el nodo presente y el nodo meta. Por ejemplo, siempre una ruta real entre dos ciudades es algo mayor y a lo sumo igual a la distancia en línea recta tomada de un mapa. Esta última distancia es así una heurística admisible pues en todo caso es "optimista", lo cual coincide con la definición.

Una heurística admisible.

La restricción consiste en escoger en una función h que nunca sobrestima el costo que implica alcanzar la meta. A tal h se le conoce como heurística admisible. Por naturaleza, este tipo de heurística es optimista pues consideran que el costo para resolver un problema siempre es inferior a lo que en realidad es. Este optimismo se transfiere a la función f. Si h es aceptable, f(n) nunca sobrestimara el costo real de la mejor solución que pase por n. en la búsqueda preferente por lo mejor, en la que se utiliza f como la función de evaluación y una función h aceptable se le conoce como búsqueda A*.

El comportamiento de una búsqueda A*

Antes de demostrar la integridad y condición optima de A*, es conveniente presentar una descripción intuitiva de cómo funciona. Aunque la descripción no remplaza a la demostración, si es más fácil recordarla y puede servir para producir la prueba cuando así se requiera. Para empezar, una observación preliminar: al observar los árboles de búsqueda de la Fig. 1.1, se nota un fenómeno interesante: a lo largo de las rutas originales en la raíz el costo f nunca disminuye. No es accidental, y se cumple en casi todas las heurísticas aceptables. **En toda heurística donde esto ocurre, se dice que muestran monotonicidad.**

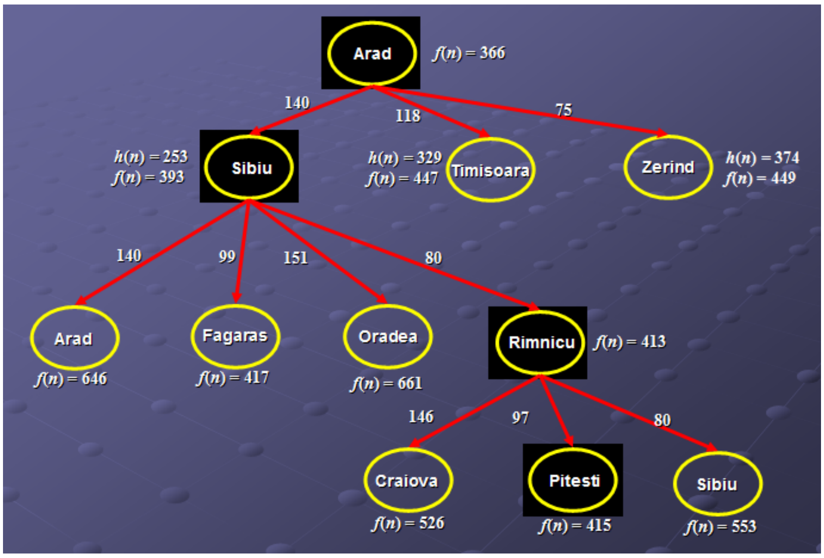


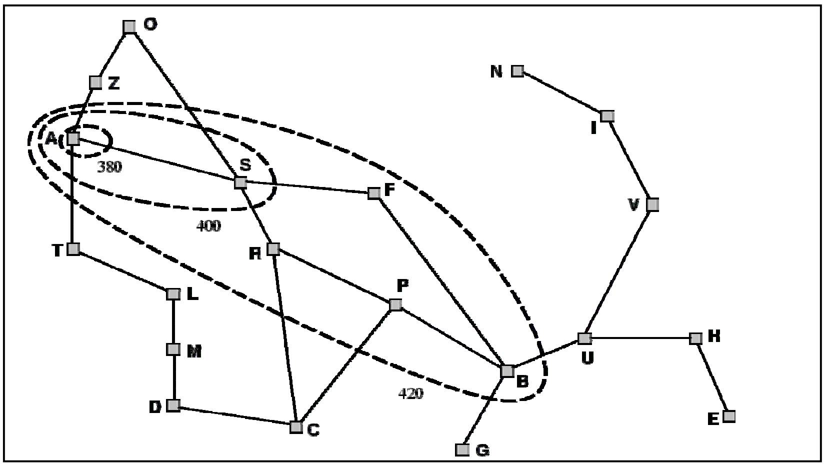
Figura 1.1

Los valores erróneos que puede producir una heurística se pueden omitir con la siguiente función:

f(n') = maxf(n), g(n') + h(n')

A esta ecuación se le denomina **Ruta Máxima**. El utilizarla garantiza que f nunca decrecerá a lo largo de una trayectoria originada en la raíz, siempre y cuando h sea aceptable.

El objetivo de la observación anterior es legitimar la descripción de lo que hace A*. Si f nunca disminuye a través de una ruta que parte de la raíz, conceptualmente es posible dibujar contornos en el espacio de estados. En la figura 1.2 se muestra un ejemplo. Dentro del contorno identificado como 400,f(n) en todos los nodos es igual o menor que 400, y así sucesivamente. Entonces, puesto que A* expande el nodo hoja que tiene la f mas baja, se concluye que una búsqueda A* se extiende desde el nodo de partida, y va añadiendo nodos en banda concéntricas cuyo valor f también va aumentado.



Fundamentos del algoritmo A*

A* es un algoritmo informado que basa su comportamiento en la evaluación de una función
expresada del siguiente modo:

f(n) = g(n) + h(n)

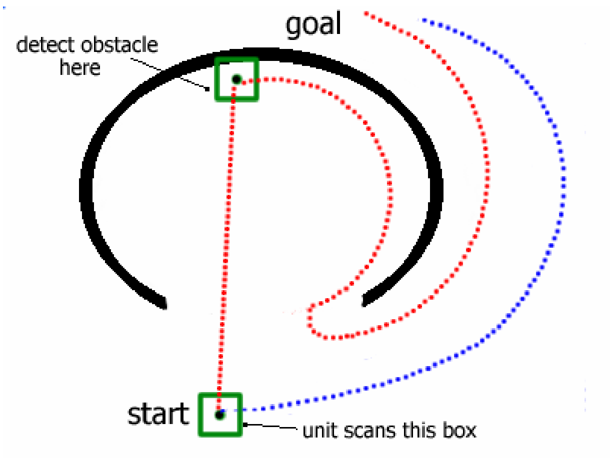
La función se encuentra compuesta por:

g(n) : es el costo de las movidas realizadas.
h(n) : es la función heurística. Representa el costo estimado del mejor camino.

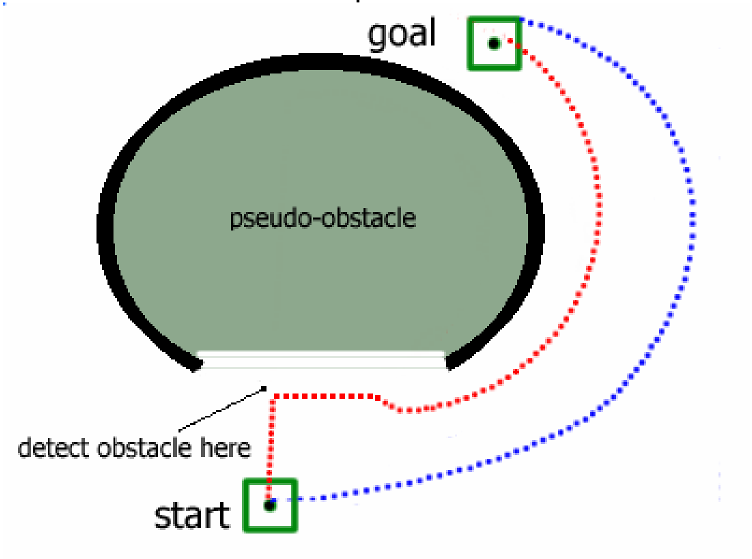
Dicha estimación debe ser realizada por defecto, es decir, siempre menor o igual a la real. En búsqueda de caminos, la función heurística suele ser el camino recto hacia la meta, ya que no importa como sea el mapa, es imposible que exista camino de costo menor.

El modo de realizar el cálculo de la distancia necesaria para llegar a la meta depende del tipo de movidas permitidas. Si solo podemos movernos vertical y horizontalmente podremos realizar el cálculo de la distancia Manhattan, que consiste en sumar la cantidad de bloques en horizontal y vertical que restan para llegar a la meta. Si además se permiten movidas diagonales, deberemos aplicar Pitágoras y el cálculo será la raíz cuadrada de la suma de los cuadrados de los catetos.

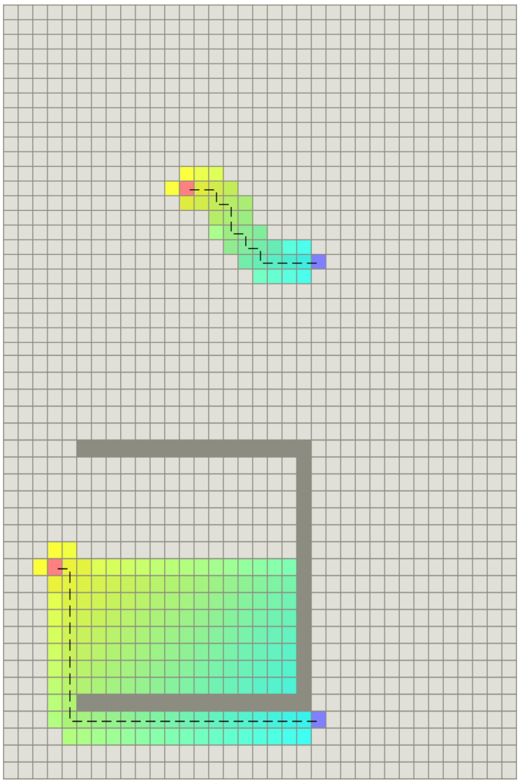
El movimiento de un punto a otro en simple pero la búsqueda de rutas optimas en mas compleja.



En la figura mostrada el algoritmo trata de llegar del punto de inicio (recuadro verde Start) en la parte inferior hasta la parte superior, no hay nada en el área que muestre que no debería moverse directamente hacia arriba, cerca de llegar a la parte de arriba se encuentra con un obstáculo y cambia su ruta dándole la vuelta al obstáculo en forma de "U" siguiendo la trayectoria roja, de haber empleado un buscador de ruta como el A* este seguiría la ruta mas optima a la meta que en esta caso se describe por la ruta azul.



Se puede extender el algoritmo para detectar movimientos avanzados, por ejemplo en esta figura se detecta una “trampa” para evitar que el algoritmo entre a la forma cóncava, detecta que no hay salida y no entra a menos que la meta estuviera dentro (la parte sombreada “pseudos-obstacle” es hueca no es un obstáculo el obstáculo en el contorno negro en forma de “U”) en otras palabras el algoritmo empleado en esta figura es mas avanzado mas extendido que el anterior.



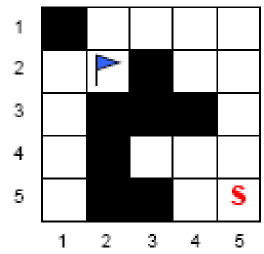
En las graficas anteriores, los recuadros claros muestran las áreas mas lejanas al punto meta, mientras que los cuadros obscuros muestran las áreas mas lejanas al punto de inicio, el algoritmo balancea los tonos para llegar al punto meta. $f(n) = g(n) + h(n)$.

PSEUDOCODIGO DE A*

```
Initialize OPEN list
Initialize CLOSED list
Create goal node; call it node_goal
Create start node; call it node_start
Add node_start to the OPEN list
while the OPEN list is not empty
{
        Get node n off the OPEN list with the lowest f(n)
        Add n to the CLOSED list
         if n is the same as node_goal we have found the solution; return Solution(n)
         Generate each successor node n' of n
         for each successor node n' of n
         {
           Set the parent of n' to n
           Set h(n') to be the heuristically estimate distance to node_goal
           Set g(n') to be g(n) plus the cost to get to n' from n
           Set f(n') to be g(n') plus h(n')
           if n' is on the OPEN list and the existing one is as good or better then discard n'
and continue
           if n' is on the CLOSED list and the existing one is as good or better then discard
n' and continue
           Remove occurrences of n' from OPEN and CLOSED
           Add n' to the OPEN list
         }
}
 return failure (if we reach this point weve searched all reachable nodes and still havent
found the solution, therefore one doesnt exist)
```

EJEMPLO DE A*

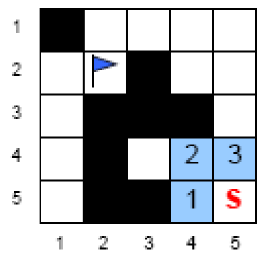
Analicemos un ejemplo para ver el algoritmo en acción:

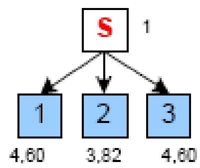


S representa el punto de partida. La bandera azul representa la meta. Los casilleros de color negro representan obstáculos y los casilleros blancos representan caminos posibles.
Nuestra función heurística será:

$$h(n) = \sqrt{(X_{(n)} - X_{(nf)})^2 + (Y_{(n)} - Y_{(nf)})^2}$$

En pocas palabras, lo que expresamos en la fórmula anterior fue la distancia que resta para llegar a la meta desde la posición en la cual nos encontramos.
¿Y cuáles serían las primeras movidas posibles?





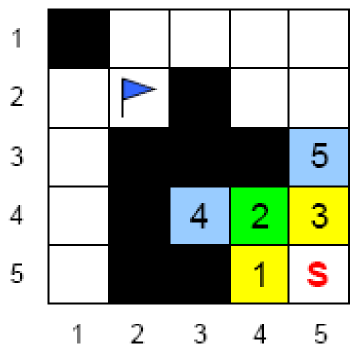
En la figura anterior vemos a izquierda como avanzamos por el mapa y derecha como queda conformado el árbol con el valor de la función **f(n)** para cada una de las bases abiertas. De todas ellas seleccionaremos la de menor **f**, es decir, la de menor costo calculado como la suma del costo del camino recorrido y el costo estimado del camino que resta recorrer.

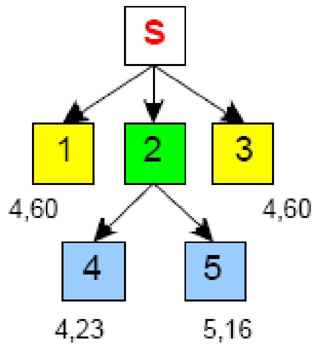
Para entender como realizamos el cálculo del valor de la función **f(n)**, veamos en mayor detalle como realizamos el cálculo de la primer movida:

$$f(1) = g(1) + h(1) = 1 + \sqrt{(4-2)^2 + (5-2)^2} = 1 + 3{,}60 = 4{,}60$$

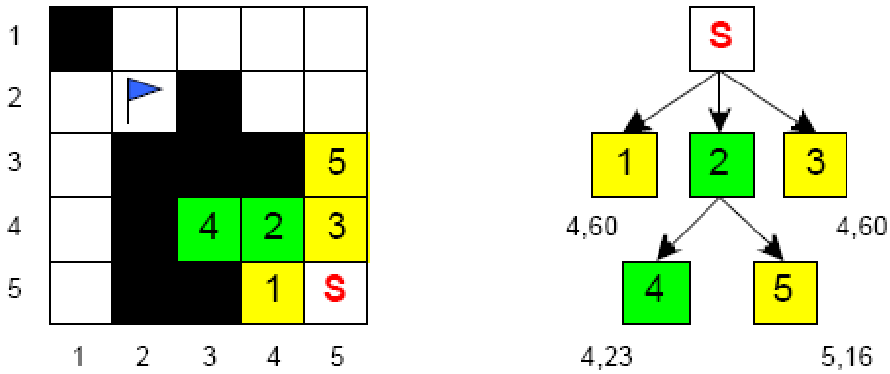
¡Es muy sencillo!, **g** siempre incrementará en uno a medida que avancemos por el mapa (ya que estamos tomando el costo de avance en forma unitaria para todas las posiciones) y **h** será el resultado de aplicar la fórmula de la figura 2, que es la distancia de dicha posición a la meta.

Una vez abiertas las bases del nivel, deberemos elegir cual tomar para seguir nuestro camino. Para esto, siempre deberemos seleccionar la de menor **f** (en caso de haber más de una con mismo valor, la elección es arbitraria). Siguiendo nuestro ejemplo, deberemos seleccionar la movida 2.

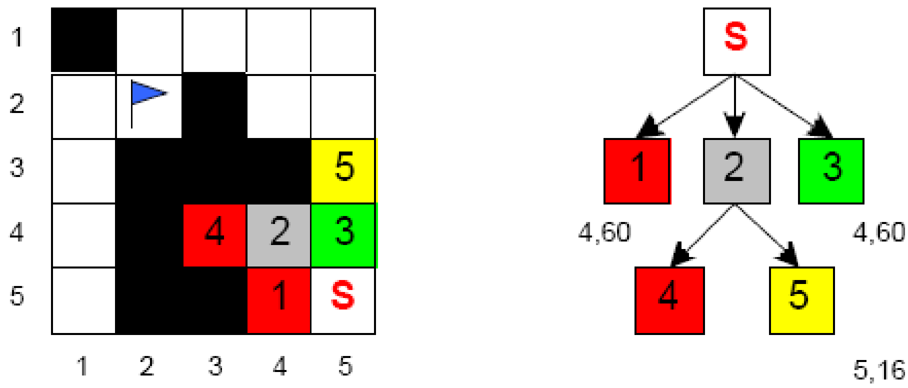




Evaluamos las movidas posibles a partir de la posición elegida y volvemos a seleccionar la de menor **f.** Es muy importante destacar que en dicha elección también intervienen las bases abiertas en las movidas anteriores (que en el árbol están pintadas de color amarillo). Por lo tanto, es posible (y muy factible) que en un determinado momento el algoritmo abandone el camino por el cuál se dirige para "probar suerte" por otro lado.

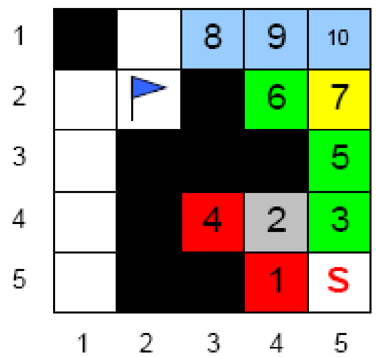


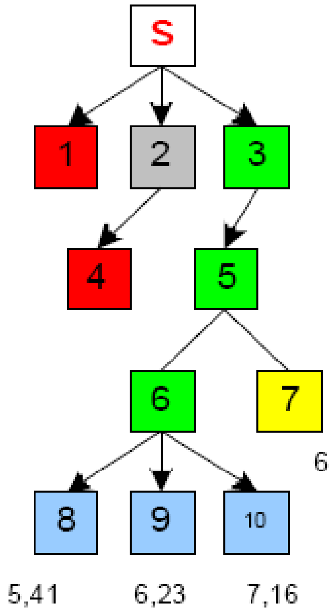
De la figura 6 es interesante notar como el algoritmo intenta elegir un camino que lo lleve en línea recta hacia la meta, ¡pero dicho camino está obstruido! Efectivamente, vamos camino a un callejón sin salida. Llegado el momento, el algoritmo tomará cuenta que a partir de una posición no es posible realizar más movidas y deberá volver sobre sus pasos para seleccionar una base abierta anteriormente en su búsqueda por la meta.



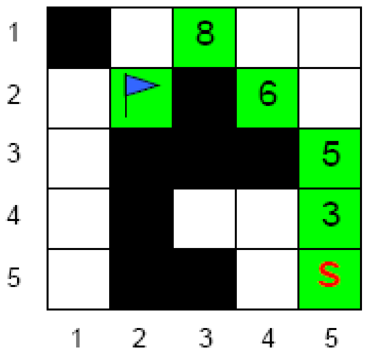
Aplicando varias movidas las novedades son las siguientes:

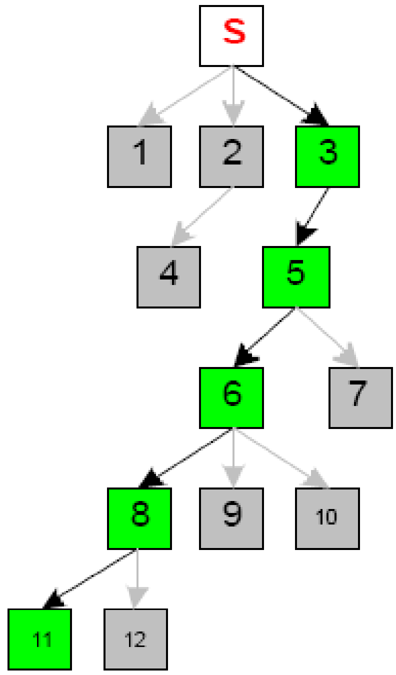
- Abandonamos el camino por la movida 4 debido a que quedamos encerrados.
- Intentamos seguir por el camino de la movida 1 pero tampoco nos llevaba a ningún lado.
- Tomamos el camino de la movida 3 y avanzamos por el único lugar que podemos.





Continuamos avanzando por el mapa, hemos abierto las bases 6 y 7, de las cuales seleccionamos la 6. A partir de allí abrimos las bases 8, 9 y 10, de las cuales como se puede
apreciar en el árbol de la figura anterior, seleccionaremos la 8.





Finalmente encontramos el camino a la meta, que como podemos apreciar es el mas corto de
todos los caminos posibles.

Aplicaciones

El algoritmo A* se puede aplicar en la resolución de diversos problemas. Podríamos, por ejemplo, resolver el cubo de Rubik con él, y el algoritmo nos indicaría como llegar resolver el problema en la menor cantidad de movidas posibles.

En ciertos problemas no es sencillo el hallar una buena función heurística, pero notemos que como la misma debe ser estimada por defecto, especificar que sea cero es una opción válida.

Sin embargo, en dicho caso, si bien el algoritmo funcionará correctamente, se comportará como un ancho-primero ya que sólo pesará el costo del camino recorrido en la elección de la próxima movida y por lo tanto se tenderá a abrir una gran cantidad de bases, empleando para ello un tiempo mayor en comparación con otro caso donde la función heurística sea mas parecida al costo real.

Exploración.

Si la parte de su función de costo castiga los caminos que están sobre el territorio sabido (conocido), los caminos con mayor probabilidad examinan el territorio inexplorado. Estos caminos están bien para unidades de explorador.

Espiar.

Si la parte de la función de costo castiga caminos cerca de las atalayas del enemigo y otras unidades, su unidad tenderá a quedarse en huida. Sin embargo, que para trabajar bien, se debe poner al día el camino de vez en cuando para tener movimientos de unidad enemigos en cuenta.

Construcción de Caminos.

Históricamente, los caminos han sido construidos a lo largo de los caminos que a menudo son usados. Como los caminos son usados cada vez más a menudo, la vegetación es quitada y substituida con la suciedad, y más tarde con la piedra u otro material. Un uso del A* debe encontrar caminos. Los sitios dados donde la gente quiere ir (ciudades, lagos, primaveras, fuentes de minerales, etcétera), encuentran caminos al azar entre estas posiciones importantes. Después del encuentro de cientos o quizás miles de caminos, determine cuales espacios sobre el mapa el más a menudo ocurren sobre caminos. Conviértase en aquellos espacios en caminos. Repita el experimento, con el pionero que prefiere caminos, y usted encontrará más caminos construyendo. Esta técnica puede trabajar para múltiples tipos de caminos también (carreteras, caminos, etc.): los espacios el más comúnmente usados se harían carreteras y los espacios menos comúnmente usados se harían caminos de suciedad o caminos.

La combinación de mapas de influencia, el algoritmo de búsqueda y la línea de vista puede darle modos interesantes de analizar el terreno.

Usando el mismo acercamiento que el edificio de camino, podemos usar el A* para determinar que áreas son las más probables para ser atravesado dado algún juego de puntos de destinación y fuente. Estos puntos, y áreas cerca de ellos, tienden a ser estratégicamente importantes.

Construcción de Ciudades.

Las ciudades a menudo se forman alrededor de recursos naturales como tierras de labranza o fuentes de riqueza mineral. Como la gente de estas ciudades, comercia el uno con el otro, ellos necesitan rutas comerciales. En esta situación seria factible utilizar el algoritmo A* para encontrar sus rutas de comercio, y luego marcar el valor de un día de viajes sobre estas rutas. Después de que una caravana viaja durante un día, esto necesitará un lugar para pararse: ¡un lugar perfecto para una ciudad! Las ciudades que mienten (están) a lo largo de más que una ruta de viajes son grandes sitios para negociar pueblos, que eventualmente se convierten en ciudades.

Una combinación de construcción de edificios y construcción de ciudades puede ser útil para producir mapas realistas, para guiones o para mapas aleatorios