Búsqueda Heurística

Tomás de la Rosa

Esquema

Esquema

Exploración de Alternativas

- Cuando la solución a un problema de búsqueda es desconocido, podemos intentar construirla mediante prueba y error de las alternativas
- La concatenación de alternativas produce un efecto multiplicativo de las opciones, lo que lo transforma en un problema exponencial
- El tiempo disponible para resolver los problemas es lineal y los recursos son limitados
- Objetivo: Buscar algoritmos que encuentren una solución en tiempo polinomial

Clases de Problemas de Búsqueda

- Búsqueda de caminos para un agente
- Búsqueda para varios agentes
- Problemas de satisfacción de restricciones

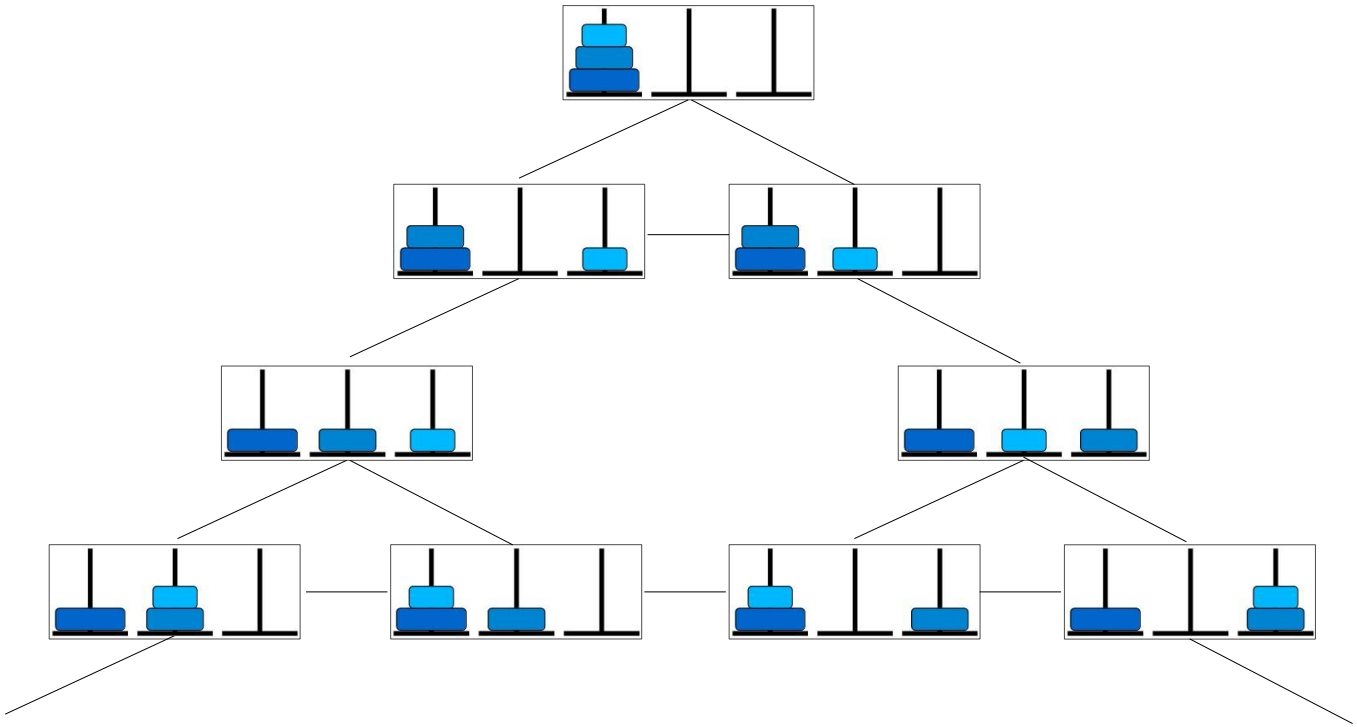
Modelado y Resolución de Problemas

- Modelo matemático
  - Define qué es un problema
  - Define qué es una solución
- Lenguaje de representación
- Algoritmos de resolución

Búsqueda en el Espacio de Estados

- Descripción del Modelo
  - Espacio de estados S
  - Estado inicial $s_0 \in S$
  - Conjunto de estados meta G
  - Conjunto de operadores o acciones $a_i \in A$ con un coste asociado $c(a_i)$
  - Función de transición $f(S, A) \rightarrow S'$, que indica como transformar unos estados en otros al aplicar una acción
- Solución al problema: Secuencia de acciones $a_1, a_2, \ldots a_n$ que transforma el estado inicial en estado meta

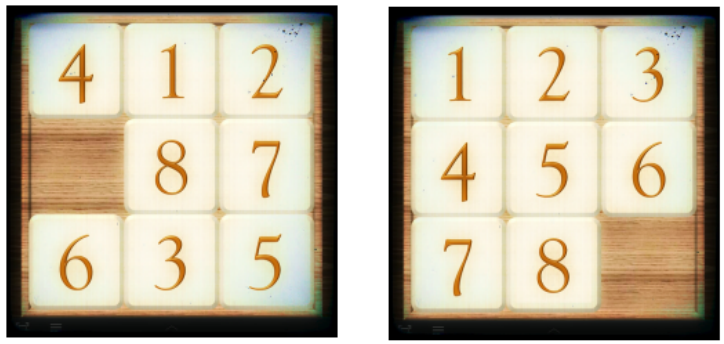
Grafo de Estados



Ejemplos de Espacio de Estados

Cubo de Rubik

8-Puzzle



Búsqueda Espacio de Estados: Ejemplo

Problema de las Garrafas

Se tienen dos garrafas de agua, una de cinco litros de capacidad y otra de tres. Ninguna de ellas tiene marcas de medición. Se tiene una bomba que permite llenar las jarras de agua, vaciarlas, y traspasar contenido de una garrafa a otra. ¿Cómo se puede lograr tener exactamente cuatro litros de agua en la jarra de cinco litros de capacidad?

Modelado en el Espacio de Estados: Ejemplo

- Espacio de Estados: Pares (x, y) con $x \in \{0, \ldots, 5\}$, e $y \in \{0, \ldots, 3\}$
- Estado Inicial: $(0, 0)$
- Estado Meta: $(4, ?)$, que de forma explícita sería cualquiera de $(4, 0), (4, 1), (4, 2), (4, 3)$
- Acciones:
  - Llenar garrafa grande, llenar garrafa pequeña
  - Vaciar garrafa grande, vaciar garrafa pequeña
  - Verter grande en pequeña, verter pequeña en grande

Modelado de Transiciones

- La función de transición representa:
  - El conjunto de acciones aplicables en cada estado
  - El estado resultante de aplicar una acción desde un estado previo
- Las restricciones que limitan las acciones para que sean aplicables las denominamos precondiciones
- Al ejecutar una acción aplicamos sus post-condiciones o efectos

Acción	Precondición	Efecto
Llenar Grande (x, y)	$x < 5$	$(5, y)$
Vaciar Grande (x, y)	$x > 0$	$(0, y)$
Verter Grande en Peq. (x, y)	$x > 0$	$(x - min\{x, 3 - y\}, y + min\{x, 3 - y\})$

Algoritmos de Búsqueda de un Agente

- Sistemáticos o Deterministas
  - Búsqueda No Informada o Ciega
    - ★ Se exploran todas las alternativas. Las metas tienen un rol pasivo
    - ★ Algoritmos: Primero-en-profundidad, Primero-en-amplitud, coste uniforme (Dijkstra)
  - Búsqueda Heurística
    - ★ Se guía la búsqueda explorando primero las alternativas más prometedoras
    - ★ Ejemplo: búsqueda avariciosa, primero-el-mejor (BFS),
- Estocásticos (próxima sesión):
  - El orden de explorar las alternativas tiene un componente aleatorio
  - Algoritmos: simulated annealing, algoritmos genéticos

Algoritmos de Búsqueda Determinista

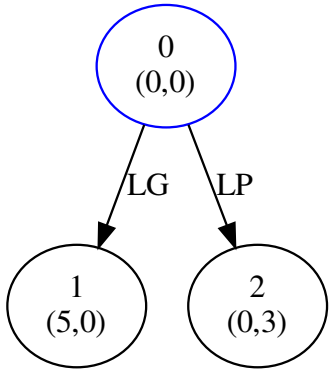
- La exploración sistemática de alternativas genera un árbol de búsqueda
- En implementación normalmente se estructura como:
  - Lista Abierta: Nodos hoja, candidatos a ser explorados
  - Lista Cerrada: Nodos expandidos
- Los algoritmos de búsqueda se diferencian por:
  - Cómo seleccionar el siguiente nodo a explorar
  - Cómo agregar los sucesores de un nodo a la lista abierta
- Notación
  - $f(n)$: función de evaluación que asigna un valor al nodo n
  - $g(n)$: coste desde el estado inicial hasta el nodo n
  - $h(n)$: heurística o coste estimado desde el nodo n hasta el objetivo

Esquema

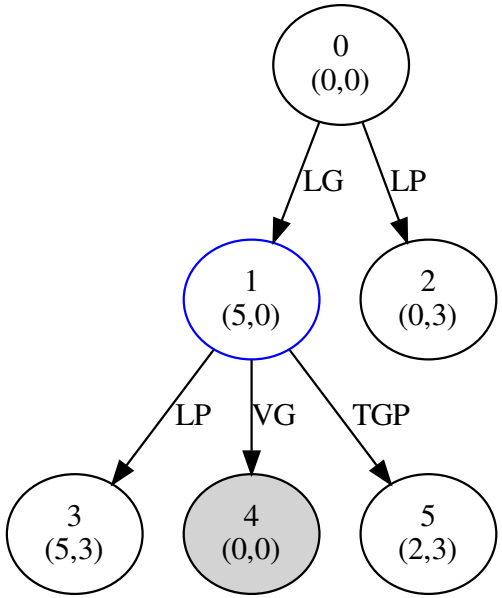
Acciones (Problema de las Garrafas)

1. Llenar grande (LG)
2. LLenar pequeña (LP)
3. Vaciar grande (VG)
4. Vaciar pequeña (VP)
5. Traspasar grande a pequeña (TGP)
6. Traspasar pequeña a grande (TPG)

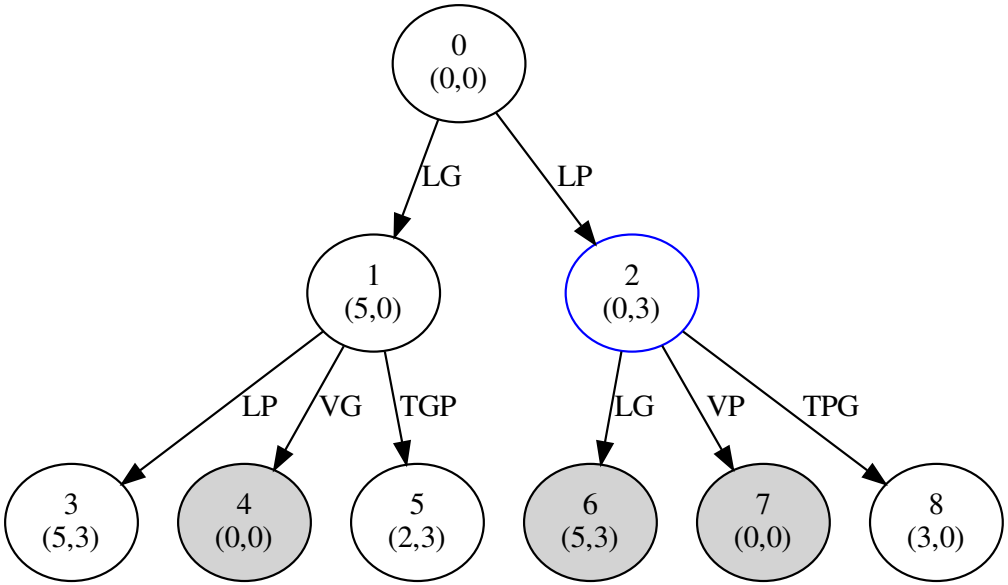
Primero en Amplitud



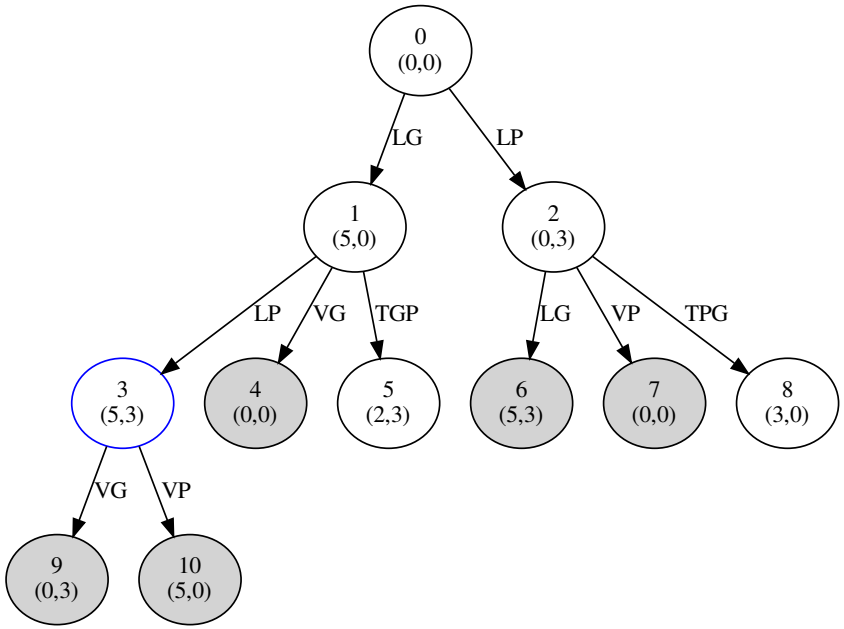
Primero en Amplitud



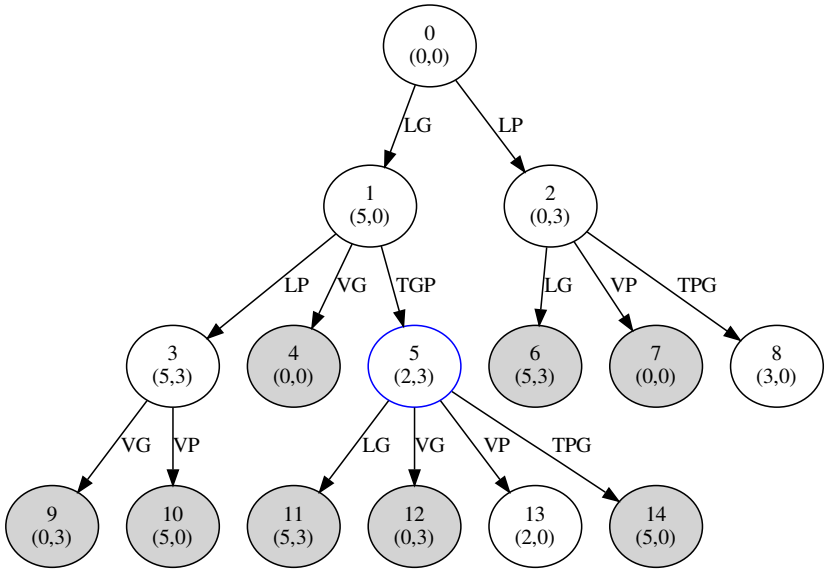
Primero en Amplitud



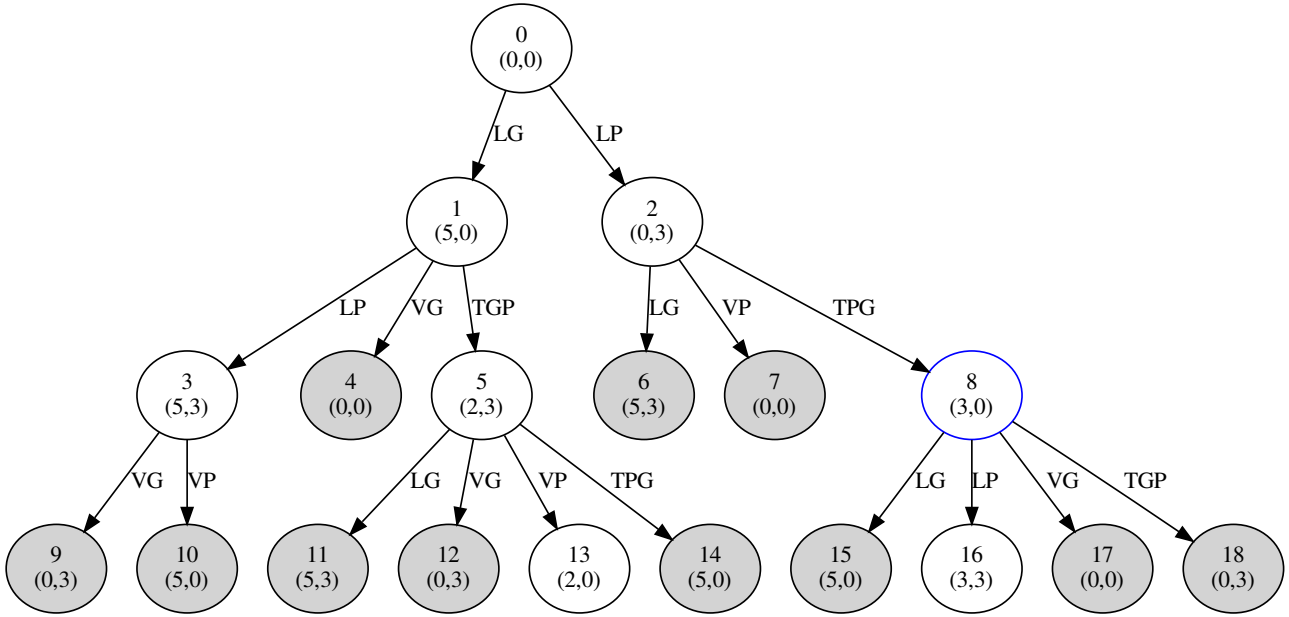
Primero en Amplitud



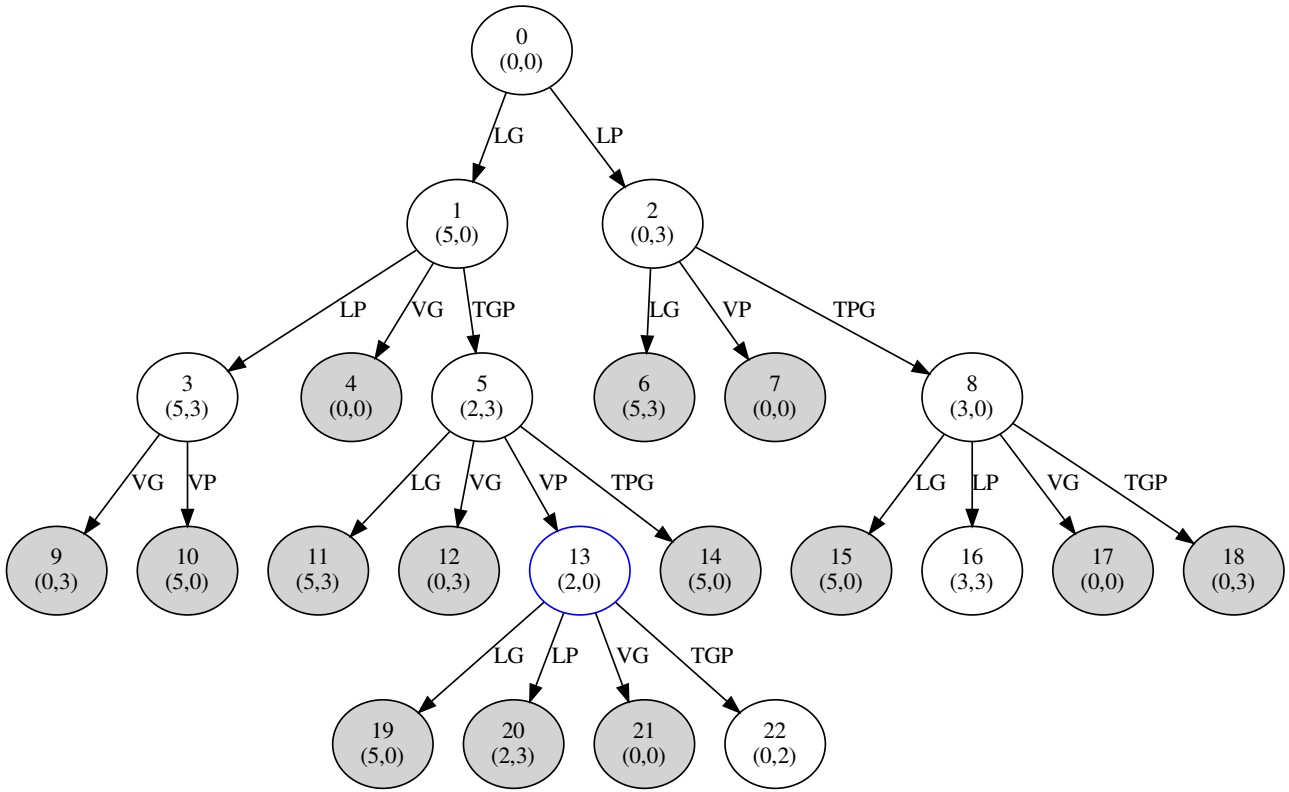
Primero en Amplitud



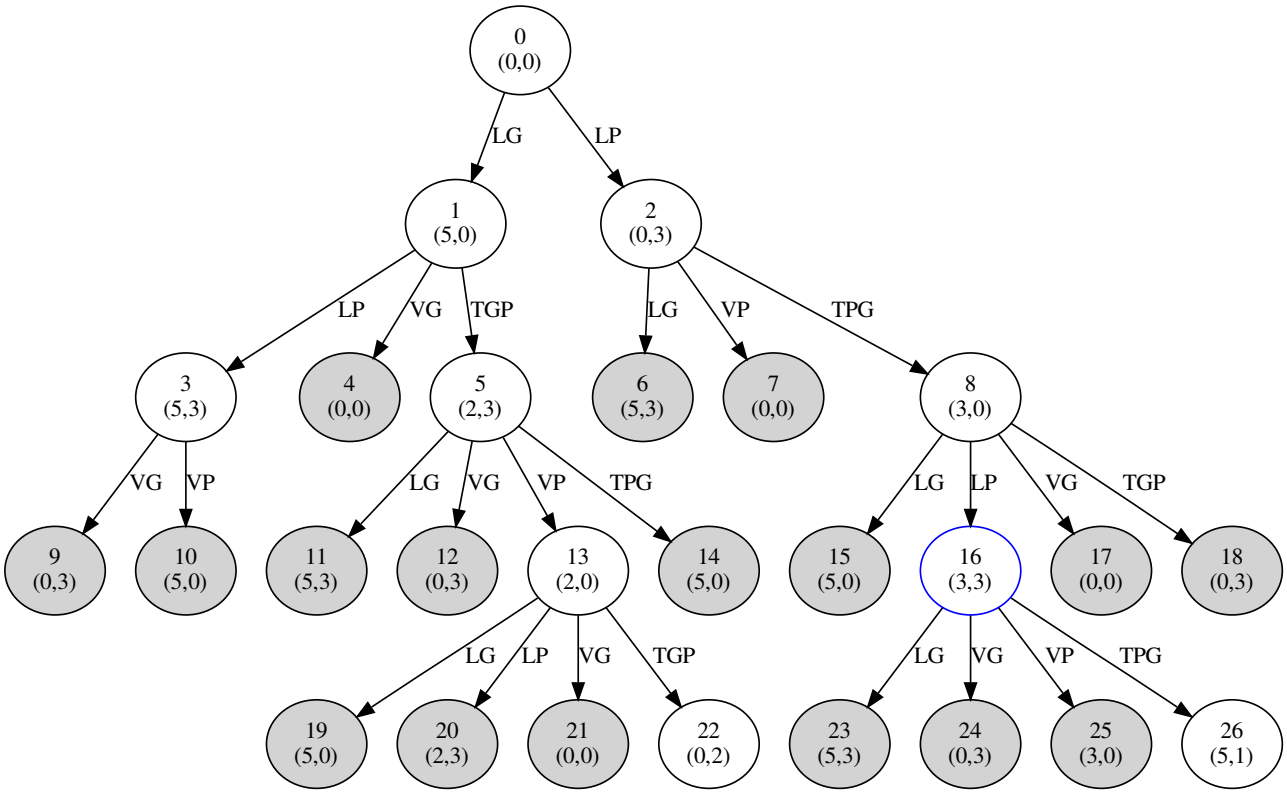
Primero en Amplitud



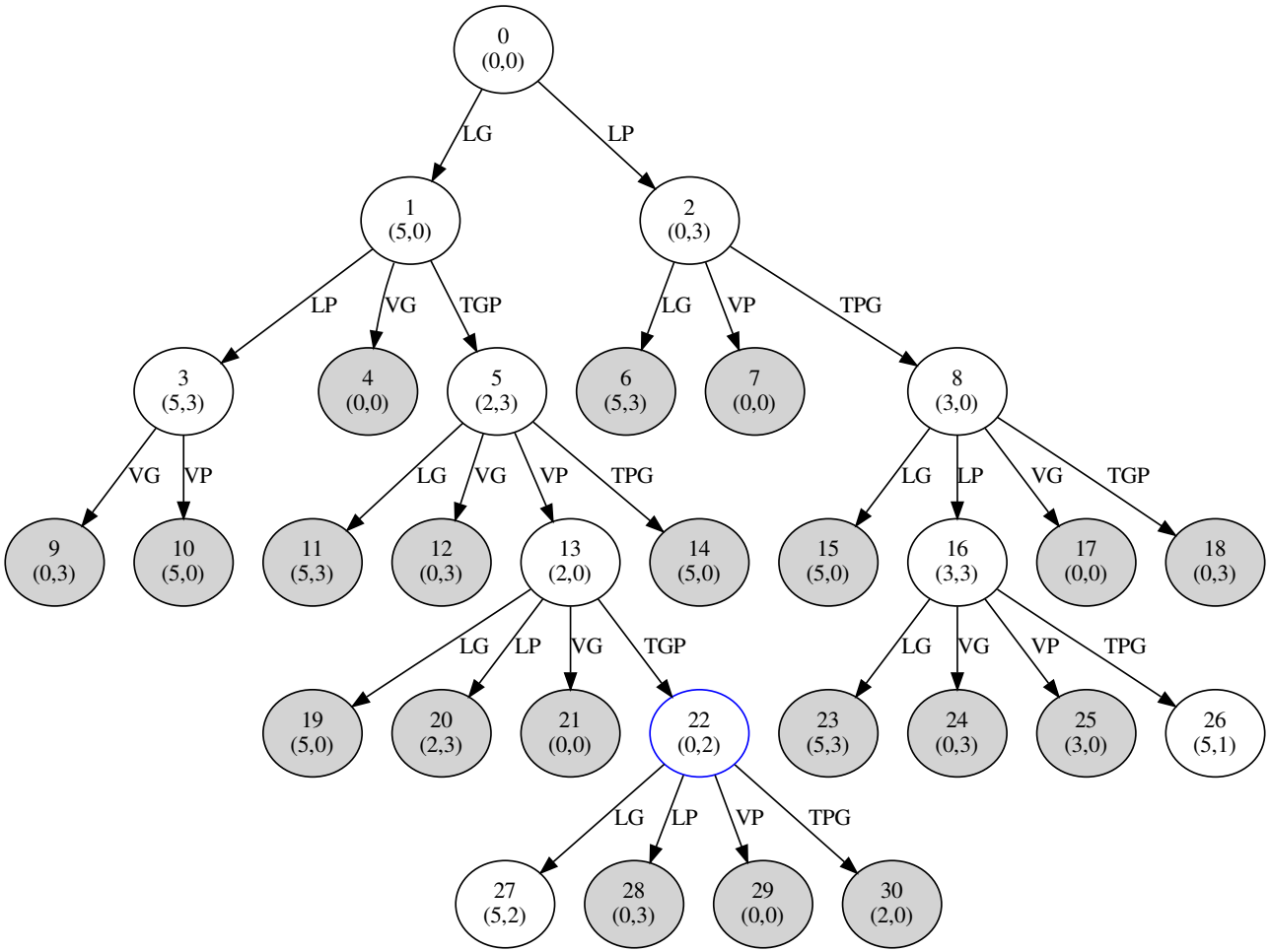
Primero en Amplitud



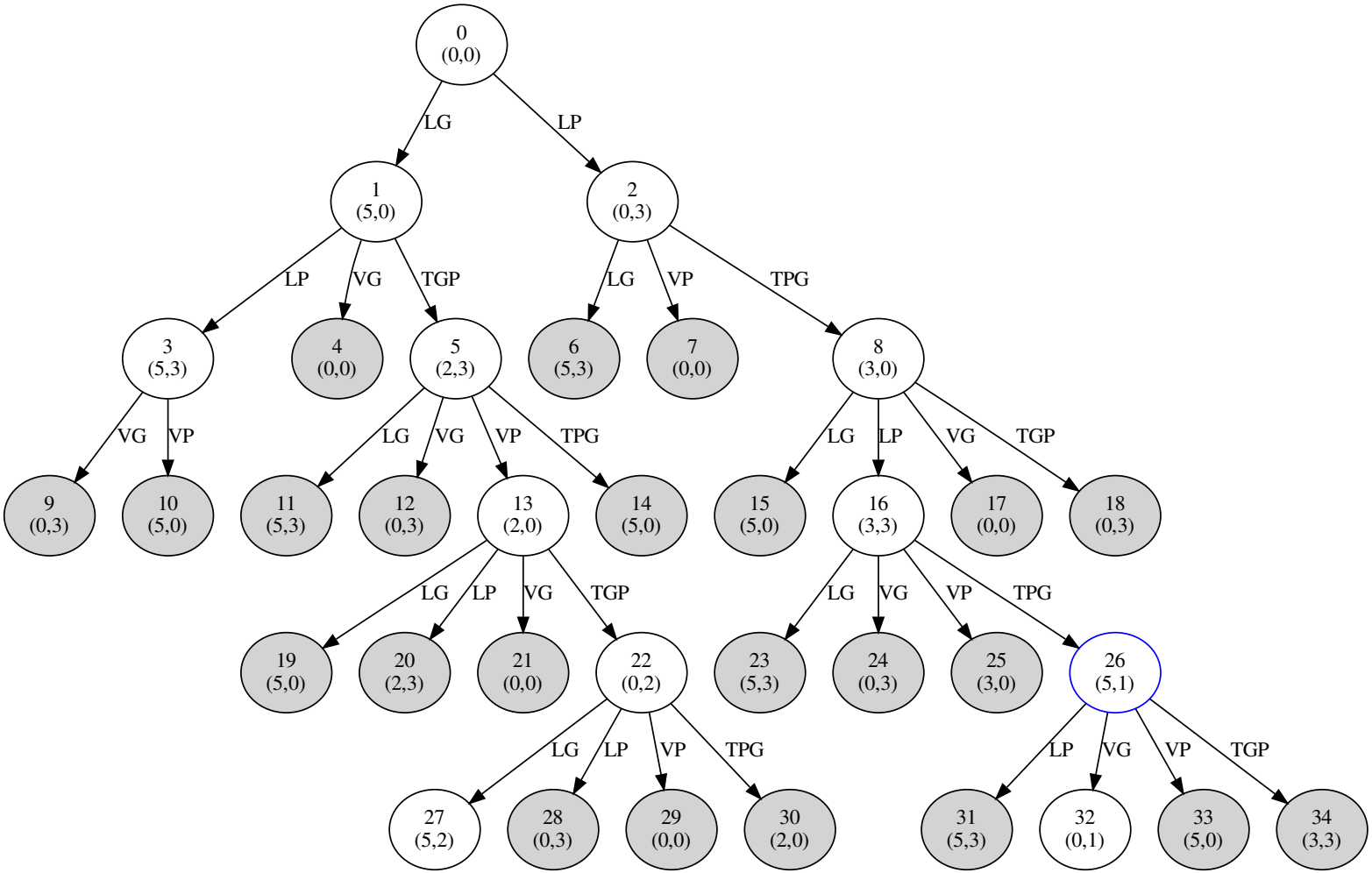
Primero en Amplitud



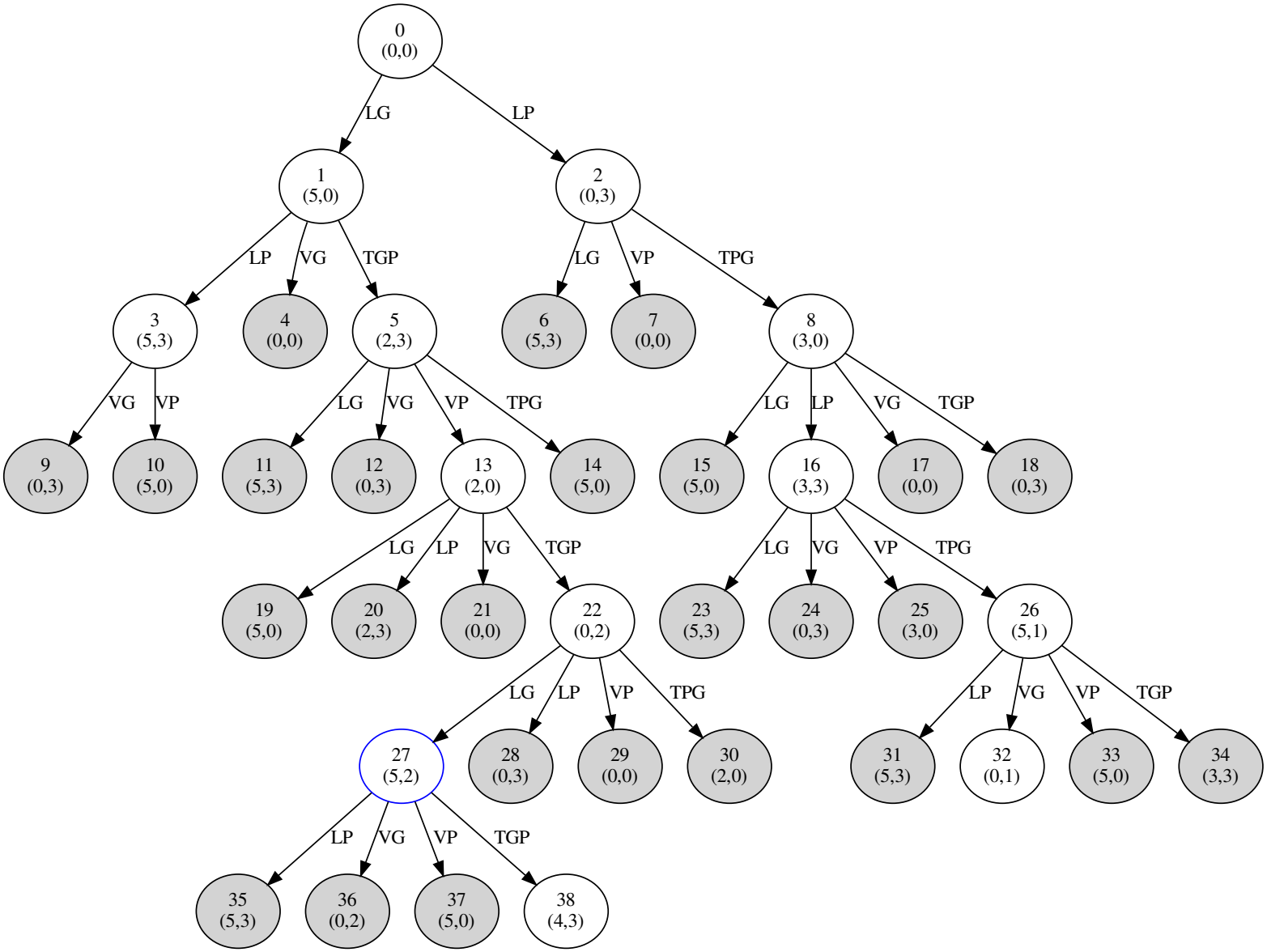
Primero en Amplitud



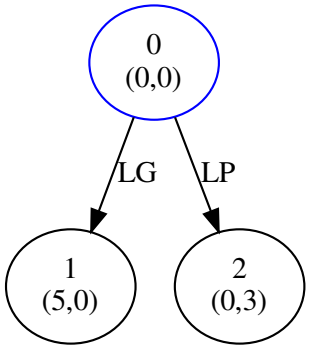
Primero en Amplitud



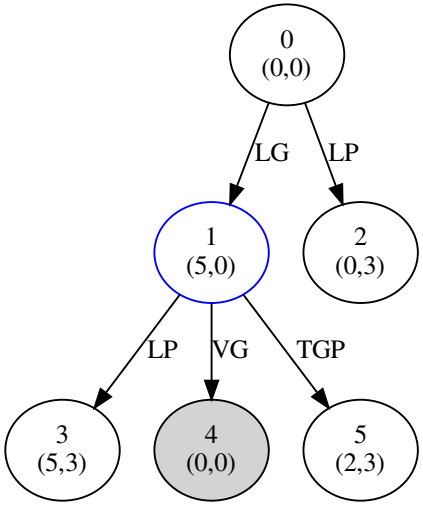
Primero en Amplitud



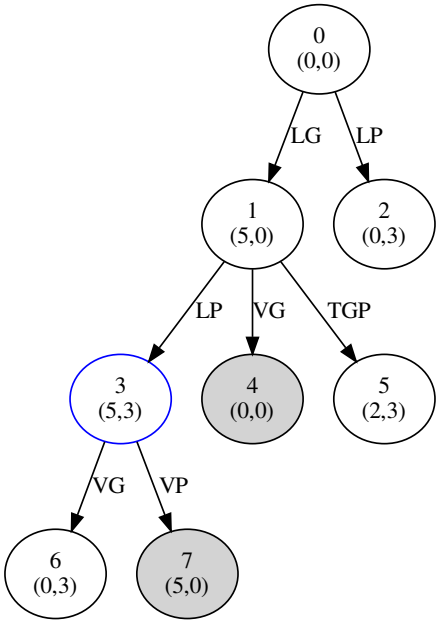
Primero en Profundidad



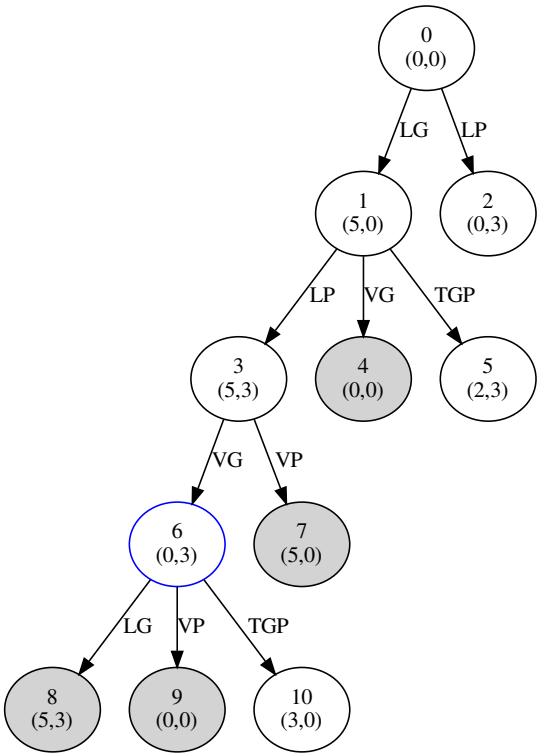
Primero en Profundidad



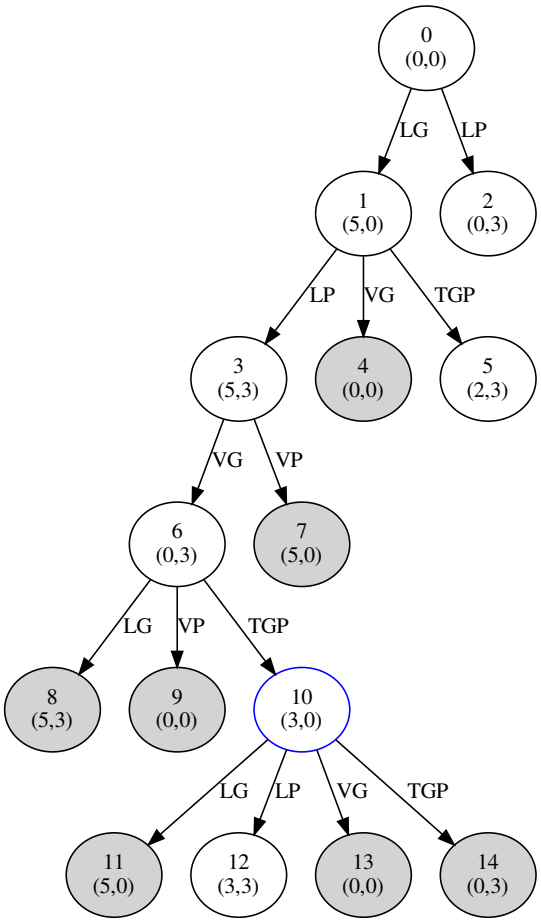
Primero en Profundidad



Primero en Profundidad



Primero en Profundidad



Esquema

Heurísticas

- Información sobre el problema que se utiliza para guiar el proceso de búsqueda
- Suele ser información imperfecta, pero que es mucho más fácil de obtener que la solución que se busca
- Diferentes tipos:
  - Funciones de evaluación: Dan una ordenación de los nodos
  - Estimación de coste hasta la meta:
    - ★ Puede obtenerse mediante relajación de restricciones o en una versión simplificada del problema
    - ★ Cuando no se sobre-estima el coste hasta la meta las heurísticas son admisibles
  - Reglas de control: selección o poda de nodos por criterios del dominio de aplicación

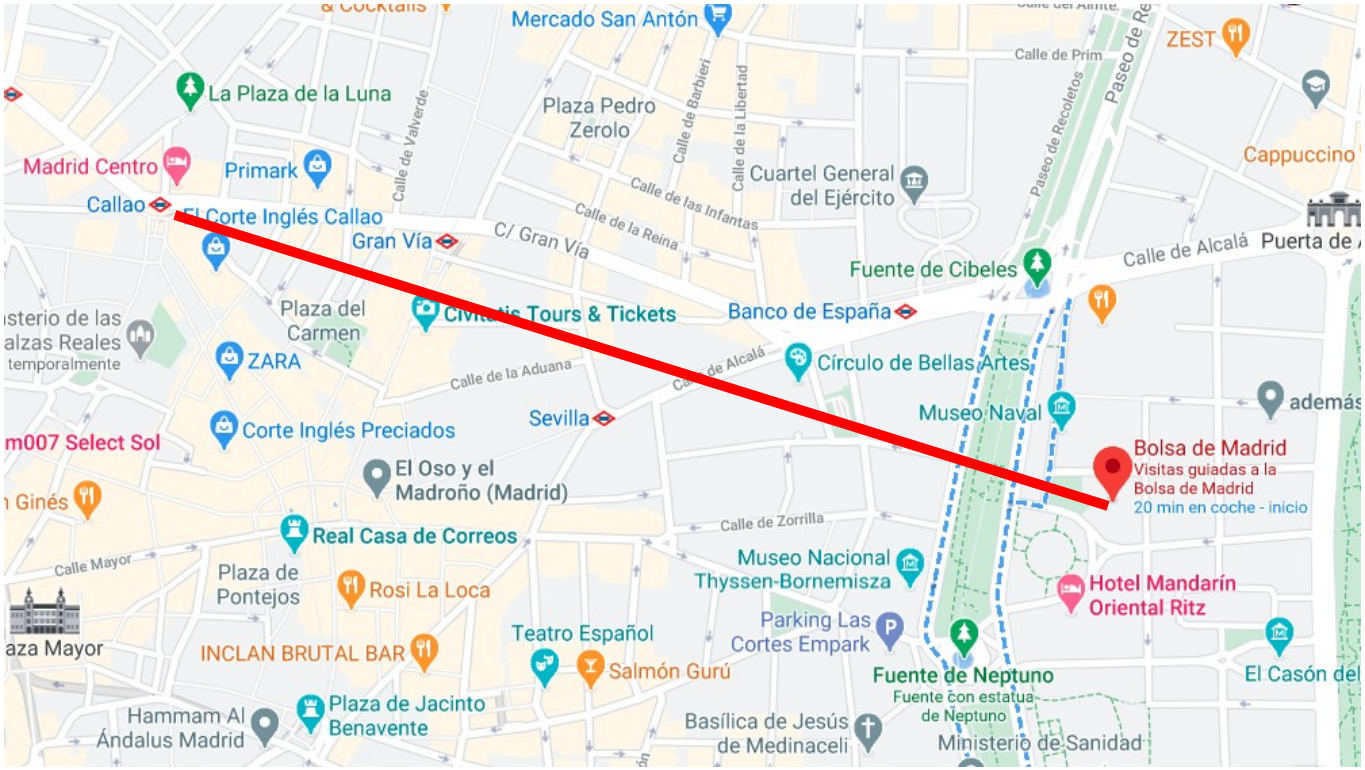
Algoritmo de Escalada

Idea General

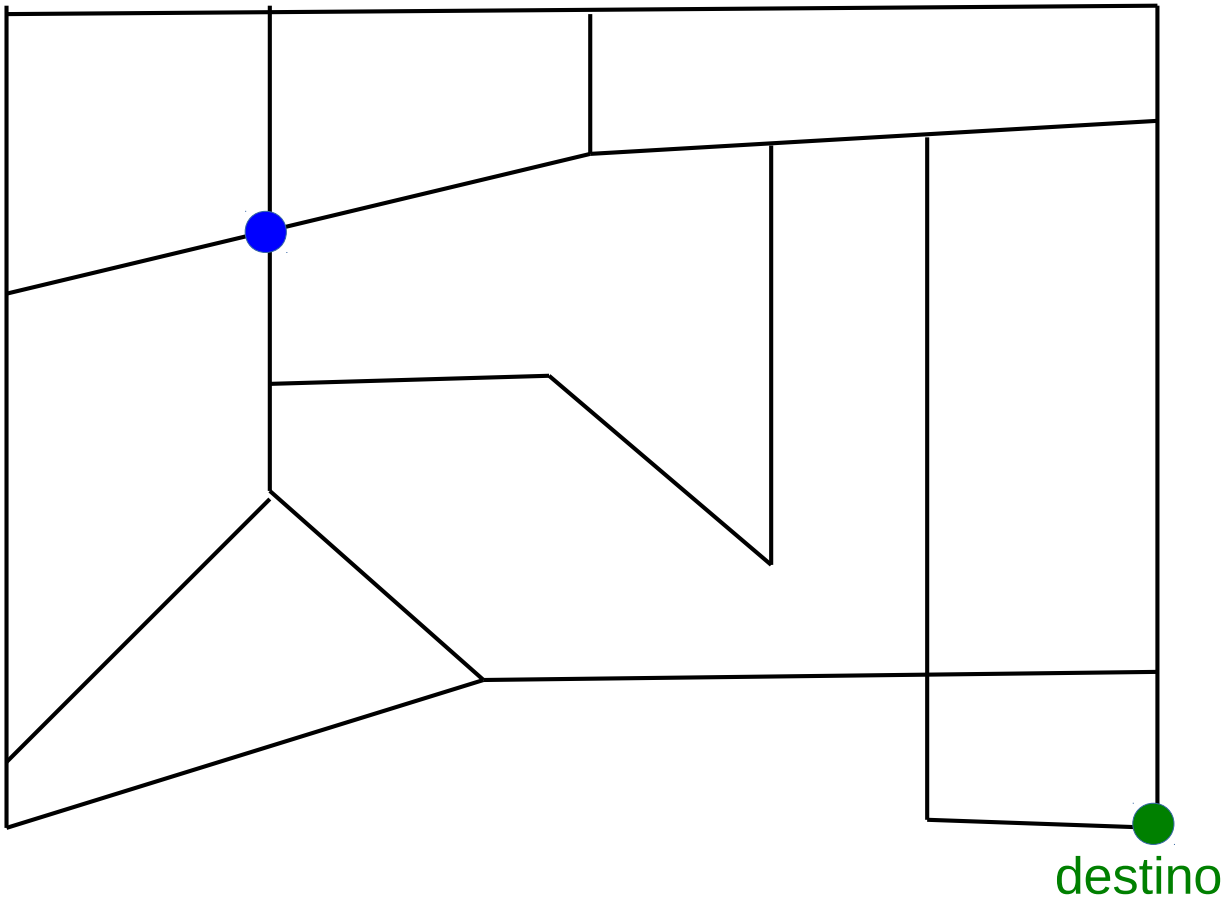
Elegir a cada momento el mejor nodo entre los sucesores directos

- Técnica avariciosa, no reconsidera el camino elegido
- Necesitamos una función de evaluación $f(n)$ que nos diga cuál es el mejor nodo
  - El valor más alto (maximizar $f(n)$)
  - El valor más bajo (minimizar $f(n)$)
  - Utilizando una heurística que nos estime el camino hasta la meta. En este caso $f(n) = h(n)$

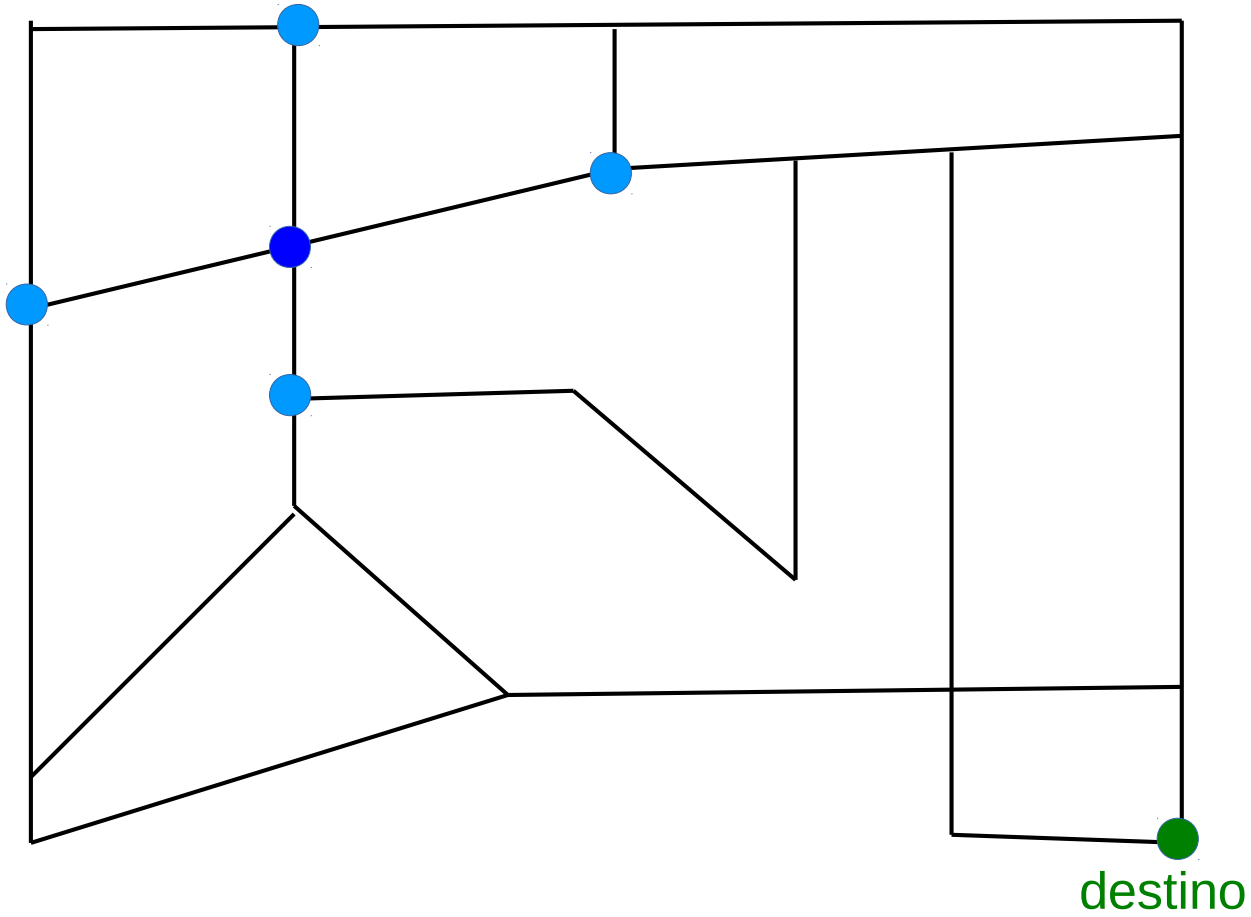
Heurísticas en el Plano



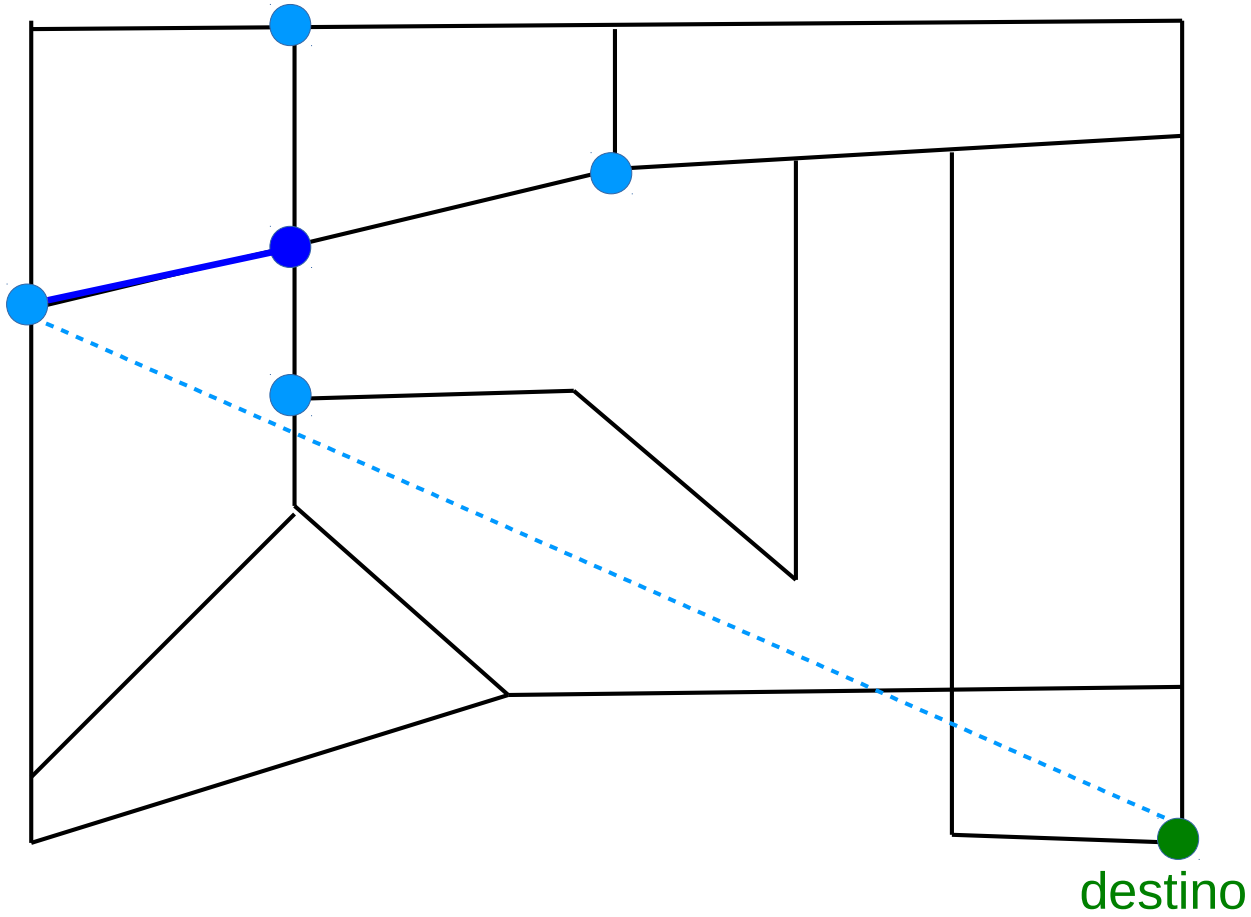
Heurísticas en el Plano



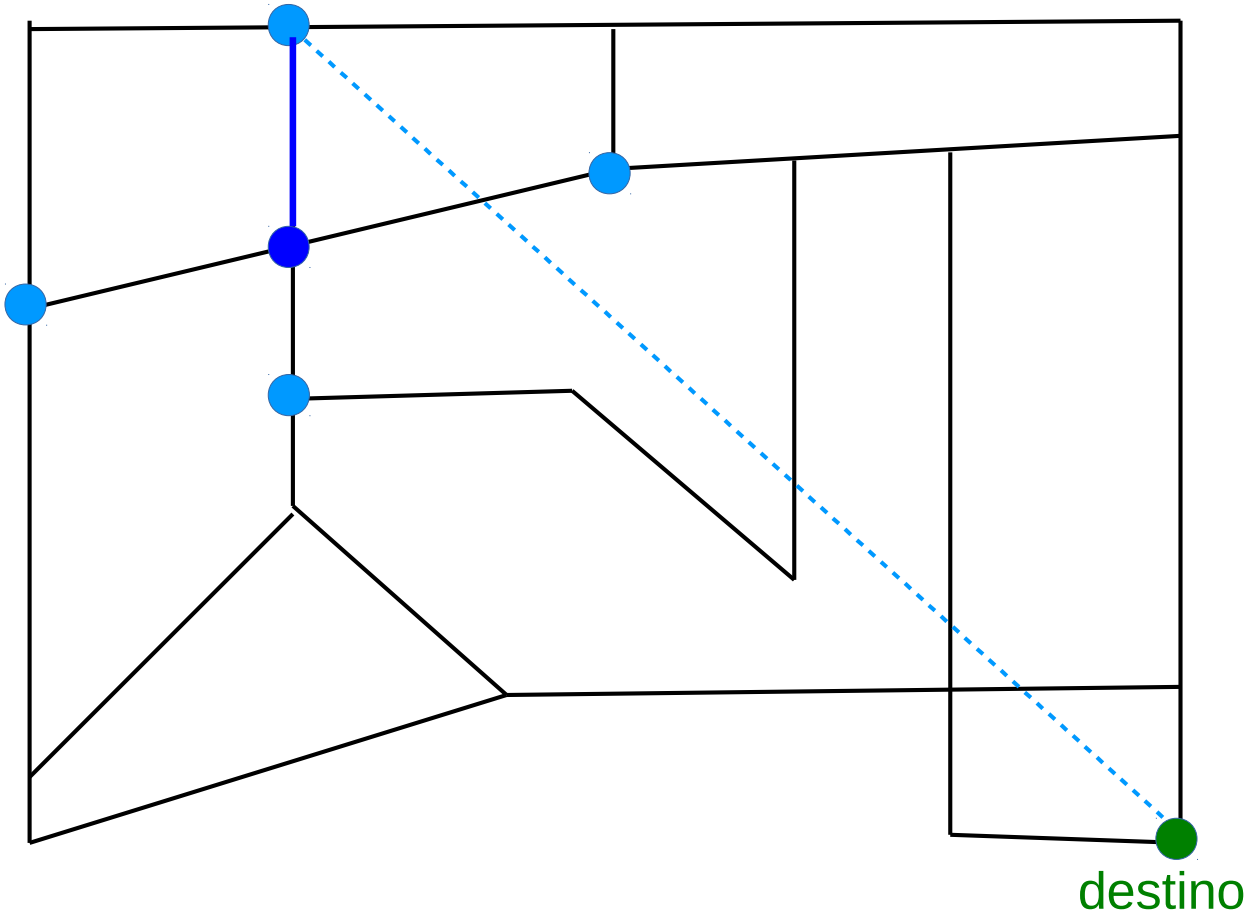
Heurísticas en el Plano



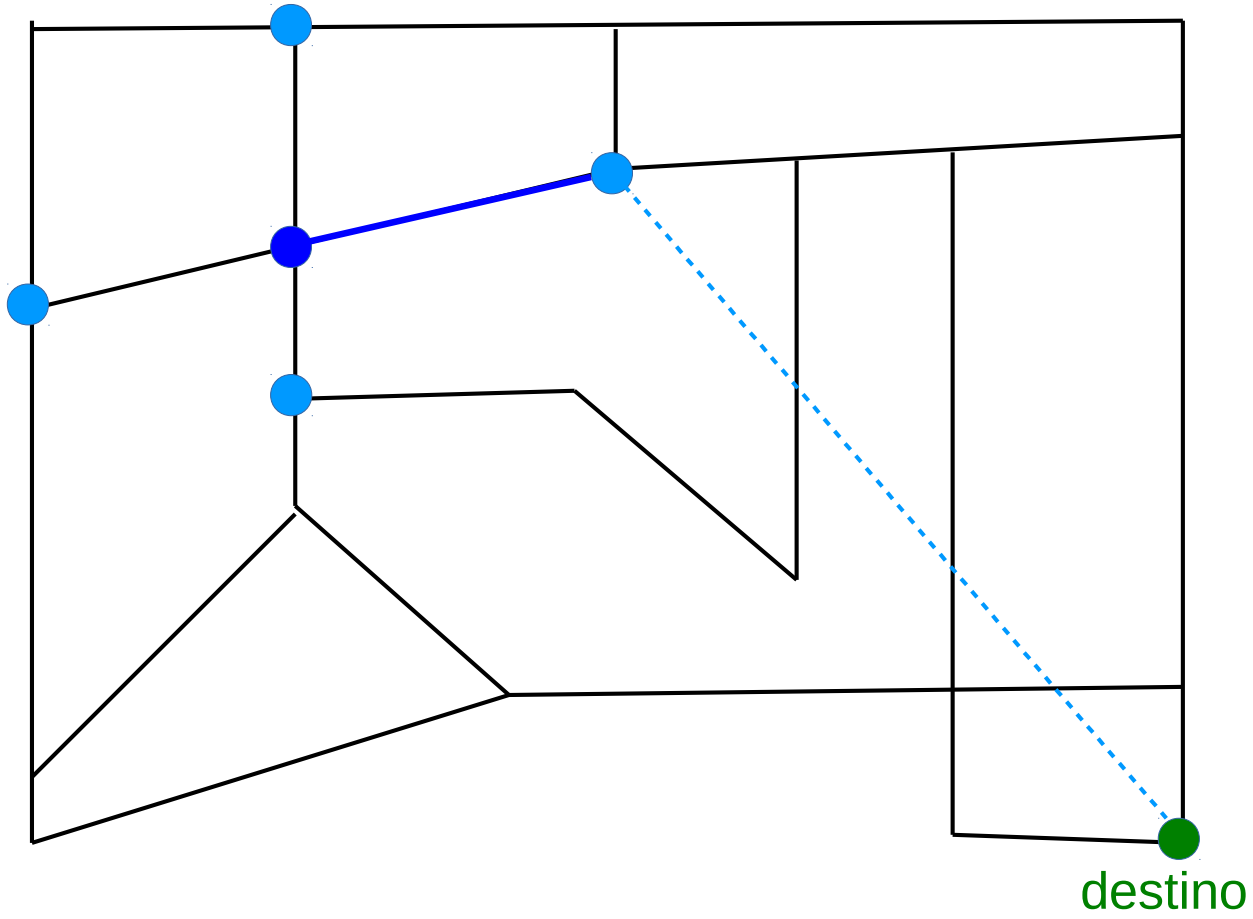
Heurísticas en el Plano



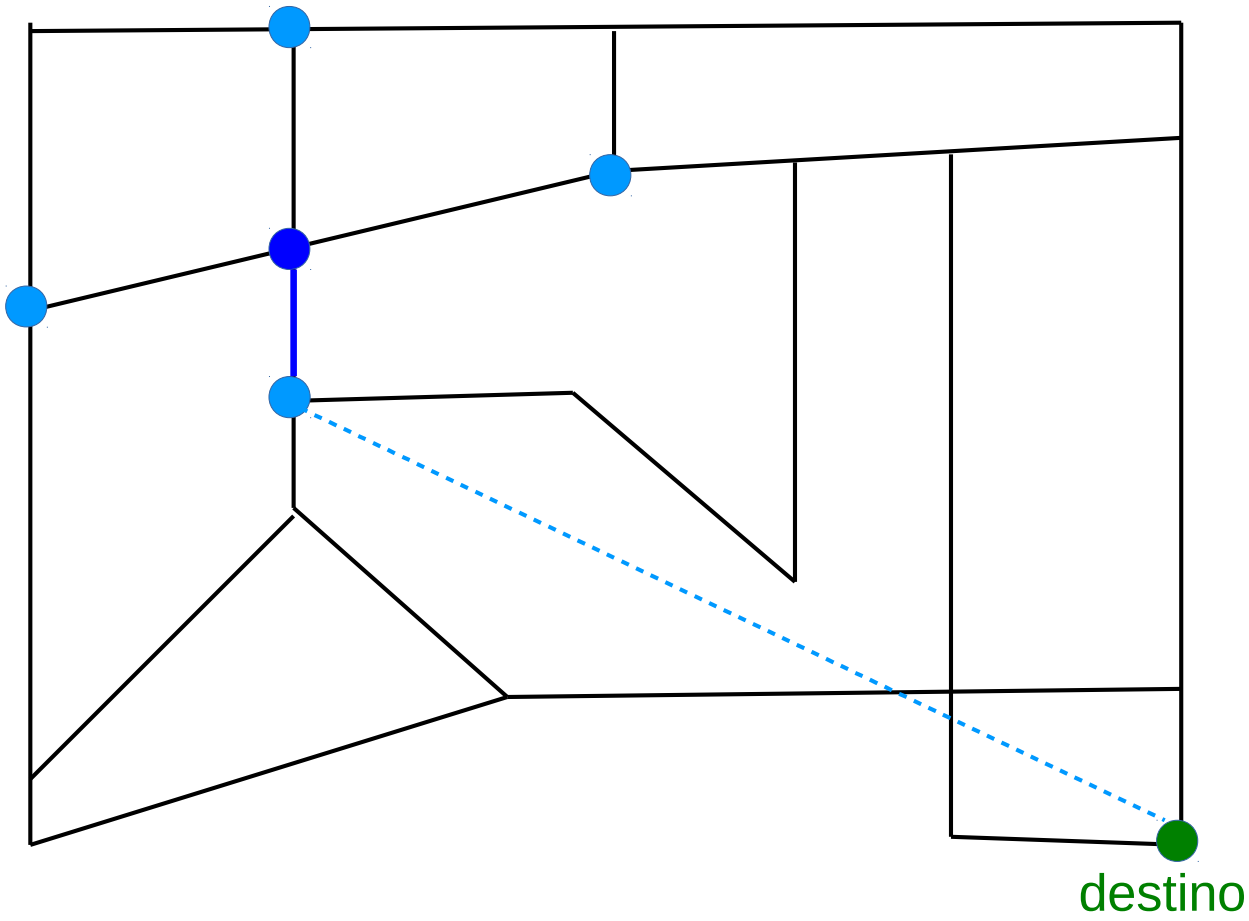
Heurísticas en el Plano



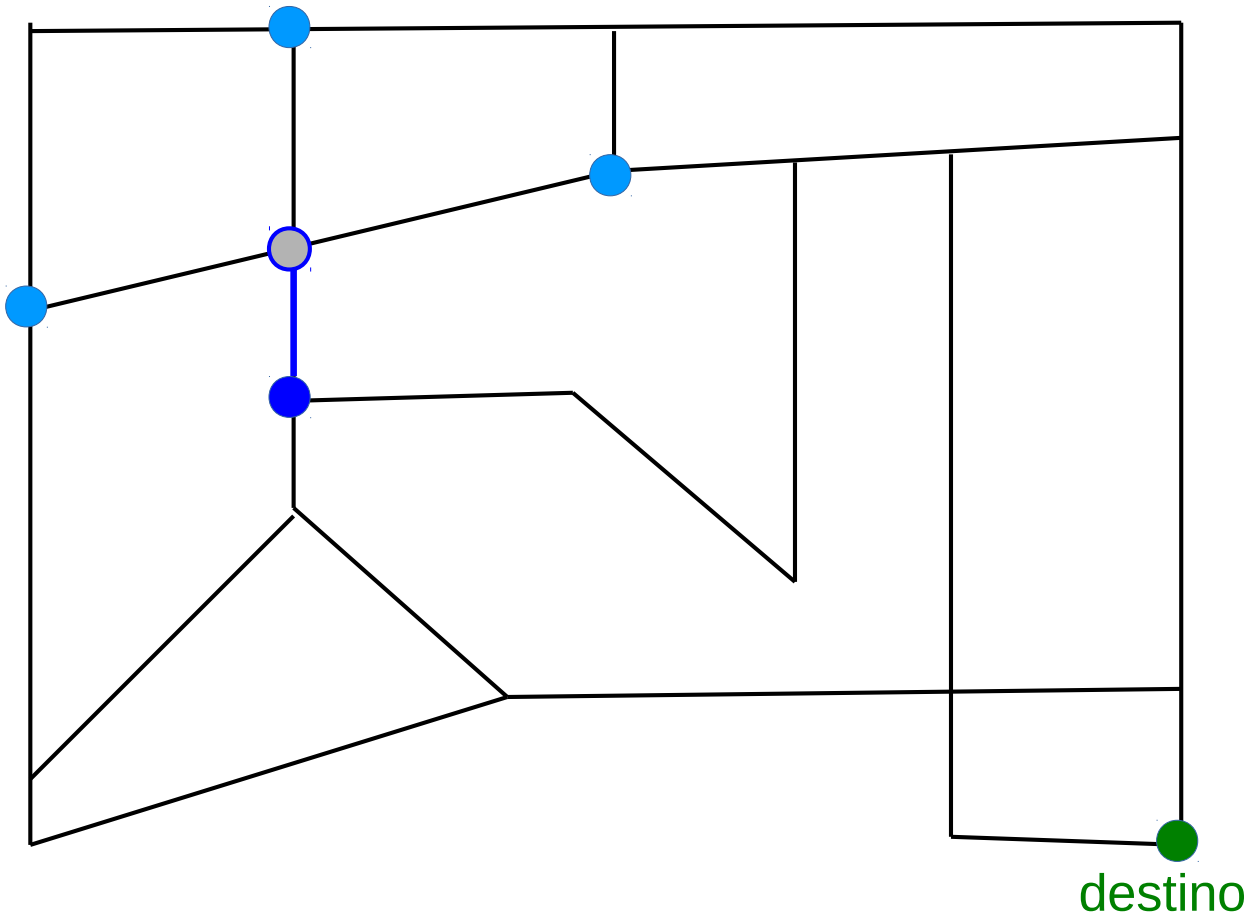
Heurísticas en el Plano



Heurísticas en el Plano



Heurísticas en el Plano



Algoritmo de Primero el Mejor

Idea General

Elegir a cada momento el mejor nodo entre todos los sucesores que hayan sido generados y no explorados

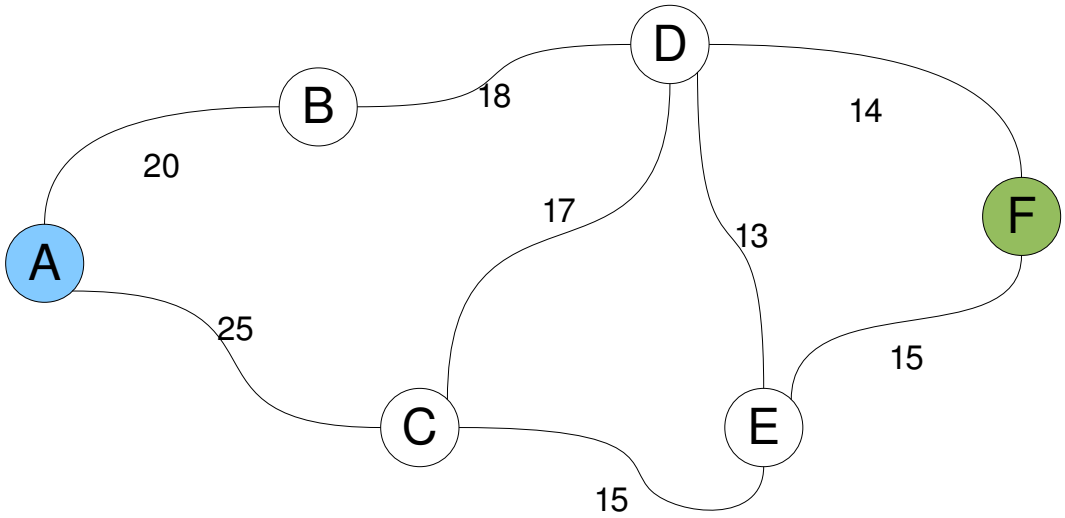
- **BFS**: *Best-First Search*
- Búsqueda global, reconsidera los caminos
- Igualmente, necesitamos una función de evaluación $f(n)$ que nos diga cuál es el mejor nodo

Algoritmo A*

- Realiza una búsqueda BFS con función de evaluación $f(n) = g(n) + h(n)$
  - g(n): coste desde el estado inicial hasta el nodo n
  - h(n): heurística que mide el coste estimado hasta la meta
  - h*(n): coste real desde el estado actual hasta la meta
- Encuentra la solución óptima si:
  - h(n) es admisible, $h(n) \leq h^*(n)$
  - el coste de cada acción es positivo

Ejemplo A*

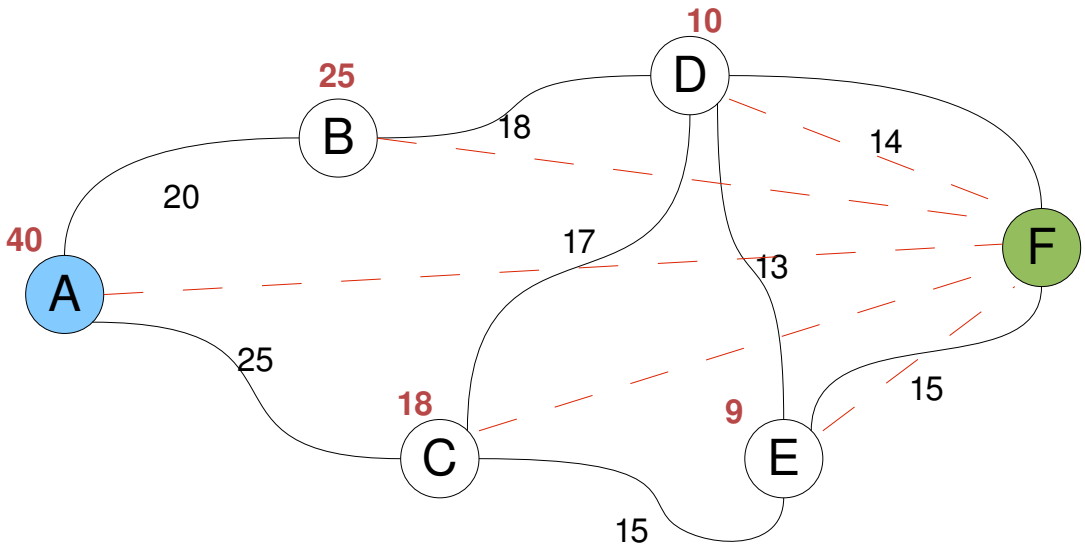
Se quiere determinar el camino más corto de la ciudad A a la ciudad F



Cada arco tiene el coste de ir de una ciudad a otra

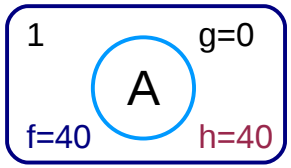
Ejemplo A*

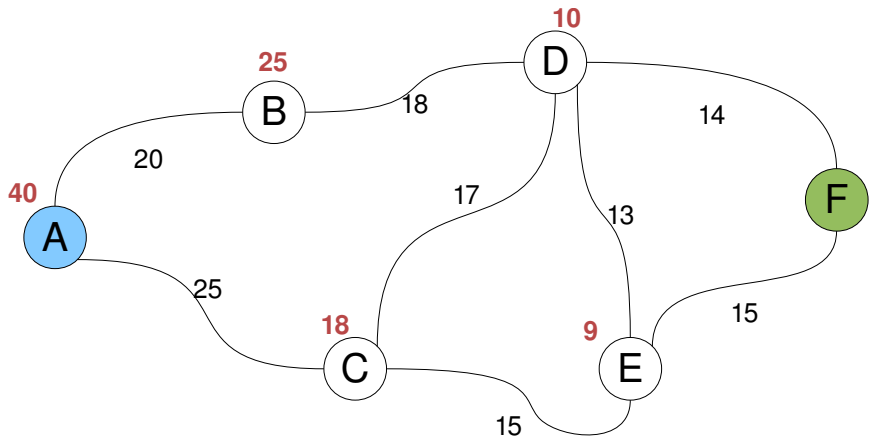
Se quiere determinar el camino más corto de la ciudad A a la ciudad F



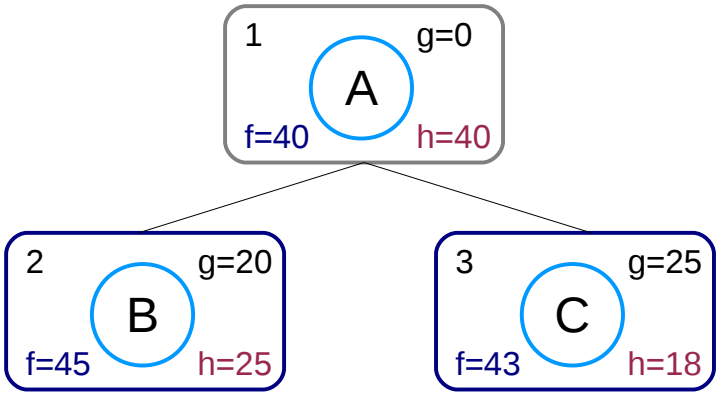
Desde cada ciudad se puede *estimar* la distancia que queda para llegar a F

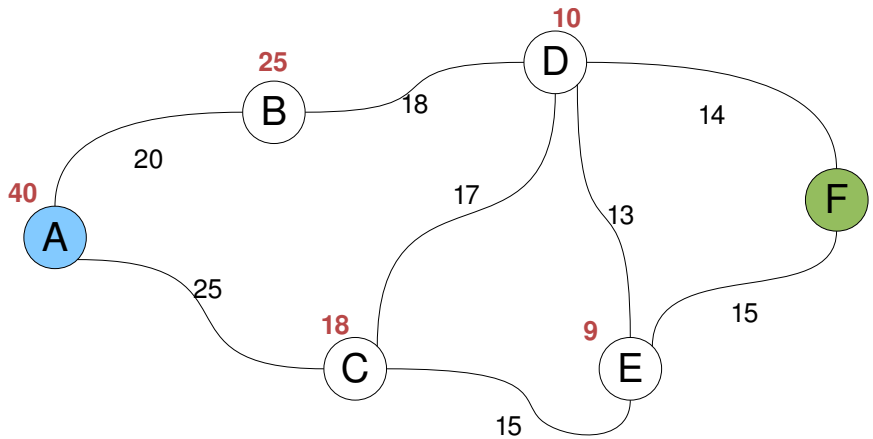
Ejemplo A*



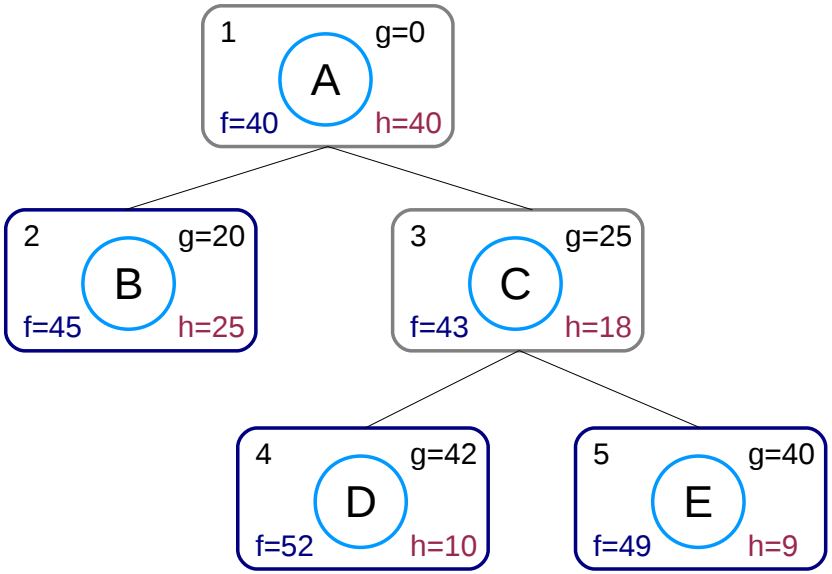


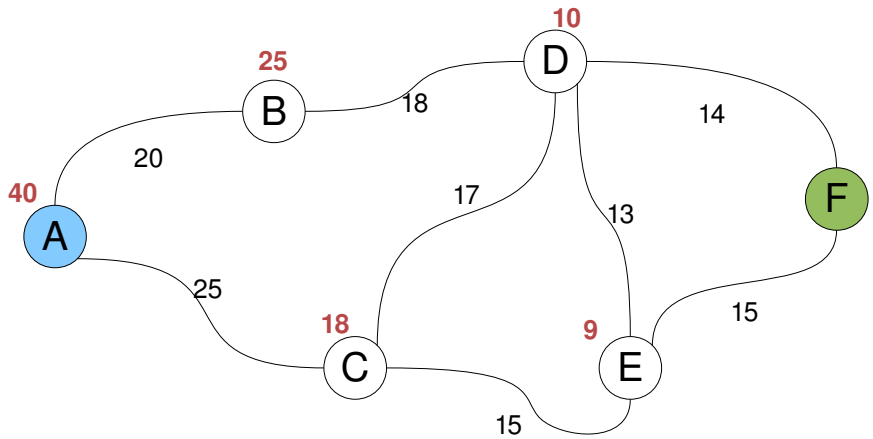
Ejemplo A*


1
g=0
A
f=40
h=40
2
g=20
B
f=45
h=25
3
g=25
C
f=43
h=18

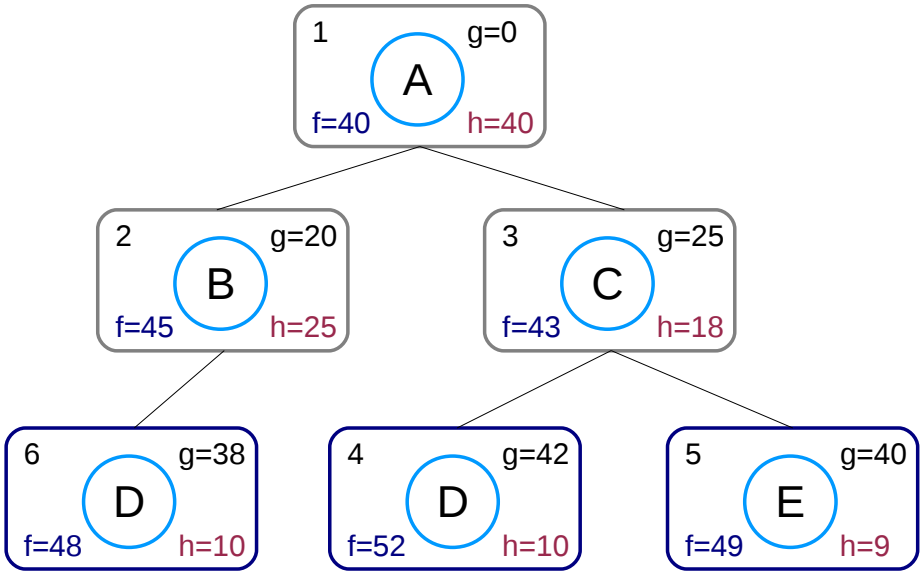

10
25
D
B
18
14
20
40
17
13
F
A
25
18
9
15
C
E
15

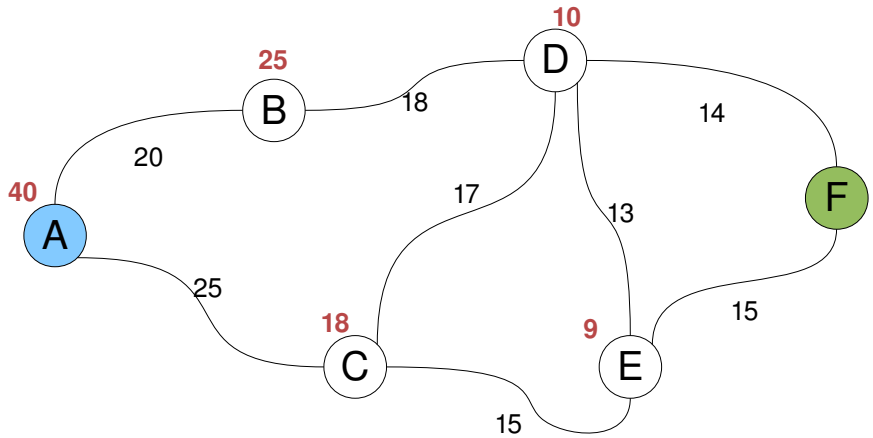
Ejemplo A*


1
g=0
A
f=40
h=40
2
g=20
B
f=45
h=25
3
g=25
C
f=43
h=18
4
g=42
D
f=52
h=10
5
g=40
E
f=49
h=9

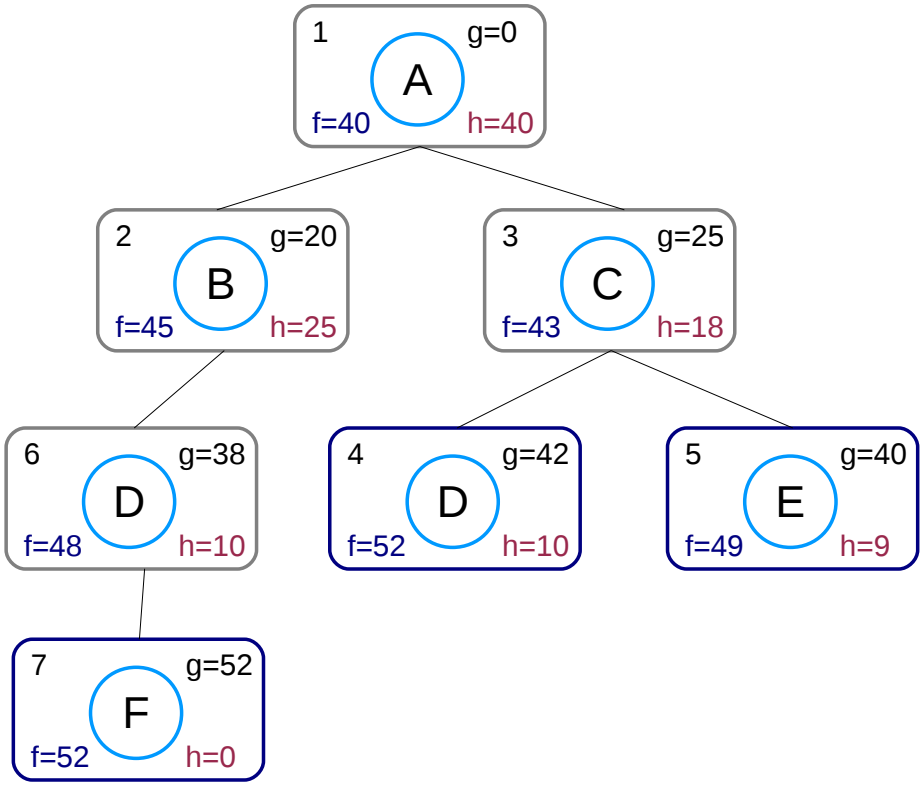

10
D
25
B
18
14
20
40
A
17
13
F
25
18
9
15
C
E
15

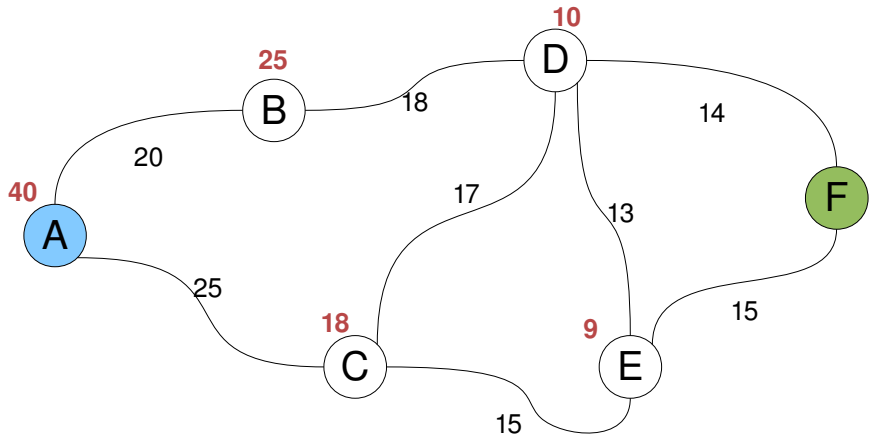
Ejemplo A*


1
g=0
A
f=40
h=40
2
g=20
B
f=45
h=25
3
g=25
C
f=43
h=18
6
g=38
D
f=48
h=10
4
g=42
D
f=52
h=10
5
g=40
E
f=49
h=9


10
25
D
B
18
14
20
40
17
13
F
A
25
18
9
15
C
E
15

Ejemplo A*





Se continúa, pues el algoritmo termina cuando se expande el nodo que contiene la solución (nodo 7).

Consideraciones sobre la heurística

- Si $h_1(n) <= h_2(n)$ en todos los nodos, entonces $h_2(n)$ está **más informada** que $h_1(n)$ y servirá para expandir menos nodos
- Extremos
  - $h(n) = 0$ para cada nodo: no se tiene información. Es equivalente al **algoritmo de Dijkstra**
  - $h(n) = h^*(n)$ para cada nodo: se tiene información perfecta
- No tiene sentido dedicar más coste computacional a calcular una buena $h(n)$ que a realizar la búsqueda equivalente: equilibrio