Tecnológico Nacional de México
Campus Colima

Ingeniería en Sistemas Computacionales

Inteligencia Artificial

Parcial 3

Docente: Noel García Díaz

12:00 - 14:00

Actividad 1: Algoritmos de búsqueda

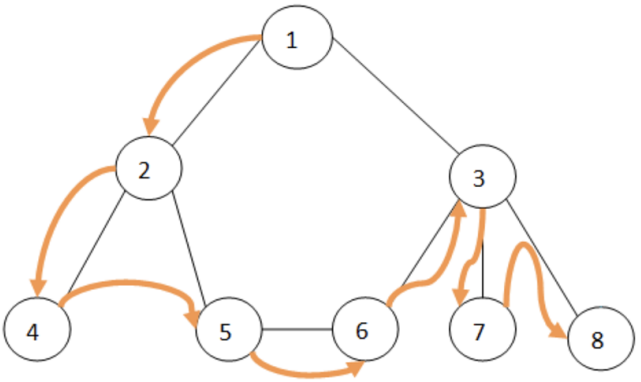
Sánchez Díaz Luis Gerardo-17460278

Fecha: 29/10/2021

Cuando no tenemos nada de información

Depth-First (Búsqueda en profundidad)

Una búsqueda en profundidad (DFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo. Su funcionamiento consiste en ir expandiendo cada uno de los nodos que va localizando, de forma recurrente (desde el nodo padre hacia el nodo hijo). Cuando ya no quedan más nodos que visitar en dicho camino, regresa al nodo predecesor, de modo que repite el mismo proceso con cada uno de los vecinos del nodo. Cabe resaltar que si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.



La búsqueda en profundidad se usa cuando queremos probar si una solución entre varias posibles cumple con ciertos requisitos; como sucede en el problema del camino que debe recorrer un caballo en un tablero de ajedrez para pasar por las 64 casillas del tablero.

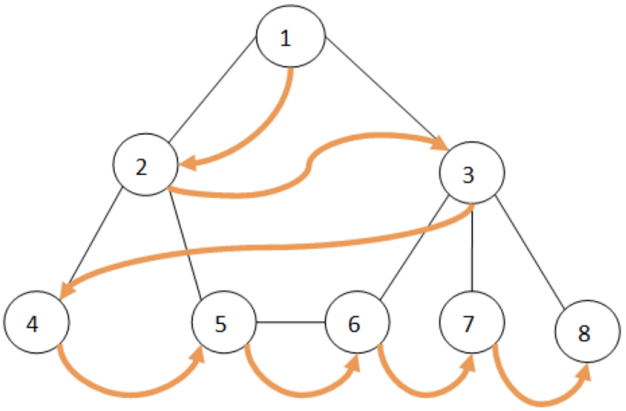
Explicación de depth first - backtracking (LIFO)

```
Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
  si el primer elemento es la meta
    entonces acaba
  si no
    elimina el primer elemento
    y añade sus sucesores al frente de la agenda
```

Breadth-First (Búsqueda en Anchura)

Una búsqueda en anchura (BFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el

caso de un grafo), para luego explorar todos los vecinos de este nodo. A continuación, para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el grafo.



Cabe resaltar que si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.La búsqueda por anchura se usa para aquellos algoritmos en donde resulta crítico elegir el mejor camino posible en cada momento del recorrido.

Explicación Breadth first

```
Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda esté vacía o se alcance la meta
  si el primer elemento es la meta
    entonces acaba
  sí no
    elimina el primer elemento y añade sus sucesores
al final de la agenda.
```

Cuando contamos con información

Hill climbing (Ascenso a la cima)

Hill climbing es una técnica que permite solo ir mejorando la solución por que aplica un mecanismo de restar o multistart tratando de mejorar la solución. En cada iteración, un nuevo punto es seleccionado de la vecindad del punto actual.
Un algoritmo de ascenso a colina comienza con una solución al problema a mano, normalmente elegida al azar:

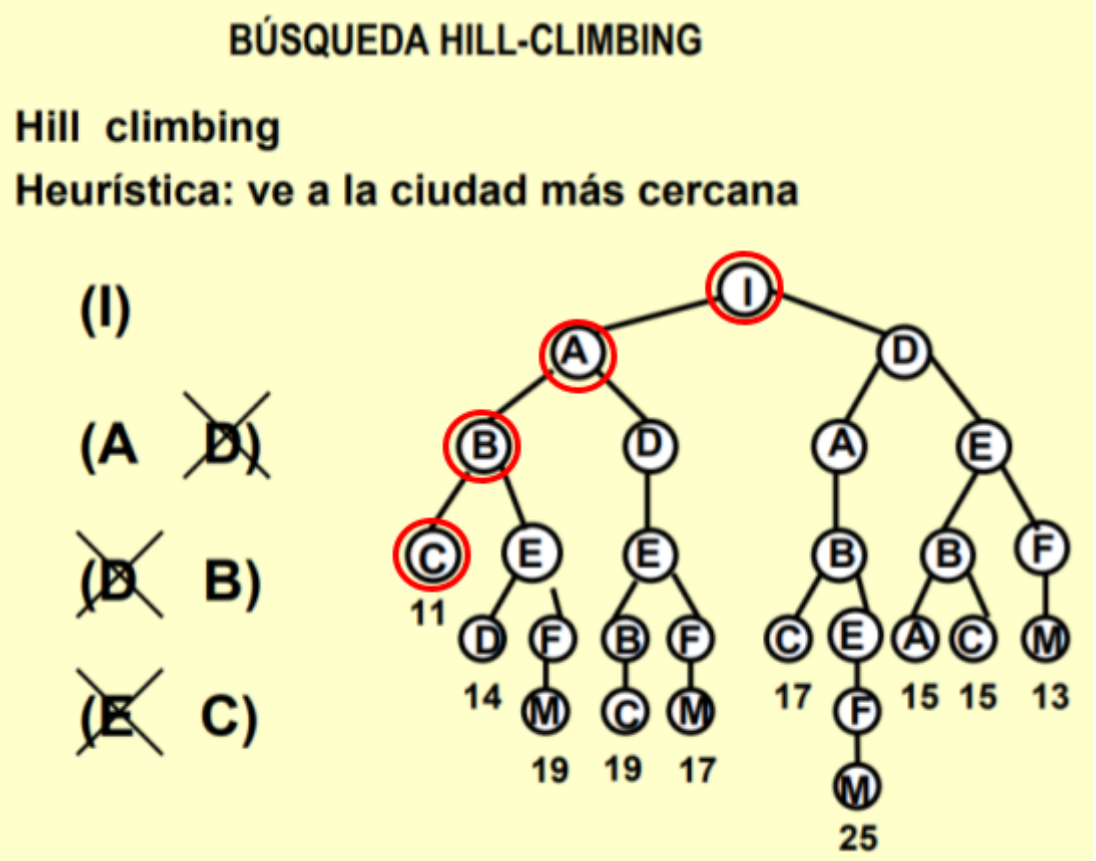
- Es una técnica que permite solo ir mejorando una solución.

- Luego, la cadena se muta, y si la mutación proporciona una solución con mayor aptitud que la solución anterior, se conserva la nueva solución; en caso contrario, se conserva la solución actual.
- Si el nuevo punto es mejor, se transforma en el punto actual, sino otro punto vecino es seleccionado y evaluado.
- El método termina cuando no hay mejoras, o cuando alcanza un número predefinido de iteraciones.

Explicación Hill-Climbing

```
evaluar el estado INICIAL
si es un estado objetivo
  entonces devolverlo y parar
si no ACTUAL := INICIAL
  mientras haya operadores aplicables a ACTUAL y no se haya
  encontrado solución
    seleccionar un operador no aplicado todavía a ACTUAL
    aplicar operador y generar NUEVOESTADO
    evaluar NUEVOESTADO
    si es un estado objetivo
      entonces devolverlo y parar
    sí no
      si NUEVOESTADO es mejor que ACTUAL
        entonces ACTUAL := NUEVOESTADO
      fin si
    fin si
  fin mientras
fin si
```

Ejemplo Hill-Climbing



Beam search (Recorte de Caminos)

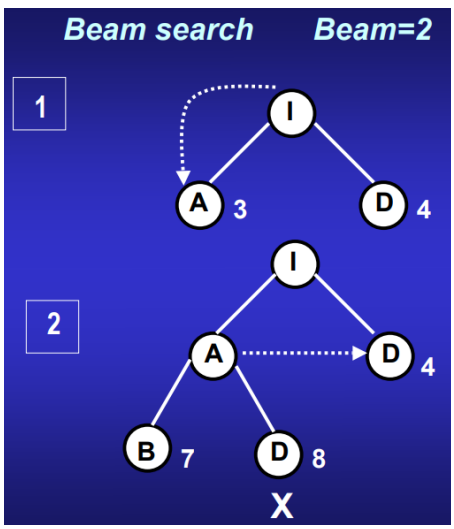
El algoritmo de Beam Search (BS) es un derivado de un branch-and-bound que utiliza la búsqueda de primero en anchura, e incorpora una heurística para escoger en cada nivel solo los mejores β nodos. Los mejores nodos se guardan en lo que llama el bear, y β suele denominarse bear width. Este método sacrifica la completitud a cambio de un enfoque heurístico muy efectivo.

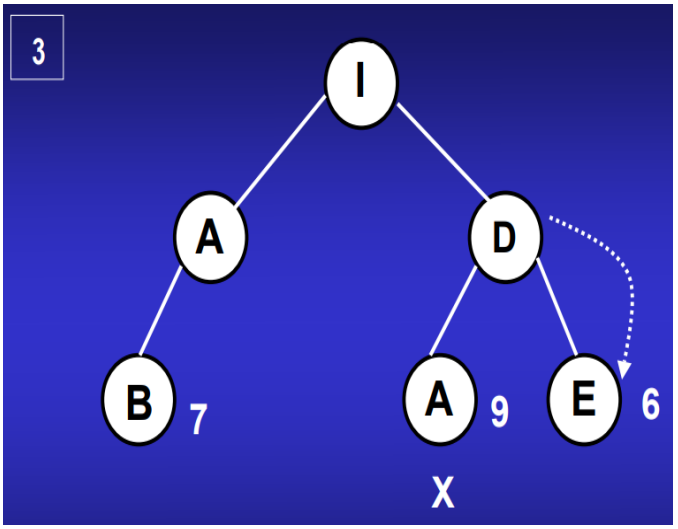
Los mejores β nodos son seleccionados por funciones de evaluación de un paso, las cuales estiman el costo de expandir la solución actual. Note se que los nodos en el vean representan soluciones parciales (evita el problema de la explosión combinatoria de búsqueda en amplitud por primera vez por ampliar solo los β nodos más prometedores en cada nivel. Una heurística se utiliza para predecir cuáles nodos son propensos a ser el más cerca a la meta o goal).

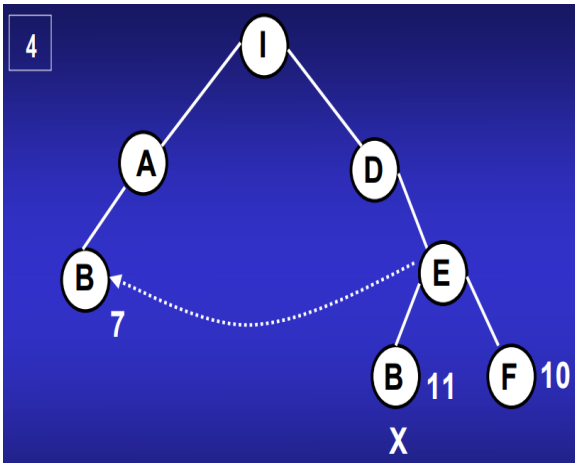
Explicación Beam search

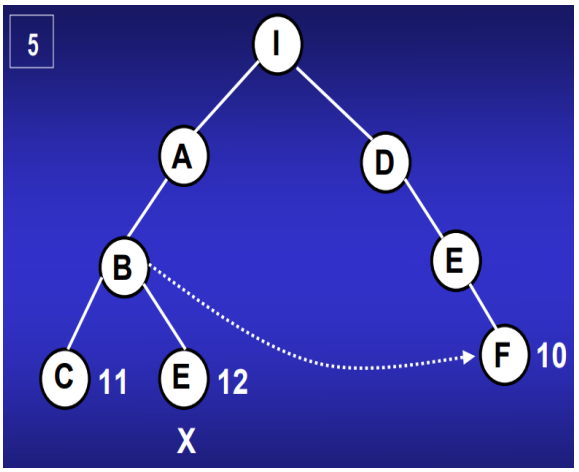
1. Formar una cola de un elemento Q que consiste en el nodo raíz
2. Hasta que Q está vacío o el objetivo se ha llegado a determinar si alguno de los elementos de Q es el objetivo.
    a. Si es así, no hacen nada
    b. Si no es así, quitar los elementos de Q y añadir a sus hijos,en su caso, la parte posterior de Q.
    c. Ordenar Q por heurística.
    d. Quite todos, pero los primeros p nodos de Q
3. Si el objetivo es alcanzado, éxito, si no fracaso

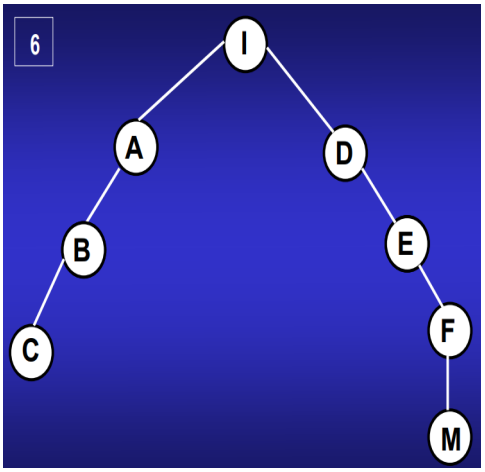
Ejemplo Beam search











Best first (Primero el mejor)

La búsqueda primero el mejor es un caso en el cual se selecciona un nodo para la expansión basada en una función de evaluación f(n). Esta función evaluación devuelve un número que sirve para representar lo deseable o indeseable que sería la expansión de un nodo. Se expande primero aquel nodo que tiene mejor evaluación. Se escoge el que parece ser el mejor pero puede no serlo.

Hay una familia entera de algoritmos de Búsqueda-Primero- Mejor con funciones de evaluación diferentes. Un componente clave de estos algoritmos es una función heurística, denotada h(n):

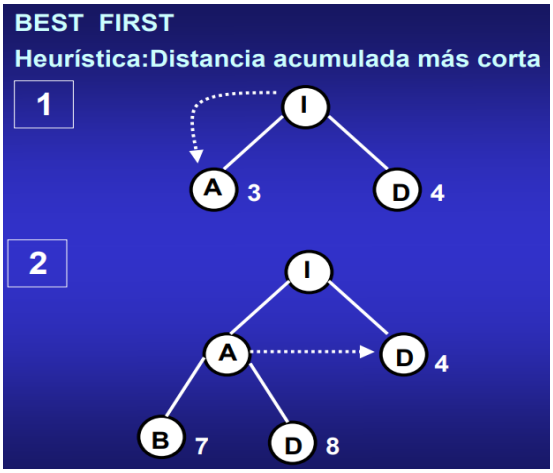
- h(n) = coste estimado del camino más barato desde el nodo “n” a un nodo objetivo.

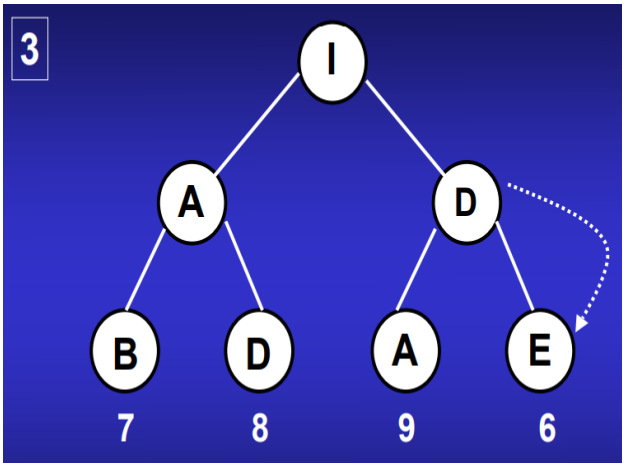
O en pocas palabras el algoritmo Best-first Search expande todos los nodos vecinos del nodo actual donde se encuentra el agente, buscando el siguiente nodo que tenga la distancia más corta hasta el nodo objetivo. Cada nodo vecino que es expandido se evalúa con la función heurística.

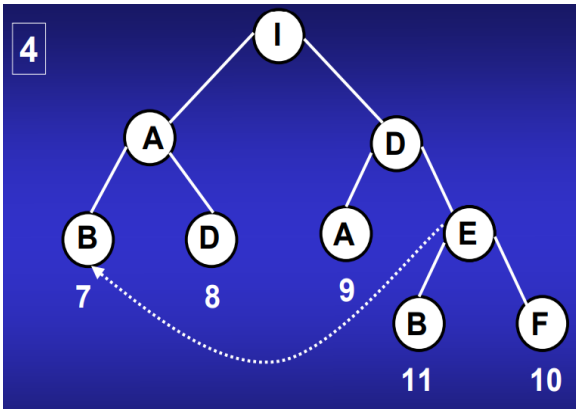
Explicación Best first

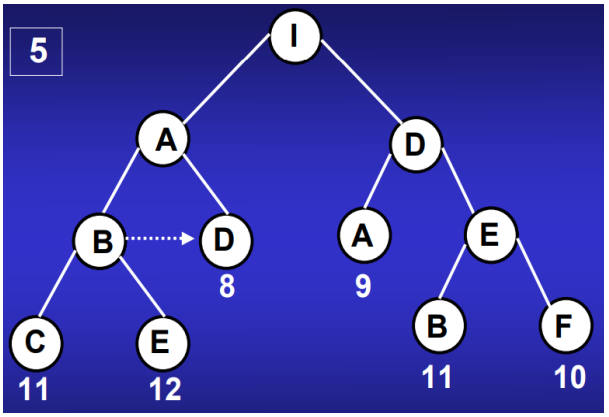
```
1. Sea L una Lista de nodos iniciales.
2. Sea N el nodo más cercano a la meta (el mejor).
    Si L está vacía, falla.
3. Si N es la meta. Regrese la trayectoria desde el nodo inicial
al nodo N.
4. Si N no es una meta. Buscar los hijos de N, colocarlos en L,
etiquetándolos con la trayectoria desde el nodo inicial.
    Retorne al paso 2.
```

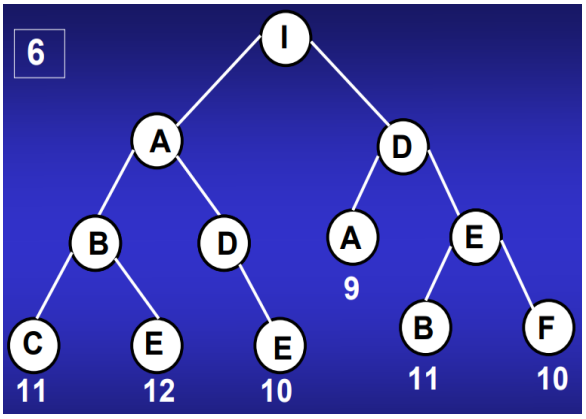
Ejemplo Best first

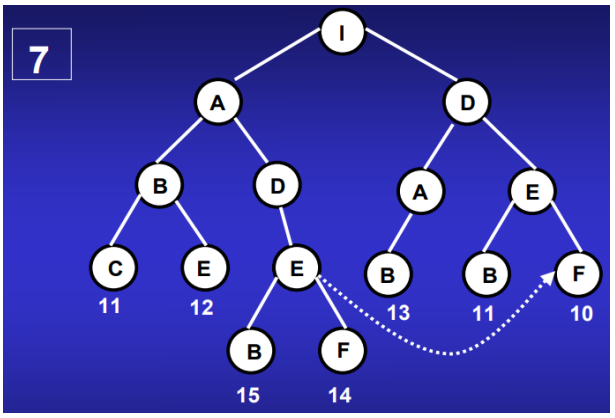


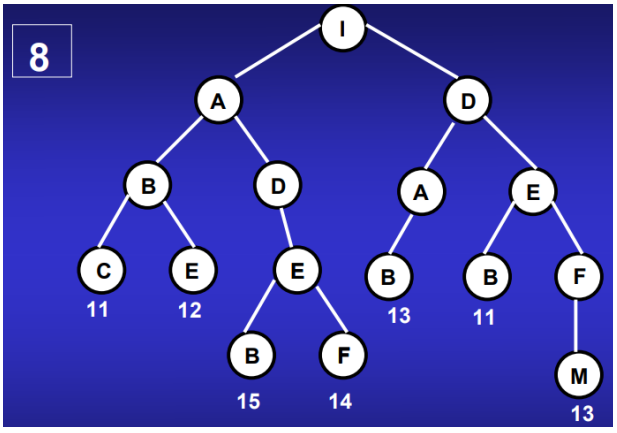












Cuando buscamos el camino óptimo

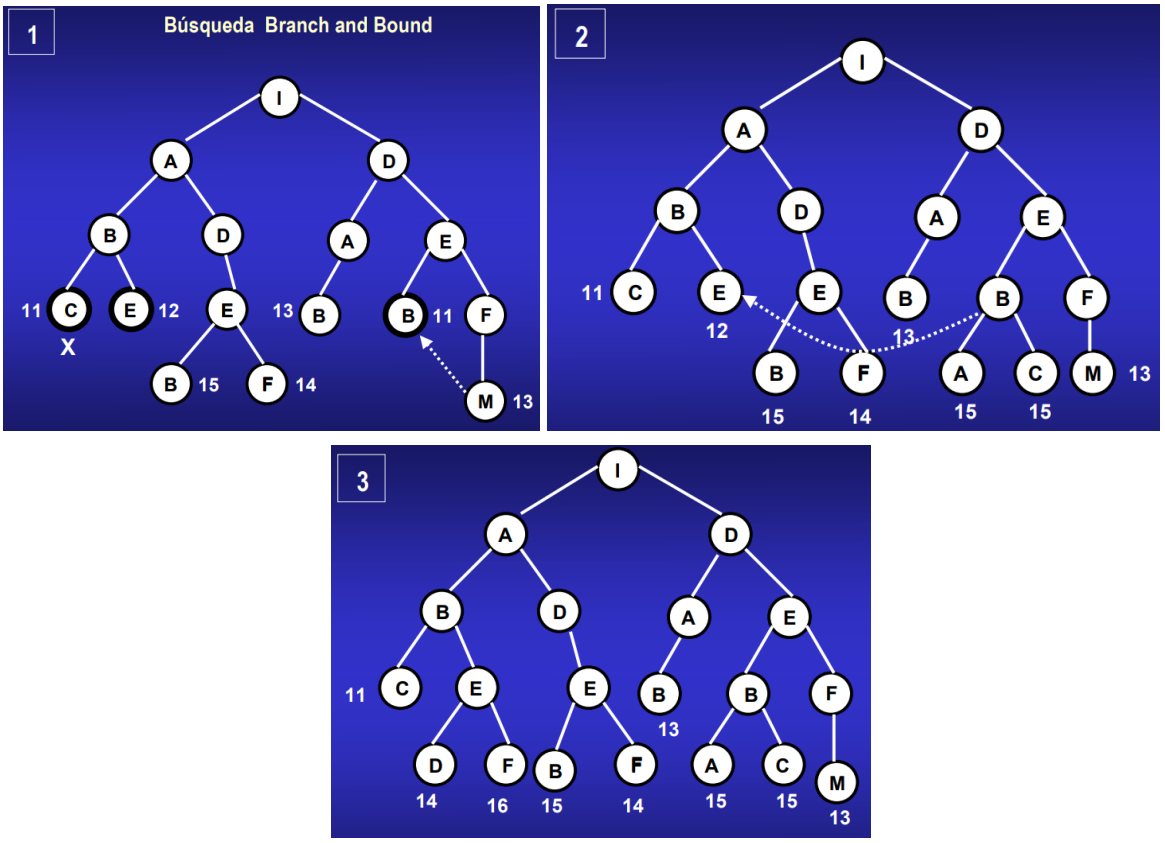
Branch and Bound (Ramificación y Acotamiento)

El método de Branch and Bound es un algoritmo diseñado para la resolución de modelos de Programación Entera. Su operatoria consiste en alinear el modelo de Programación Entera, es decir, resolver éste como si fuese un modelo de Programación Lineal y luego generar cotas en caso que al menos una variable de decisión (entera) adopte un valor fraccionario.
El algoritmo genera en forma recursiva cotas (o restricciones adicionales) que favorecen la obtención de valores enteros para las variables de decisión. En este contexto resolver el modelo lineal asociado a un modelo de Programación Entera se conoce frecuentemente como resolver la relajación continua del modelo entero.

Explicación Branch and Bound

```
Crea una agenda de un elemento (el nodo raíz)
  hasta que la agenda este vacía o se alcance la meta y los demás
  nodos sean de costos mayores o iguales a la meta
     si el primer elemento es la meta y los demás nodos son de
menor
    o igual costo a la meta
      entonces acaba
    sí no
         elimina el primer elemento y añade sus sucesores a la
agenda
 ordena todos los elementos de la agenda
```

Ejemplo Branch and Bound



A*

El algoritmo A* es un algoritmo de búsqueda que puede ser empleado para el cálculo de caminos mínimos en una red. Se va a tratar de un algoritmo heurístico, ya que una de sus principales características es que hará uso de una función de evaluación heurística, mediante la cual etiquetar los diferentes nodos de la red y que servirá para determinar la probabilidad de dichos nodos de pertenecer al camino óptimo.

Esta función de evaluación que etiquetar los nodos de la red estará compuesta a su vez por otras dos funciones. Una de ellas indicará la distancia actual desde el nodo origen hasta el nodo a etiquetar, y la otra expresa la distancia estimada desde este nodo a etiquetar hasta el nodo destino hasta el que se pretende encontrar un camino mínimo. Es decir, si se pretende encontrar el camino más corto desde el nodo origen s, hasta el nodo destino t, un nodo intermedio de la red n tendría la siguiente función de evaluación f(n) como etiqueta:

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

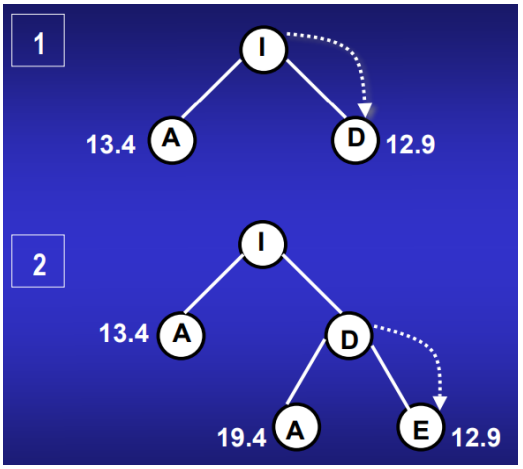
Donde:

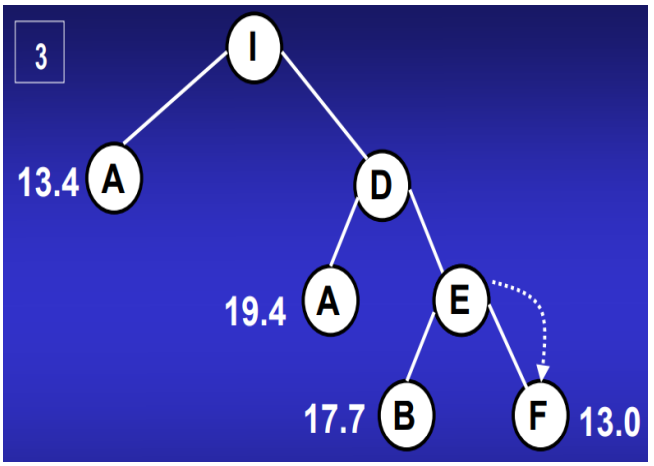
- g(n) indica la distancia del camino desde el nodo origen s al n.
- h(n) expresa la distancia estimada desde el nodo n hasta el nodo destino t.

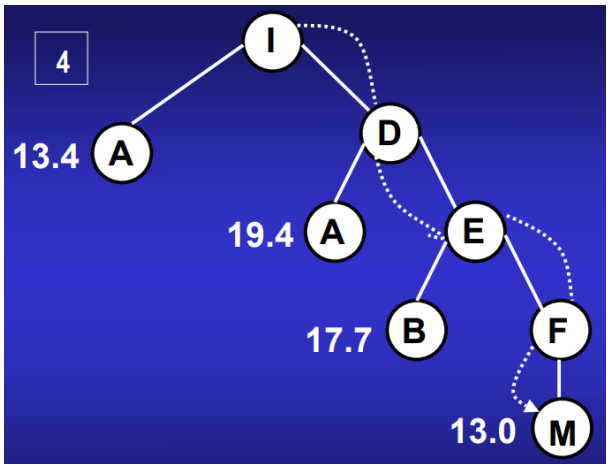
Explicación A*

```
Crea una agenda de un elemento (el nodo raíz)
  hasta que la agenda este vacía o se alcance la meta y los demás
  nodos sean de costos mayores o iguales a la meta
     si el primer elemento es la meta y los demás nodos son de
menor
    o igual costo a la meta
      entonces acaba
    si no
         elimina el primer elemento y añade sus sucesores a la
agenda ordena todos los elementos de la agenda de acuerdo al
costo acumulado más las subestimaciones de los que falta
```

Ejemplo A*







British museum (Museo Británico)

El algoritmo del Museo Británico se refiere a una técnica conceptual no a una técnica que pueda ser llevada a la práctica, pues el número de veces que habría que repetir el procedimiento sería enorme. En teoría, con tiempo suficiente, se lograría encontrar la solución óptima. Pero ese tiempo suficiente tendería a ser infinito, con lo que el algoritmo es intratable computacionalmente.

Se utiliza mucho para desechar soluciones costosas, se dice “esa solución es como aplicar el algoritmo del Museo Británico”, con eso está todo dicho, hay que buscar otra manera de hacer las cosas. La idea es que el algoritmo en cuestión es cómo sentar un grupo de monos delante de máquinas de escribir, si tienen todo el tiempo del mundo, llegarán a escribir todos los libros que se ven en aquella sala circular del Británico.