BÚSQUEDA HEURÍSTICA

- **"estudio de los métodos y reglas del descubrimiento y la invención".**

- **Búsqueda en e.e. --> reglas para elegir entre las ramas que con más probabilidad lleven a la solución.**

- **Situaciones**
  - ✓ **Problemas sin solución exacta**
  - ✓ **Problemas con soluciones exactas pero con elevados costes computacionales.**

- **Las heurísticas son falibles.**

  - ✓ **solución sub-óptima**
  - ✓ **fracasar en la búsqueda**

- **Ejemplo : tic-tac-toe**
  - ✓ **simetrías**
  - ✓ **heurística : mover a la casilla con mayor número de líneas ganadoras.**

- **Procedimiento de escalada**
  - ✓ **expansión del nodo actual en la búsqueda y evaluación de hijos**
  - ✓ **Se selecciona el mejor hijo, no se retienen ni los hermanos ni los padres.**
  - ✓ **la búsqueda se detiene cuando el estado es mejor que cualquiera de sus hijos.**
  - ✓ **heurísticas erróneas conducen a caminos infinitos**
  - ✓ **máximos locales.**
  - ✓ **perturbación aleatoria**

EL PRIMERO MEJOR

```
begin
open:=[Start]
closed:=[ ]
    while open≠ [ ]   do   ‘ mientras existan estados explorar
        begin
            eliminar el estado más a la izquierda de open, denominarlo X
            if X es una meta then return(el camino de Start a X)
                else begin .
                    generar los hijos de X;
                    for cada hijo de X do
                    case
                        el hijo no está en open o en closed
                            begin
                                asignar al hijo un valor heurística
                                añadir el hijo a open
                            end
                        el hijo ya está en open
                            if el hijo ha sido alcanzado por un camino más corto
                            then dar al estado en open el camino más corto
                        el hijo ya está en closed
                            if el hijo ha sido alcanzado por un camino más corto then
                                begin
                                    eliminar el estado de closed
                                    añadir el hijo a open
                                end
                    esac
                poner X en closed
                reordenar los estados de open por mérito heurístico ( el mejor a la izda)
        end
    devolver FALLO
end.
```

IMPLANTACIÓN DE FUNCIONES DE EVALUACIÓN HEURÍSTICAS

- **El algoritmo puede perderse en profundidad si la heurística no es buena.**

- **Si dos estados tienen la misma evaluación heurística es mejor examinar el estado que está más cerca del estado raíz del grafo.**

- **Se introduce un contador de profundidad para cada estado : g(n).**

- **La función de evaluación dos componentes : f(n) = g(n) + h(n)**
  - ✓ **n = estado**
  - ✓ **g(n) = longitud real del camino desde el estado n a start**
  - ✓ **h(n) = estimador heurístico de la distancia de n a la meta.**

- **El primero mejor + f = formulación general de la búsqueda heurística**

1. **Las operaciones sobre los estados generan los hijos del estado actual**

2. **Comprobar ocurrencias previas del estado para evitar bucles**

3. **Asignar a cada estado el valor f(n)=g(n)+h(n)**

4. **Clasificar los estados en open en virtud de sus valores de f**

- **Heurísticas que localizan el camino más corto se dice que son ADMISIBLES.**

- **Considérese la función de evaluación f(n) = g(n) + h(n) donde:**
  - **n = es cualquier estado localizado en la búsqueda.**
  - **g(n) = es el coste de n desde el estado inicial.**
  - **h(n) = estimador heurístico del coste de ir desde n hasta la solución.**

  Si la función de evaluación se utiliza con el algorítmo de búsqueda el primero mejor, el resultado se denomina ALGORITMO A.

- **Un algoritmo de búsqueda es ADMISIBLE si, para cualquier grafo, siempre termina en el camino solución óptimo cuando el camino desde el estado inicial hasta la solución existe.**

- **$f^*(n) = g^*(n) + h^*(n)$ donde**
  - **$g^*(n)$ = es el coste del camino más corto desde el nodo inicial hasta n.**
  - **$h^*(n)$ = coste real del camino más corto desde n hasta la meta.**

  $f^*(n)$ es el coste real del camino óptimo desde el nodo inicial hasta la meta que pasa a través de n.

- **Objetivo : f un estimador próximo a f^***

- **En un algoritmo A : $g(n) \geq g^*(n)$**

- **Si el algoritmo A se utiliza con una función de evaluación en la que h(n) es menor o igual que el coste del camino mínimo desde n hasta la meta ($h(n) \leq h^*(n)$), el algoritmo de búsqueda resultante se denomina A*.**

- **El algoritmo el primero mejor termina siempre para grafos finitos.**

- **Si existe un camino desde start hasta un nodo objetivo, A* termina.**

- **En cualquier instante de la ejecución de A*, antes de su finalización, existe en open un nodo n' que está en un camino óptimo desde start a un nodo objetivo, para el que $f(n') \leq f^*(start)$.**

- **Todos los algoritmos A* son admisibles.**

- **Una función heurística h es monótona si :**
  1. **Para todos los estados n_i y n_j donde n_j es descendiente de n_i,**
     $$h(n_i) - h(n_j) \leq coste(n_i,n_j)$$
     $coste(n_i,n_j)$ es el coste del arco que va desde el estado n_i hasta el estado n_j
  2. **La evaluación heurística del estado meta es cero, h(meta)=0.**

- **Si se satisface la restricción de monotonía, A* ha encontrado ya un camino óptimo para cualquier nodo que selecciona para su expansión. Esto es, si A* selecciona el nodo n para su expansión y se satisface la restricción de monotonía, $g(n)=g^*(n)$.**

- **Si se satisface la restricción de monotonía, la secuencia de los valores de f de los nodos expandidos sucesivamente por A* es no decreciente.**

- **Para dos heurísticas A*, h_1 y h_2, si $h_1(n) \leq h_2(n)$, para todos los estados n, en el espacio de búsqueda, la heurística h_2 se dice que es más informativa que h_1.**