Pràctica 1:
"RESOLUCIÓ DE PROBLEMES MITJANÇANT CERCA EN UN ESPAI D'ESTATS"

Sistemes Intel.ligents, 3A1 GII –ETSINF-

Objetiu de la pràctica

- *Avaluar l'eficiència, cost temporal i espacial de diferents estratègies de cerca aplicades al joc del 8-puzle.*

- Per a això es proporciona el programa puzle implementat en Python.

Sesions

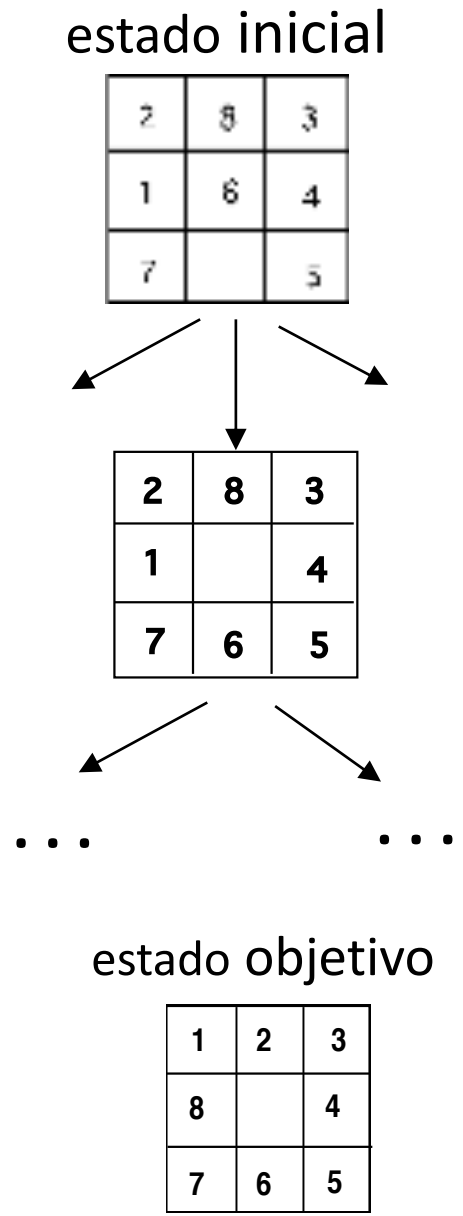
S1: 27/9
S2: 4/10
S3: 11/10
S4: 18/10
S5: 25/10 Examen Pràctica

8-puzzle

- Tauler de 3*3 caselles.
- 8 de les caselles contenen una peça o fitxa que es pot lliscar al llarg del tauler horitzontal i verticalment.
- Les fitxes venen marcades amb els números de l'1 al 8
- Hi ha una casella lliure en el tauler que permet els moviments de les fitxes.

8-puzzle

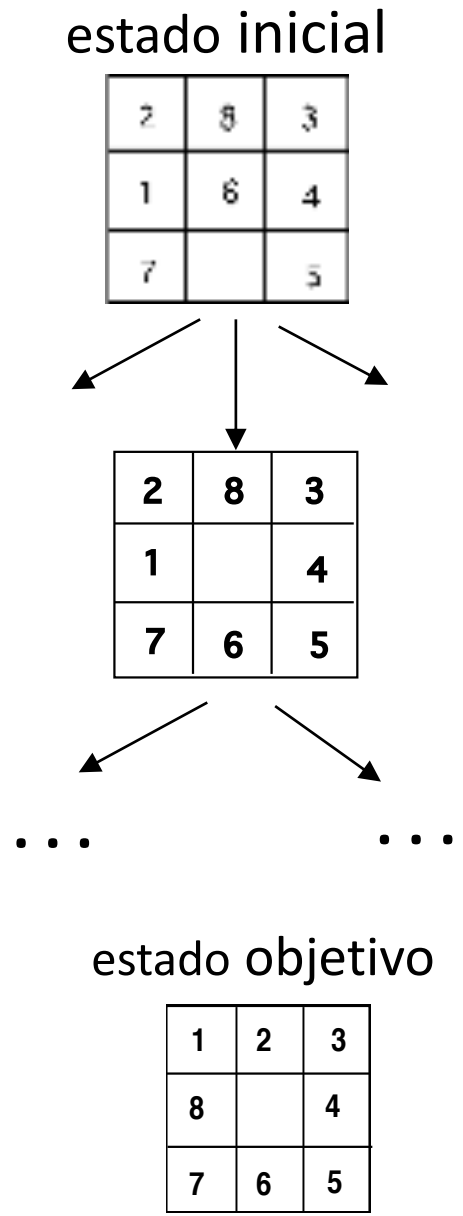
- Accions: moviments de les fitxes (o moviments de casella lliure) en 4 direccions:
  - a dalt, a baix, dreta, esquerra
- Funció de cost de les accions: cost= 1 per a totes les accions del problema
- Solució: Seqüència de moviments de les fitxes des de l'estat inicial a l'estat objectiu.
- Cost del camí: **nombre de moviments del camí**.



8-puzzle

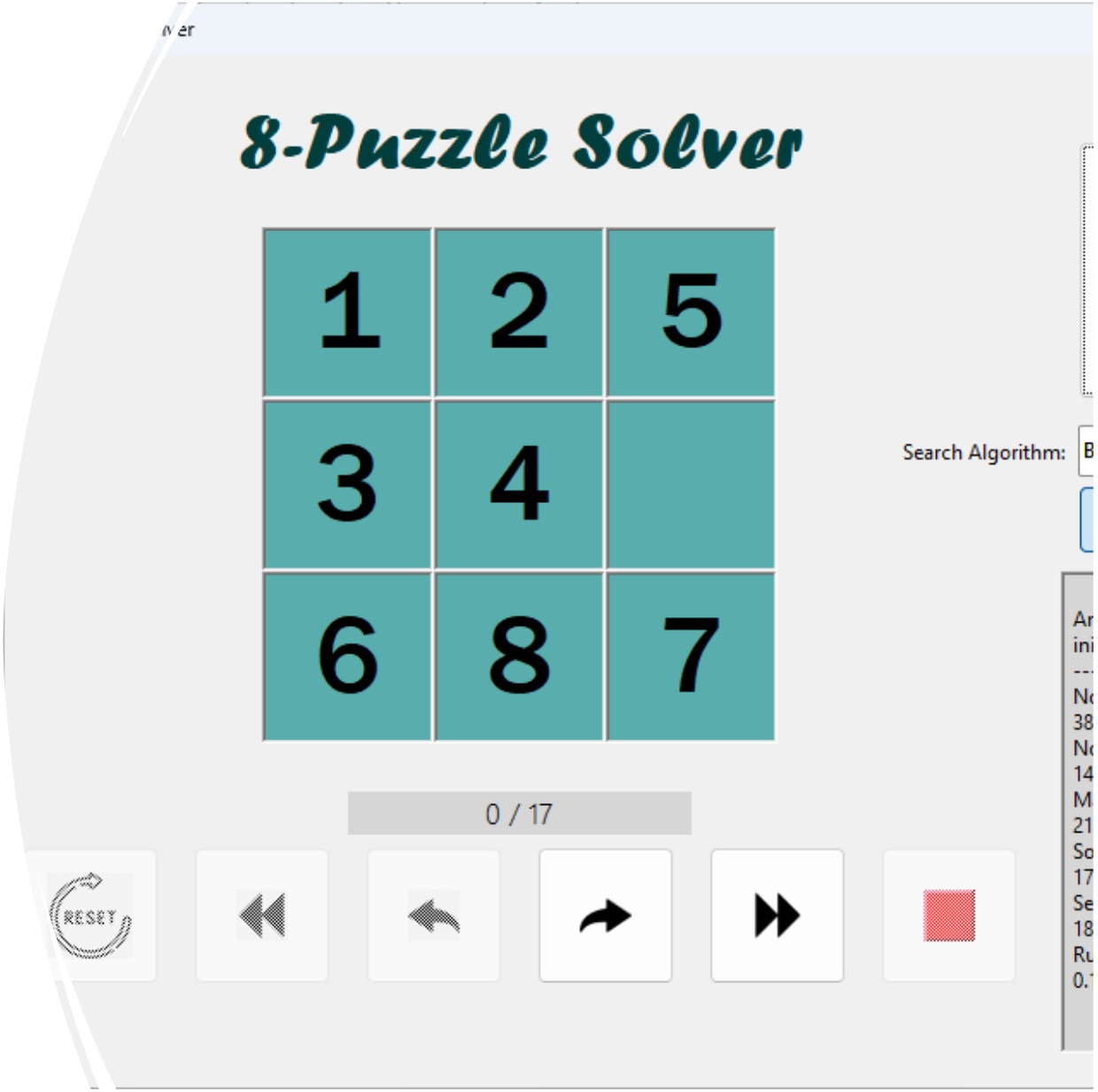
- Representació com a lista d'un estat:
  - {C1, C2, C3, …. C8}
  - La casella lliure es representa amb un 0

- Estat objetiu: {1,2,3,8,0,4,7,6,5}



8-puzzle Solver

- Programa en Python que resol el problema del 8-puzzle amb diferents estratègies
- Es poden programar noves estratègies
- Disponible a Poliformat



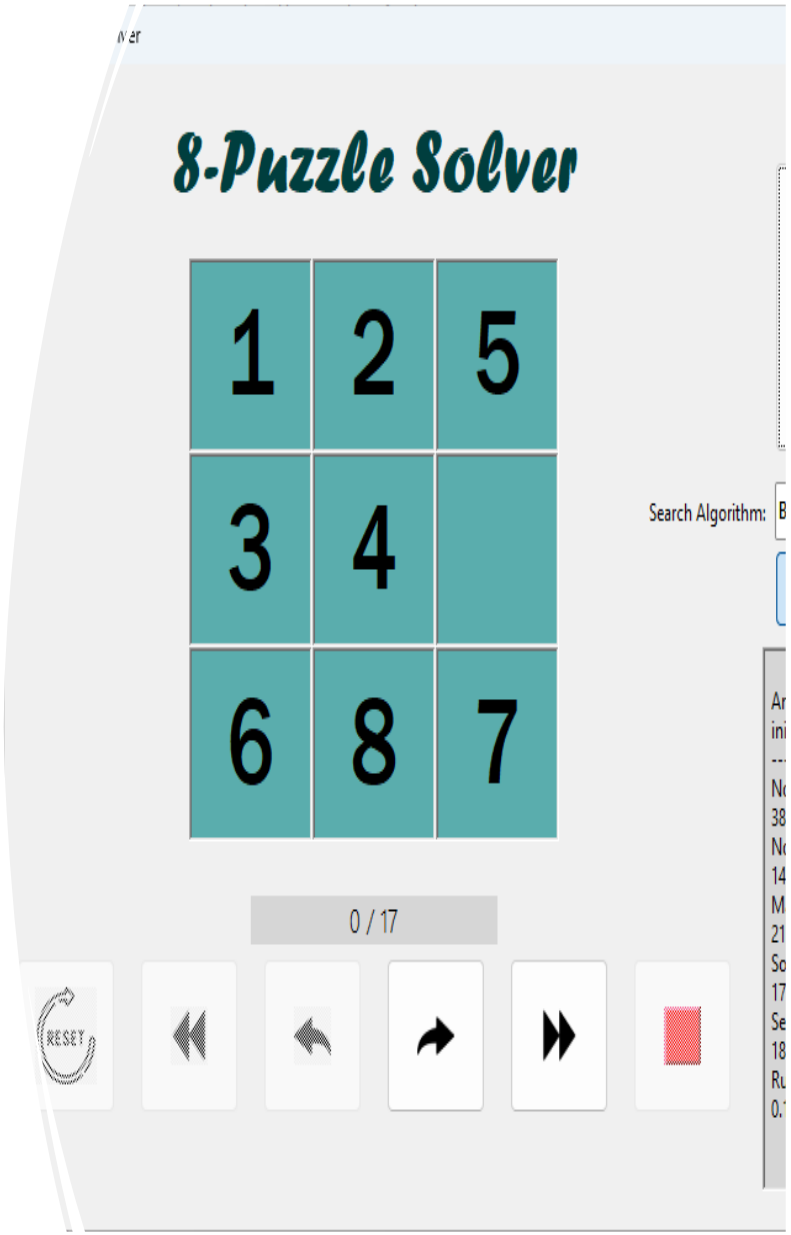
8-puzzle Solver

Instal·lació:

- Copia el fitxer puzzle.zip que es troba a PoliformaT, a la carpeta Recursos/Pràctica 1 del teu compte.
- Descomprimeix el fitxer. En descomprimir-lo es crearà un directori anomenat 'puzzle' on es troben els fitxers font.

Execució:

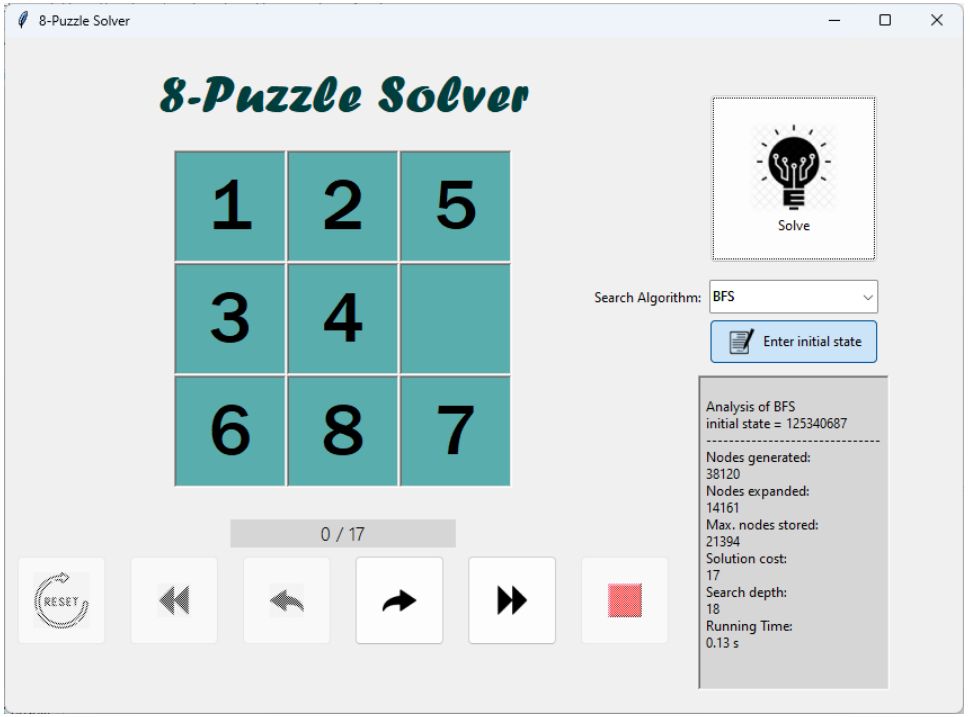
- Des d'un entorn de desenvolupament de Python (per exemple, spyder, disponible als laboratoris).
- Des d'una terminal (si tens Python instal·lat al PATH), executa el programa 'interface.py'.



8-puzzle Solver

Probes:

- Prem el botó '**Enter initial state**'
- Insertar el fet inicial : L'eixemple sería 125340687
- Seleccionar l'estratègia de resolució desitjada en la pestanya desplegable 'Search Algorithm'
  - Si se selecciona una estratègia que requereix un nivell de tall de profunditat s'obrirà una finestra
  - on es podrà introduir aquest valor.
- Prem el botó '**Solve'.**
  - Es mostraran les dades de l'execució del procés de cerca realitzat
- Utilitza els botons que es troben sota la finestra del puzle per recórrer el camí de la solució.



8-puzzle Solver (estratègies implementades)

- **BFS.** Aquesta és l'estratègia **d'Amplària** que utilitza f(n)=g(n) per a l'expansió de nodes, on g(n) és el factor cost associat al nivell de profunditat del node n en l'arbre de cerca.
- **DFS (Graph Search)**. Aquesta estratègia implementa una cerca en **Profunditat** GRAPH-SEARCH. Per a això, utilitza la funció f(n)=-g(n), on g(n) és el factor cost associat al nivell de profunditat del node n en l'arbre de cerca.
- **DFS (Bactracking)**. Aquesta estratègia implementa una cerca en **Profunditat amb backtracking** o Profunditat TREE-SEARCH. Per a això, utilitza la funció f(n)=-g(n), on g(n) és el factor cost associat al nivell de profunditat del node n en l'arbre de cerca. Només manté en memòria els nodes explorats que pertanyen al camí actual (llista PATH).
- **ID**. Aquesta estratègia implementa una cerca per Aprofundiment Iteratiu (Iterative Deepening). Es tracta de realitzar successives cerques en profunditat amb backtracking fins que s'aconsegueix la solució. Cada nova cerca incrementa el nivell de profunditat de l'exploració en 1.

8-puzzle Solver (estratègies implementades)

- **Voraz (Manhattan).** Aquesta és una estratègia que implementa un Algorisme voraç seguint la funció f(n)=D(n), on D(n) és la distància de Manhattan.
- **A* Manhattan**. Aquesta és una cerca de tipus A que utilitza la distància de Manhattan com a heurística; l'expansió dels nodes es realitza seguint la funció f(n)=g(n)+D(n), on g(n) és el factor cost corresponent a la profunditat del node n en l'arbre de cerca i h(n)=D(n) és la distància de Manhattan.
- **A* Euclidean**: Aquesta és una cerca de tipus A que utilitza la distància Euclidiana com a funció heurística; l'expansió dels nodes es realitza seguint la funció f(n)=g(n)+D(n), on g(n) és el factor cost corresponent a la profunditat del node n en l'arbre de cerca i h(n)=D(n) és la distància Euclidiana.
- **IDA* Manhattan.** Aquesta estratègia de cerca implementa una cerca IDA* on el límit de la cerca ve determinat pel valor-f, és a dir, per la funció f(n)=g(n)+D(n), on g(n) és el factor de cost i h(n)=D(n) és la distància de Manhattan. El límit d'una iteració i de l'algorisme IDA* és el valor-f més xicotet de qualsevol node que haja excedit el límit en la iteració anterior i-1.
- Es poden implementar mes ….

Sesions 2, 3 i 4

1º. Implementar tres noves funcions heurístiques

→ Modificar el codi del puzle

2º. Comparar totes les heurístiques

Com crear heurístiques?

Fitxer **interface.py**: Part de la interfície i invocacions als algorismes de cerca.

Fitxer **main.py**: Implementació dels algorismes i estratègies de cerca.

Fitxer 'interface.py'

- Afegir el nom de la nova funció de cerca perquè aparega en la llista desplegable d'estratègies de cerca

Funció `def __init__(self, master=None):` Línies 88-89 (aprox.):

```
self.algorithmbox.configure(cursor="hand2", state="readonly",
                            values=('BFS',    'DFS    (Graph    Search)',    'DFS
                           (Backtracking)',   'Voraç   (Manhattan)',   'ID',   'A*
                           Manhattan', 'A* Euclidean', 'IDA* Manhattan'))
```

Com crear heurístiques?

Fitxer 'interface.py'

- Si l'estratègia requereix una profunditat màxima

Funció `selectAlgorithm`, línia 248 (aprox):

```
if algorithm in ['DFS (Graph Search)', 'DFS (Backtracking)']:
 cutDepth = int(simpledialog.askstring('Cut depth', 'Please, enter your max
            depth'))
```

Com crear heurístiques?

Fitxer 'interface.py'

- Realitzar la cridada a la funció de cerca. Exemple de A* (línia 381 aprox.)

```
elif str(algorithm) == 'A* Manhattan':
     main.graphSearch(initialState,main.function_1,main.getManhattanDistance)
     path, cost, counter, depth, runtime, nodes, max_stored,memory_rep = \
         main.graphf_path, main.graphf_cost, main.graphf_counter,
         main.graphf_depth,          main.time_graphf,          main.node_counter,
main.max_counter,
         main.max_rev_counter
```

`function_1`. Retorna 1, representa el cost habitual d'un moviment en el puzle
`function_0`. Retorna 0, s'usa per a l'estratègia voraç
`function_N`. Retorna -1. Usat en estratègies de profunditat

cridada a la funció específica que calcula h(n) segons

Com crear heurístiques?

Fitxer 'main.py'

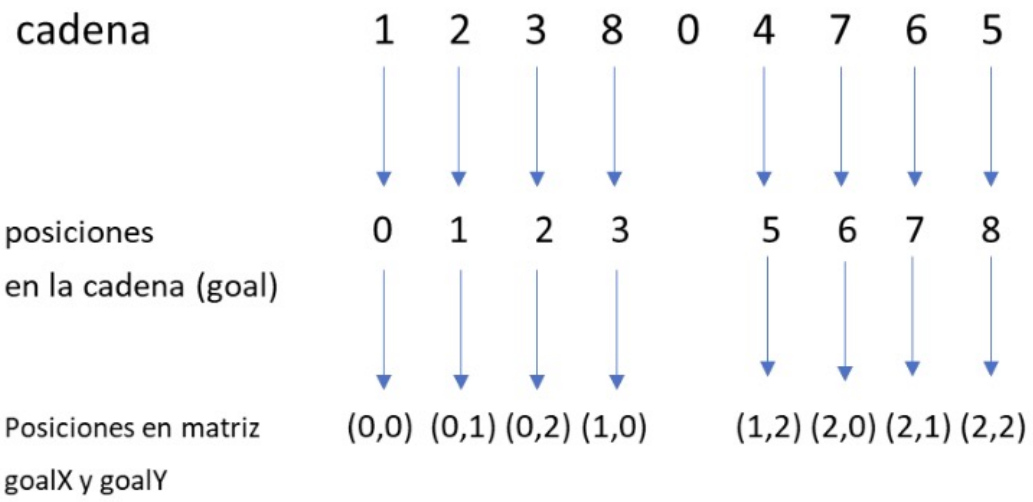
• Cal incloure una funció que implemente la nova funció heurística. Exemple "Distancias de Manhattan"

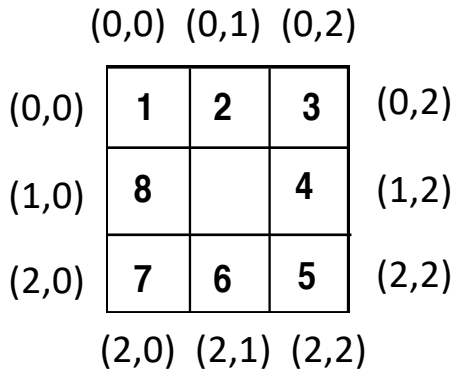
```
def getManhattanDistance(state):   #state és l'estat actual a avaluar (en forma de
cadena text)
  tot = 0
  for i in range(1,9):                    #recorre les 8 peces
    goal = end_state.index(str(i))  #posició de la peça i en l'estat final (end_state)
    goalX = int(goal/ 3)              #Passem l'end state de cadena a una matriu de 3x3
    goalY = goal % 3              #ocuparia aqueixa peça (golX) i que columna (goalY)
    idx = state.index(str(i))     #mateixa operació però per a la fila (idx) i columna (idy)
    itemX = int(idx / 3)              #que ocupa aqueixa peça en l'estat actual
    itemY = idx % 3
    tot += (abs(goalX - itemX) + abs(goalY - itemY)) #Calcul de distàncies
  return tot
```

Com crear heurístiques?

Fitxer 'main.py'

- Transformació de representació lineal (cadena) a matriu. Exemple estat objetiu.





Treball a realitzar

Implementar tres funcions heurístiques i respondre a una sèrie de preguntes.

- Entrega en tarea Poliformat el dia de l'examen (fitxer Python i text de resposta a les preguntes)

1. Implementar la funció heurística **PECES DESCOL·LOCADES** que s'ha vist en classe de teoria.

2. Implementar la funció heurística **SEQÜÈNCIES**

3. Implementar la funció heurística **FILES_COLUMNES**

Treball a realitzar

Seqüències. $f(n)=g(n)+3*S(n)$, on $g(n)$ és el factor cost associat al nivell de profunditat del node n en l'arbre de cerca i $h(n)=3*S(n)$.

$S(n)$ és la seqüència obtinguda recorrent successivament les caselles del puzle, excepte la casella central, i anotant 2 punts en el compte per cada fitxa no seguida per la seua fitxa successora i 0 punts per a les altres; si hi ha una fitxa en el centre, s'anota 1. Exemple:

2	8	3
1	6	4
7		5

Per a la fitxa 2:	Sumar 2 (perquè no va seguida de la seua fitxa successora 3)
Per a la fitxa 8:	Sumar 2 (perquè no va seguida de la seua fitxa successora 1)
Per a la fitxa 3:	Sumar 0 (perquè sí que va seguida de la seua fitxa successora 4)
Per a la fitxa 4:	Sumar 0 (perquè sí que va seguida de la seua fitxa successora 5)
Per a la fitxa 5:	Sumar 2 (perquè no va seguida de la seua fitxa successora 6)
Per a la fitxa 0:	Sumar 2 (se sumarà 2 punts sempre que no estiga en la casella central)
Per a la fitxa 7:	Sumar 2 (perquè no va seguida de la seua fitxa successora 8)
Per a la fitxa 1:	Sumar 0 (perquè va seguida per la seua fitxa successora 2)
Per a la fitxa 6:	Sumar 1 (perquè és la fitxa central).
TOTAL	**11 punts**

Treball a realitzar

Files_Columnes. h(n)=FilCol(n).

On FilCol(n) es calcula de la següent forma: per a cada fitxa del tauler, si la fitxa no està en la seua fila destí correcta se suma 1 punt; si la fitxa no està en la seua columna destí correcta se suma 1 punt. Per tant, el mínim valor d'aquesta heurística per a una fitxa és 0 (quan la fitxa està col·locada correctament en la seua posició objectiu), i el màxim valor és 2 (quan ni la fila ni la columna de la posició de la fitxa són iguals a valor de la fila i columna de la posició objectiu).
Exemple:

2	8	3
1	6	4
7		5

La fitxa 1 està a la columna correcta però no a la fila: sumaria 1
La fitxa 2 està a la fila correcta però no a la columna: sumaria 1
Les fitxes 3, 4 i 5 sumen 0
La fitxa 6 està a la columna correcta però no a la fila: sumaria 1
La fitxa 7 suma 0
La fitxa 8 no està ni a la fila ni a la columna correcta: suma 2

Treball a realitzar (resum)

1. **Implementar** les tres funcions heurístiques (Descol·locades, Seqüències, Files_Columnes).
2. **Executar** diverses configuracions del trencaclosques i anotar els resultats de totes les estratègies de cerca que es mostren al programa, a més de les tres noves funcions heurístiques implementades.
   - Com a exemple, es pot executar les configuracions proposades (es recomana executar més configuracions per analitzar detalladament el comportament de les estratègies).
   - **Omplir** taules.

	BFS	DFS (GS)	DFS (Back)	Voraz	ID	A* Manh	A* Euclid	IDA* Manh	Desc	Sec	FilCol
Nodes generated											
Nodes expanded											
Max nodes stored											
Solution cost											
Search depth											
Running time											

3. **Contestar** de forma razonada les preguntes del butlletí.

Treball per a hui (I ja serveix per a la pràctica)

- Executa les següents configuracions amb els algoritmes de cerca no informada i omple les taules amb els resultats al butlletí.

	2	3
8	4	5
7	1	6

7	8	1
4		6
2	3	5

Treball per a hui (I ja serveix per a la pràctica)

- Executa les següents configuracions amb els algoritmes de cerca (excepte els 3 últims) i omple les taules amb els resultats al butlletí.

	BFS	DFS (GS)	DFS (Back)	Voraz	ID	A* Manh	A* Euclid	IDA* Manh	Desc	Sec	FilCol
Nodes generated											
Nodes expanded											
Max nodes stored											
Solution cost											
Search depth											
Running time											