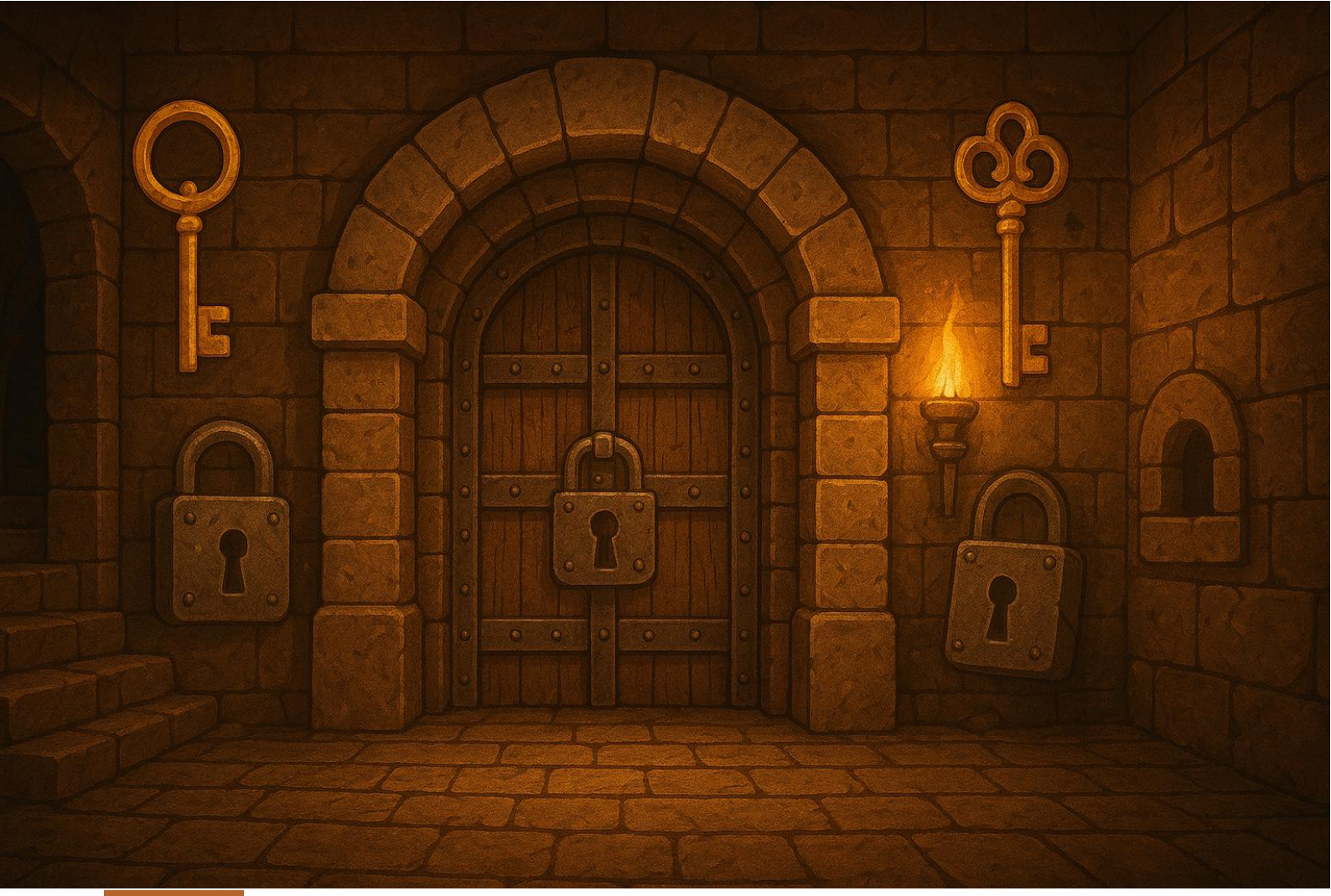


Locks & Keys

PROP Q1 25/26: Primera activitat

STRIPS, PDDL, cerca cega i informada

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Campus de Vilanova i la Geltrú

ENUNCIAT

Es vol desenvolupar un sistema de cerca per decidir les accions que han de prendre un conjunt d'agents dins d'un laberint per trobar el camí a la sortida. Els agents comencen a una posició inicial preestablerta i han d'aconseguir de forma cooperativa que qualsevol dels agents arribi a la sortida. Segons el disseny del laberint pot ser necessari travessar portes, cosa que només podrem fer si recollim abans les claus corresponents.

El projecte és farà en equips de dos alumnes.

Regles

- **Moviments:** els agents es podran moure exclusivament en les 4 direccions amunt, avall dreta i esquerra, evitant les parets.
- **Sempre es podrà moure pels espais buits, sobre les claus, i la sortida.**
- **Travessar portes:**
  - Només podem estar a la casella que té una porta si tenim la clau corresponent. Altrament no podrem ocupar la cel·la de la porta.
- **Recollida de les claus:**
  - N'hi ha prou en ocupar la cel·la de la clau per agafar-la.
- Parelles clau-porta: cada clau obre un model de porta. La notació de les claus és {a,b,c,...}, i cada porta pot obrir la porta corresponent {A,B,C,....}
- **Compartició de Claus:**
  - Les Claus que recullen els agents són immediatament compartides per tots els altres. Per exemple, si un agent recull la clau a, tots els altres agents poden obrir la porta A.
- **Model temporal**
  - Només un agent pot moure en cada pas (cost = 1 per acció d'un agent)
- **Col·lisions**
  - Està prohibit que dos agents ocupin la mateixa cel·la.
- **Objectiu**
  - Un agent qualsevol arriba a la casella de sortida.

Notació utilitzada pels mapes del laberint

Els laberints es descriuran en arxius de text UTF-8 usant els següents caràcters. Teniu els arxius de mapes d'exemple a la carpeta **src/main/resources**.

Agents:	{1, 2,...},
Claus:	{a,b,..},
Porta:	{A,B,...}, on *a* obre *A*, *b* obre *B*.....
Sortida:	@ (arrova),
Paret:	# (Sharp)
Espai:	· (punt volat)

Es mostren a continuació dos dels mapes per la vostra referència visual:

Mapa A) fàcil, solució òptima en 5 moviments.	**Mapa C)** mig, solució òptima en 37 moviments.
######## #·2···## #a···#@# #···1·A# ########	############## #·····a·#·# #1··········# #··########## ##··A·B··C@## ##·########c# #b··###·#···# #·····2····# ##############

EXERCICIS

PRIMERA PART: Descripció en PDDL [2p]

Descriviu el domini i el problema bàsic (Mapa A) usant sintaxi PDDL. Anomeneu als arxius **domain.pdd** i **map.pddl**

> **Important**: definiu el domini de tal forma que l'acció d'agafar la clau ja vagi conjunta amb el moviment. És a dir, per agafar una clau no cal moure's a sobre i agafar-la (2 passos), sinó que amb un sol pas ja l'agafem. Haureu de fer dos accions diferenciades de moviment per aconseguir-ho.

Proveu el vostre PDDL a https://editor.planning.domains , useu el Solver **LAMA-first planner** per trobar la solució.

SEGONA PART: Implementació Java [4p]

A partir del projecte base que es proporciona, se us demana:

- Acabar la implementació de la classe ***Mapa***, que representa l'estat del joc i coneix les regles de moviment per determinar moviments futurs. Heu d'implementar tots els mètodes marcats amb **@TODO**.
- Implementar els mètodes de cerca BFS, DFS, IDS i A* completant el codi a les classes corresponents. (***CercaBFS, CercaDFS, CercaIDS, CercaAstar***). Es valorarà que les cerques reutilitzin codi si és possible (p.ex. usant una superclasse base o mètodes compartits). Podeu crear noves classes si ho considereu oportú.

L'heurística que usareu per la A* és la distancia a la següent meta "obligatòria": Mentre hi hagi claus pendents de recollir, prendrem la distancia de Manhattan a la clau més propera als agents. Quan totes les claus estan recollides, es dona la mínima distancia de Manhattan dels agents a la sortida. En ambdós casos, ignorarem els obstacles per computar la distancia (parets, agents i portes).

Cal que implementeu l'heurística en una classe ***HeuristicaBasica*** i li passeu al constructor de ***CercaAStar*** per fer-la servir. Així podem provar diferents heurístiques amb el mateix algorisme.

MOLT IMPORTANT: DFS sucks

Limiteu la cerca DFS a profunditat 50 per evitar caure en el pou de la perdició.

MOLT IMPORTANT: Control de cicles

Via un paràmetre extra (*boolean usarLNT*), se'ns indica quina gestió de cicles hem de fer. Hi ha dues opcions:

- **Control dins de la branca actual (***usarLNT=false***).** Controleu si l'estat actual està repetit dins del camí/branca actual (heu de recórrer recursivament l'arbre cap amunt buscant l'estat). Aquest sistema té com avantatge el consum mínim de memòria.
- **Control de cicles per LNT(***usarLNT=true***).** Habiliteu l'opció d'usar una Llista de Nodes Tancats (LNT) per registrar els mapes que ja han estat explorats (per qualsevol branca). **IMPORTANT**: A la cache d'estats cal que deseu també la profunditat, doncs només podem descartar el mapa actual si ja hem visitat el mateix mapa a menor profunditat.

ENCARA MÉS IMPORTANT:

Tota la feina es dura a terme dins d'un repositori **GitHub**. Cal que hi hagi com a mínim un commit per cada sessió de treball. Si és una sessió de treball llarga, feu varis commits quan us sembli oportú. **Heu de compartir** l'enllaç al repositori i convidar al professor com a part del lliurament.

TERCERA PART: Concurs [1p]

Dissenyeu una heurística més avançada per accelerar la cerca, implementeu-la a ***HeuristicaAvancada***.

L'heurística competirà amb la resta de companys, i la puntuació de l'apartat sortirà de la vostra posició al ranking.

Documenteu l'heurística mínimament.

QUARTA PART: Estudi [3p]

En aquest apartat he de fer un estudi comparatiu de les diferents alternatives de cerca.

Podeu fer servir els 4 mapes (A,B,C,D) que us proporcionem, que tenen nivell de dificultat incremental.

[1] **[0.5p]** Calculeu el factor d'embrancament mig (real) del mapa C per BFS usant LNT i sense fer-la servir. Justifiqueu les avantatges d'usar la LNT respecte la l'opció que només evitar cicles en el camí actual.

[2] **[1.5p]** Analitzeu comparativament els resultats dels diferents algorismes (només amb la versió LNT), justificant-ne el comportament. Tingueu en compte els següents aspectes:

- Si acaben la cerca o no (i si no acaben, indicar el motiu: temps excessiu, manca de memòria...)
- Si han trobat la solució, i en aquest cas, si és l'òptima.
- El nombre de nodes explorats. (en el cas de IDS, cal acumular per cada iteració que no troba solució)
- La memòria pic utilitzada (s'aproximarà com la mida màxima LNO (+ la mida de la LNT si s'utilitza))
- Nodes tallats per evitar cicles (*prunning*)
- Temps d'execució.

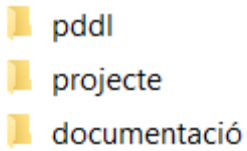
Podeu usar quadres de resum i gràfiques.

Format de lliurament

Un dels components de l'equip lliurarà un arxiu comprimit amb tots els arxius del lliurament:

Activitat1_[NIF1]_[NIF2].zip

Contingut:



- Carpeta **pddl**: (exercici 1)
  - domain.pddl
  - problem.pddl
- Carpeta **projecte**: (exercici 2 i 3)
  - ***pom.xml***
  - Carpeta ***src*** amb les fonts del projecte
  - Readme.md amb el link al github del projecte
- Carpeta **documentació**: (exercici 4)
  - estudi.pdf

El projecte ha de compilar correctament, altrament l'exercici tindrà un zero directament.