

Locks & Keys

PROP Q1 25/26: Primera activitat

STRIPS, PDDL, cerca cega i informada

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.

Mapa C:

```
######
#.....a.#.#
#1.....#
#...#####
##..A..B..C@##
##..###..##C#
##..##..#...
#....2....
######
```

- **Sense LNT:** cada branca pot redescobrir els mateixos estats, per tant hi ha una exploració redundant, i això gasta més temps i memòria.

- **Amb LNT:** cada estat es visita una sola vegada, per tant, menys successors inútils i recorregut molt més eficient.

En mapes amb múltiples camins i claus (com el mapa C), la **LNT redueix dràsticament la redundància i el factor d'embrancament efectiu.**

Per tant, el control per **LNT** redueix el factor d'embrancament real i accelera la cerca, ja que evita expandir estats repetits que només augmenten la *LNO* sense aportar nova informació.

[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.

Amb la LNT Activada, els diferents algoritmes:

BFS

- **Acaben la cerca?**
 - Sí, sempre que hi hagi suficient memòria per a la LNO/LNT. BFS és complet (troba solució si n'hi ha).
- **Solució i si es òptim:**
 - Si troben solució **garanteix la solució** en nombre de moviments.
- **Nombre de nodes explorats:**
 - Elevat en general. Explora tots els estats per profunditats creixents fins a la profunditat de la solució. En mapes amb moltes rames explora moltíssims nodes.
- **Memòria:**
 - **Alta.** En mapes grans amb molts estats pot fallar per **memòria** (LNO creix exponencialment amb la profunditat de la solució)..
 - A mesura que la profunditat de solució creix, la memòria creix exponencialment.
- **Nodes tallats (pruning):**
 - Bastant alt: LNT evita reexpandir estats ja vistos, per tant moltes generacions es tallen. Però encara es generen molts successors que després es detecten a LNT.
- **Temps d'execució:**
 - Pot ser alt degut al nombre d'expansions i a la gestió gran de LNO/LNT.

Quan usar-lo: quan necessites trobar sempre el **òptim** i tens memòria suficient.

DFS

- **Acaben la cerca?**

- Pot acabar si la solució està a una profunditat accessible i l'algorisme explora la branca correcta. Amb LNT, es prevé reexpansions, així que evita cicles; però no té garantia de trobar solució ràpid. En grans mapes pot trigar molt.
- En la pràctica, amb LNT activada sí evita bucles infinitos i acabarà trobant la solució. Pero no tan ràpidament.

- **Solució i si és òptim:**

- Si troba solució però **no garanteix optimització** (pot retornar una solució molt més llarga que la mínima).

- **Nombre de nodes explorats:**

- Pot ser inferior al de BFS en molts mapes, però depèn molt de l'ordre de successors. Amb LNT es redueix la reexploració, però la cerca pot explorar branques profundes inútils abans d'arribar a la solució.

- **Memòria (pic):**

- **Baixa/moderada.** Amb LNT augmenta, però la LNO sol ser menys que la cua de BFS.

- **Nodes tallats (pruning):**

- LNT permet tallar molts estats repetits.

- **Temps d'execució:**

- Pot ser alt o baix dependent de l'ordre: sovint més ràpid que BFS en trobar alguna solució però no la millor.

Quan usar-lo: quan tens **restriccions de memòria** i et val qualsevol solució (no necessites la òptima).

IDS

- **Acaben la cerca?**
 - Sí, sempre que la profunditat màxima que iteres cobreixi la profunditat de la solució.
- **Solució i si es òptima:**
 - Quan s'usa amb increments de profunditat i cost uniforme sempre **garanteix òptim i és complet**.
- **Nombre de nodes explorats:**
 - **Molt major que BFS** en general.
 - Si la profunditat de solució és d , el cost en nodes aproximat és $O(b^d)$ però amb una constant més gran per les repeticions; en la pràctica explora menys memòria però més nodes totals.
- **Memòria:**
 - **Molt baixa** comparada amb BFS.
- **Nodes tallats (pruning):**
 - Amb LNT es poden tallar molts estats durant cada iteració; però la repetició fa que el nombre acumulat de nodes tallats sigui alt.
- **Temps d'execució:**
 - Normalment **més lent** que BFS per trobar la mateixa profunditat òptima, però utilitza menys memòria.

Quan usar-lo: quan necessites **trobar la solució òptima** però tens memòria molt limitada i la profunditat de la solució no és massa gran.

A* (A-star) Manhattan

- **Acaben la cerca?**

- Sí si la heurística és admissible i consistent; A* amb LNT i heurística admissible és complet i trobarà la solució. Pot fallar per **manca de memòria** si l'espai és enorme.

- **Solució i optimalitat:**

- **Garanteix trobar la solució òptima** si la heurística és **admissible**; la proposta (distància Manhattan) és admissible perquè ignora obstacles, per tant **no sobreestima**. Així A* troba la solució.

- **Nombre de nodes explorats:**

- **Normalment molt menor** que BFS i IDS: A* va cap a estats prometedors gràcies a l'heurística. L'estalvi també depèn molt d'aquestaheurística.
- En mapes amb claus/portes la heurística (clau més propera a la sortida) és raonable i ajudarà a prioritzar endreçar recollida de claus.

- **Memòria (pic):**

- **Moderada/Alta.** A* manté LNO (oberts) + LNT; pot ser comparable a BFS en pitjors casos, però normalment es menor si l'heurística és bona.

- **Nodes tallats (pruning):**

- LNT talla estats amb g més alt que el guardat; això dona un pruning bastant bo quan la heurística evita expansions costoses.

- **Temps d'execució:**

- Normalment **molt millor** que BFS per trobar la solució òptima, excepte quan la heurística no es molt bona.

Quan usar-lo: **preferible** si tens una heurística admissible i vols solució **òptima** amb menys exploració que BFS.