1. **Seminarska naloga**

***UMETNA INTELIGENCA***

1. **Neinformirano preiskovanje**

Sistematično preiskovanje prostora. Preiščemo vse možne kombinacije stanj.

* **DFS**

Iskanje v globino (Depth-first search). Uporaba sklada za pregled elementov. Da algoritem ne išče v globino v nedogled sem uporabil mejo za maksimalno globino iskanja (maxDepth). Algoritem najde rešitev **hitro**, z **malo število elementov v pomnilnik**, ampak najdena pot **NI OPTIMALNA**.

* **BFS**

Iskanje v širino (Breadth-first search). Uporaba vrste za pregled elementov. Algoritem najde rešitev **počasneje** kot DFS, ker mora pregledat veliko več stanj (vsa stanja do določene globine).V **pomnilniku je lahko naenkrat zelo veliko elementov** (število elementov na globini). Najdena pot je **OPTIMALNA**.

* **IDDFS**

Iterativno poglabljanje Iskanja v globino (ID Depth-first search). Uporaba rekurzije za pregled elementov. Algoritem pregleduje stanja v globino, pri tem se **globina iterativno povečuje**. MaxDepth se poveča, ko preiščemo vsa stanja na trenutni globini. Ker iščemo v globino, je **max število elementov v pomnilniku (globina)**. V primerjavi z BFS imamo veliko manj elementov v pomnilniku, zato lahko s tem algoritmom preiščemo **večji prostor**. Cena tega je, da moramo **ista stanja večkrat pregledati** (Število vseh pregledanih stanj). Najdena pot je **OPTIMALNA**.

1. **Hevristično preiskovanje**

Da prostor preiskanih stanj zmanjšamo, uporabimo hevristične ocene. Ker smo prostor stanj, ki jih moramo preiskati zmanjšali, bodo vsi algoritmu z uporabo hevristike veliko hitreje našli rešitev, kot algoritmi s sistematičnem preiskovanjem prostora.

* **GBFS**

Požrešno iskanje (Greedy Best-first search). Uporaba prioritetne vrste za pregled elementov. Algoritem najde rešitev zelo **hitro**, z **malo število elementov v pomnilnik**, ampak najdena pot **NI OPTIMALNA**.

* **A\***

Uporaba prioritetne vrste za pregled elementov. Število vseh pregledanih stanj je malo, a je zelo veliko elementov hkrati v pomnilniku. Najdena pot je **OPTIMALNA**.

* **IDA\***

Iterativno poglabljanje A\*. Uporaba prioritetne vrste in rekurzije za pregled elementov. Algoritem deluje kot A\*, dodatno se beleži meja »bound«, ki se postopoma povečuje za pregled stanj z večjo hevristično oceno. Ker iščemo, do dane meje »bound«, se **max število elementov v pomnilniku (globina)** **drastično zmanjša** v primerjavi z A\*. Zato lahko s tem algoritmom preiščemo **večji prostor**. Cena tega je, da moramo **ista stanja večkrat pregledati** (Število vseh pregledanih stanj). Najdena pot je **OPTIMALNA**.

1. **Hevristična ocean**

* **Ocena 1**: Hevristika stanja se določi po pregledu pravilnih položajev škatel. Ce imamo škatlo na napačnem mestu, bomo potrebovali vsaj se 1 premik, da bo a pravem mestu.
* **Ocena 2**: Hevristika je mal obol napredna. Deluje enako kot Ocena 1 z dodatkom, da za pravilno pozicionirano škatlo pregleda škatle pod njo. V primeru, da je skala pod njo na napačnem mestu, moramo trenutno škatlo premakniti s pozicije, na pozicijo dodati pravilno škatlo in trenutno škatlo dodati nazaj na pravo pozicijo. Pomeni da bomo potrebovali se vsaj 2 dodatna premika.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated**Primer 4 (DFS vs BFS vs IDDFS)**

Text

Description automatically generated**Text

Description automatically generatedText

Description automatically generatedPrimer 4 (GBFS vs A\* vs IDA\*)**

**Primer 5 (DFS vs BFS vs IDDFS)**

**Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated**

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated**Primer 5 (GBFS vs A\* vs IDA\*)**