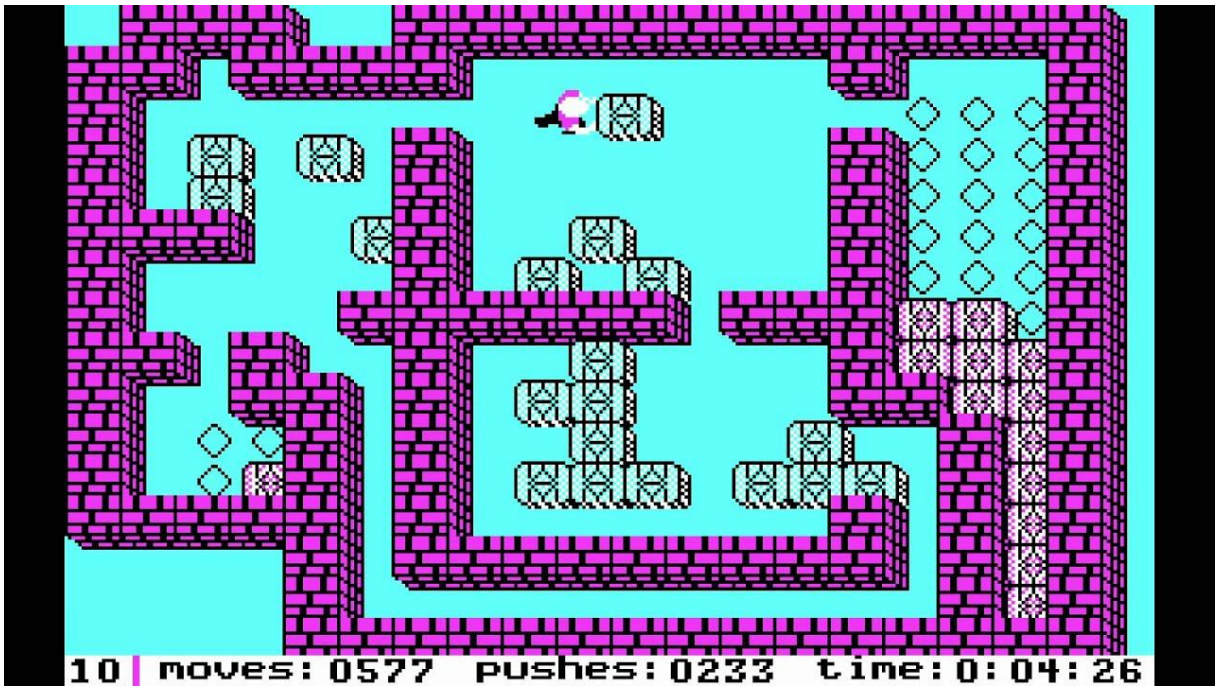
2-AIN-108/15: Výpočtová logika
Projekt

Erik Róbert Ján Jakubovský 4.12.2022

Hra Sokoban

Sokoban je logická hra navrhnutá Hiroyuki Imabayashi-om prvkýkrát vydaná v roku 1982. Cieľom hry je presunúť všetky škatuľe na vyznačené políčka. Skladník/Sokoban sa vie posunúť na lubovoľné voľné políčko. Okrem presúvania vie Sokoban tlačiť škatuľe. Problém hry spočíva v tom, že Sokoban nevie škatule ťahať, teda nejaké presunutie škatule môže viesť k tomu, že úloha nebude riešiteľná, preto treba každé posunutie dobre zvážiť.



Úloha

Úlohou je nájsť plán, ktorý rieši ľubovoľnú zadanú riešiteľnú mapu.

Úlohu riešim pomocou SAT solver-a MiniSat. Pre ľubovoľnú mapu sa vygeneruje teória vo formáte CNF tá sa následne konvertuje do formátu pre MiniSat dimacs. MiniSat potom zisťuje, či je teória splniteľná, ak je z odpovede MiniSat-u sa extrahuje plán ako postupnosť inštrukcií, ktorá vedie k vyriešeniu úlohy.

Formalizácia

Ortogonálne susediace políčko

Každé políčko: $[i, j]$ má ortogonálne susediace políčka:

$$\{[i-1, j], [i+1, j], [i, j-1], [i, j+1]\}$$

$$i, j \in \mathbb{N}$$

Políčka T

$$T = \{[i, j]\}$$

Pre každé i zodpovedajúce riadku v mape, pre každé j zodpovedajúce stĺpcu v mape, $i, j \in \mathbb{N}$

Fluenty

$$wall([i, j])$$

Pre každé políčko $[i, j]$ patriace T, také že na políčku v i-tom riadku a j-tom stĺpci v mape je stena.

$$goal([i, j])$$

Pre každé políčko $[i, j]$ patriace T, také že políčko v i-tom riadku a j-tom stĺpci v mape je cieľové políčko.

$$box([i, j])$$

Pre každé políčko $[i, j]$ patriace T, také že na políčku v i-tom riadku a j-tom stĺpci v mape je škatuľa.

$$at([i, j])$$

Pre políčko $[i, j]$ patriace T, také že políčko v i-tom riadku a j-tom stĺpci v mape je štartovacia pozícia Sokoban-a.

Akcie

$move([i,j],[k,l])$

$p^+ = \{at([i,j])\}$

$p^- = \{box([k,l]), wall([k,l])\}$

$e^+ = \{at([k,l])\}$

$e^- = \{at([i,j])\}$

Pre všetky políčka $[i,j]$, $[k,l]$, také že $[k,l]$ je ortogonálne susediacim políčkom s políčkom $[i,j]$ a $[i,j],[k,l] \in T$.

Akcia move zodpovedá posunutiu sokoban-a v mape. Sokoban sa nemôže posunúť na miesto v ktorom je prekážka (škatuľa, stena).

$push([i,j],[k,l],[m,n])$

$p^+ = \{at([i,j]), box([k,l])\}$

$p^- = \{box([m,n]), wall([m,n])\}$

$e^+ = \{at([k,l]), box([m,n])\}$

$e^- = \{at([i,j]), box([k,l])\}$

Pre všetky políčka $[i,j]$, $[k,l]$, $[m,n]$, také že $[k,l]$, $[m,n]$ tvoria dvojice: $\{([i-1,j],[i-2,j]),([i+1,j],[i+2,j]),([i,j-1],[i,j-2]),([i,j+1],[i,j+2])\}$ a zároveň $[i,j],[k,l],[m,n] \in T$.

Akcia push zodpovedá posunutiu škatuli v mape. Sokoban môže škatuľu jedine potlačiť na miesto, na ktorom nie je prekážka (škatuľa, stena). Pri tlačení sa posunie spolu so škatuľou.

$wait(\ \ )$

$p^+ = \{\ \ \}$

$p^- = \{\ \ \}$

$e^+ = \{\ \ \}$

$e^- = \{\ \ \}$

Sokoban môže nerobiť aj nič, stav mapy sa potom nezmení. Akcia má zmysel pri hľadaní plánu, kde stav sveta zodpovedá cieľovému a akákoľvek následné posunutie škatule by viedlo k neriešiteľnej úlohe a dopredu nepoznáme potrebný počet krokov Sokoban-a na vyriešenie úlohy.

Počiatočný stav

Pre každé políčko $[i, j]$ patriace T:

Ak je i-tom riadku a j-tom stĺpci v mape stena.

$$wall([i, j])$$

Inak:

$$-wall([i, j])$$

Ak je i-tom riadku a j-tom stĺpci v mape cieľové políčko:

$$goal([i, j])$$

Inak:

$$-goal([i, j])$$

Ak je na i-tom riadku a j-tom stĺpci v mape škatuľa:

$$box([i, j], 0)$$

Inak:

$$-box([i, j], 0)$$

Ak je na i-tom riadku a j-tom stĺpci v mape štartovacia pozícia Sokoban-a:

$$at([i, j], 0)$$

Inak:

$$-at([i, j], 0)$$

Poznámka: nakoľko je pozícia stien a cieľových políčok nemenná nie je potrebné zohľadňovať ich stav v každom kroku, tým pádom ich ani netreba zohľadňovať v riešení „Frame“ problému.

Cieľový stav

Pre každé políčko $[i, j]$ patriace T:

$$\big(goal([i, j]) \rightarrow box([i, j], n)\big)$$

Kde $n \in \mathbb{N}$ a n je posledný krok Sokoban-a.

Kódovanie akcii

Pre všetky $s \in \langle 1, n \rangle, s \in \mathbb{N}$, kde n je posledný krok Sokoban-a:

$move([i,j],[k,l],s) \rightarrow$

$\rightarrow at([i,j], s-1) \wedge \neg box([k,l], s-1) \wedge \neg wall([k,l]) \wedge at([k,l], s) \wedge \neg at([i,j], s)$

Pre všetky políčka $[i,j]$, $[k,l]$, také že $[k,l]$ je ortogonálne susediacim políčkom s políčkom $[i,j]$ a $[i,j],[k,l] \in T$

$push([i,j],[k,l],[m,n],s) \rightarrow$

$\rightarrow at([i,j], s-1) \wedge box([k,l], s-1) \wedge \neg box([m,n], s-1) \wedge \neg wall([m,n]) \wedge$

$\wedge at([k,l], s) \wedge box([m,n], s) \wedge \neg at([i,j], s) \wedge \neg box([k,l], s)$

Pre všetky políčka $[i,j]$, $[k,l]$, $[m,n]$, také že $[k,l]$, $[m,n]$ tvoria dvojice: $\{([i-1,j],[i-2,j]),([i+1,j],[i+2,j]),([i,j-1],[i,j-2]),([i,j+1],[i,j+2])\}$ a zároveň $[i,j],[k,l],[m,n] \in T$.

„Explanatory frame“ problém

Pre všetky $s \in \langle 1, n \rangle, s \in \mathbb{N}$, kde n je posledný krok Sokoban-a:

$$\neg box([m,n], s-1) \wedge box([m,n], s) \rightarrow \bigvee_{[k,l],[m,n] \in T} push([i,j], [k,l], [m,n], s)$$

$$box([k,l], s-1) \wedge \neg box([k,l], s) \rightarrow \bigvee_{[k,l],[m,n] \in T} push([i,j], [k,l], [m,n], s)$$

Pre všetky políčka $[i,j]$, $[k,l]$, $[m,n]$, také že $[k,l]$, $[m,n]$ tvoria dvojice: $\{([i-1,j],[i-2,j]), ([i+1,j],[i+2,j]), ([i,j-1],[i,j-2]), ([i,j+1],[i,j+2])\}$ a zároveň $[i,j], [k,l], [m,n] \in T$.

$$\neg at([k,l], s-1) \wedge at([k,l], s) \rightarrow$$

$$\rightarrow \bigvee_{[k,l],[m,n] \in T} push([i,j], [k,l], [m,n], s) \vee \bigvee_{[k,l] \in T} move([i,j], [k,l], s)$$

$$at([i,j], s-1) \wedge \neg at([i,j], s) \rightarrow$$

$$\rightarrow \bigvee_{[k,l],[m,n] \in T} push([i,j], [k,l], [m,n], s) \vee \bigvee_{[k,l] \in T} move([i,j], [k,l], s)$$

Pre všetky políčka $[i,j]$, $[k,l]$, $[m,n]$, také že $[k,l]$, $[m,n]$ tvoria dvojice: $\{([i-1,j],[i-2,j]), ([i+1,j],[i+2,j]), ([i,j-1],[i,j-2]), ([i,j+1],[i,j+2])\}$ a zároveň $[i,j], [k,l], [m,n] \in T$.

Exkluzivita akcií:

Pre všetky $s \in \langle 1, n \rangle, s \in \mathbb{N}$, kde n je posledný krok Sokoban-a:

$$\bigvee_{[i,j],[k,l],[m,n] \in T} push([i,j], [k,l], [m,n], s) \vee \bigvee_{[o,p],[q,r] \in T} move([o,p], [q,r], s)$$

$$\neg\big(push([i,j], [k,l], [m,n], s) \wedge move([o,p], [q,r], s)\big)$$

Pre všetky políčka $[i,j]$, $[k,l]$, $[m,n]$, také že $[k,l]$, $[m,n]$ tvoria dvojice: $\{([i-1,j],[i-2,j]), ([i+1,j],[i+2,j]), ([i,j-1],[i,j-2]), ([i,j+1],[i,j+2])\}$ a zároveň $[i,j], [k,l], [m,n] \in T$.

Pre všetky políčka $[o,p]$, $[q,r]$, také že $[q,r]$ je ortogonálne susediacim políčkom s políčkom $[o,p]$ a $[o,p], [q,r] \in T$

Hľadanie riešenia

Predpokladáme, že ľubovoľná mapa sa dá riešiť na 4 kroky, n = 4 (akcie Sokoban-a). SATSolver hľadá model vygenerovanej teórie:

Ak model existuje, zredukujeme n na polovicu a skúšame nájsť plán na n / 2 krokov.

Ak model neexistuje, zvýšime počet krokov dvojnásobne a skúšame nájsť plán na 2n krokov.

Zo všetkých plánov vyberieme ten, ktorý pozostáva z najmenšieho počtu krokov.

Spustenie programu

Program bol robený na operačnom systéme Windows v programovacom jazyku Python. Pre jeho spustenie je potrebné mať nainštalovaný pythonovský interpreter spolu s knižnicami Pillow, tkinter (slúžia na vizualizáciu). Pri riešení úlohy používa MiniSAT v priečinku minisat. Pre jeho spustenie stačí dvoj-kliknúť na skript main.py. V aplikácii treba zadať súbor spolu s cestou k mape, ktorú treba riešiť (všetky mapy zo zadania sú v priečinku maps). Následne možno zadať meno pre pomocné súbory. V GUI je uvedená default mapa a názov pomocného súboru ako fungujúca ukážka. Po nájdení riešenia možno vo vyskakovacom okne vybrať možnosť „áno“ pre vizualizáciu plánu. Pomocné súbory predstavujú teóriu zakódovanú v CNF v priečinku cnf, teóriu zakódovanú vo formáte dimacs v priečinku dimacs a odpoveď MiniSAT-u v priečinku answer. Každý vygenerovaný súbor končí číslovkou reprezentujúcou maximálny počet krokov Sokoban-a.

Skúsenosť s projektom

Pri riešení som použil poznatky z cvičení a domácich úloh na vytvorenie plánovača hry Sokoban. Úlohu som riešil pomocou SATsolver-a nakoľko, som nemal toľko času a príležitostí na precvičenie plánovania v Prolog-u alebo ASP.

Pri riešení úlohy som narazil na problém kedy generovanie CNF súboru trvalo príliš dlho kvôli zapisovaní „Frame“ problému. Čas generovania súboru sa mi podarilo skrátiť využitím „Explanatory Frame“ problému. Súbory obsahujú zbytočné množstvo literálov, ktorých počet by sa dal v budúcnosti zredukovať pomocou „action overriding“.

Projekt sa mi riešil celkom ťažko, nakoľko už pri jednoduchých úlohách bol problém nájsť chybu v tisícriadkových CNF súboroch.