Úvod do umělé inteligence (NAIL120)

1. a 2. cvičení

Jirka Fink

https://ktiml.mff.cuni.cz/~fink/

Katedra teoretické informatiky a matematické logiky
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Letní semestr 2025/26

Poslední změna 16. února 2026

Oblasti umělé inteligence

Oblasti umělé inteligence

- Strojové učení
- Plánování
- Evoluční algoritmy a metaheuristiky
- Zpracování obrazu, zvuků a videí
- Zpracování přirozených jazyků
- Autonomní vozidla
- Robotika
- Reprezentace znalostí

Obsah předmětu

- Zavedení pojmů, historie
- Řešení úloh prohledáváním (A* a spol.)
- Splňování podmínek (Constraint satisfaction programming)
- Logické uvažování (dopředné a zpětné řetězení, rezoluce, SAT)
- Automatické plánování
- Pravděpodobnostní uvažování a rozhodování
- Hry a teorie her
- Strojové učení (rozhodovací stromy, regrese, zpětnovazební učení)
- Filozofické a etické aspekty

Zápočet

Podmínky

- Zápočet bude udělen za vypracování domácích úkolů
- Programovací jazyk: Python
- Úkoly budou každý týden a na vypracování budou (většinou) 2 týdny
- Pozdní odevzdání úkolu se nepřipouští
- Úkoly se vypracovávají samostatně
- Minimální počet bodů k zápočtu je 70 bodů
- Podrobnosti: `https://ktiml.mff.cuni.cz/~fink/teaching/artificial_intelligence_intro/`

Zápočet

Podmínky

- Zápočet bude udělen za vypracování domácích úkolů
- Programovací jazyk: Python
- Úkoly budou každý týden a na vypracování budou (většinou) 2 týdny
- Pozdní odevzdání úkolu se nepřipouští
- Úkoly se vypracovávají samostatně
- Minimální počet bodů k zápočtu je 70 bodů
- Podrobnosti: `https://ktiml.mff.cuni.cz/~fink/teaching/artificial_intelligence_intro/`

Hodnocení

- Máte k dispozici všechny testy, které jsou v Recodexu
- Odevzdané řešení musí řešit úlohu v plné obecnosti zadání
  tj. nesmíte mít předpočítané hodnoty na zadané testy
- Bodování závisí na počtech testů, které splníte
- Počet bodů závisí též na kvalitě zpracování (hodnoceno ručně)

Komunikace

Web

`https://ktiml.mff.cuni.cz/~fink/`

E-mail

fink@ktiml.mff.cuni.cz

GIT

- Zadání domácích úkolů
- Šablony, které máte za domácí úkoly doplnit
- `https://gitlab.mff.cuni.cz/finkj1am/introai`

Recodex

- Odevzdávání a automatická kontrola (unit testy)
- Komentáře k Vašim řešením

Plánované domácí úkoly

1. Heuristiky pro A* algoritmus
2. Úplné barvení pomocí Constraint satisfaction programming (CSP)
3. 3-partition pomocí logických podmínek (SAT)
4. Přeprava balíků s využitím automatického plánování
5. Hledání min s použitím podmíněné pravděpodobnosti
6. Lokalizace robota pomocí Markovských procesů
7. Hledání cesty pro rozbitého robota pomocí Bellmanovy rovnice
8. Alfa-beta prořezávání pro jednoduchou hru
9. Rozhodovací stromy pro zjišťování cukrovky
10. Klasifikace článků neuronovou sítí

Literatura a další zdroje

- Russell, Norvig: Artificial Intelligence: A modern approach
  - K půjčení v malostranské knihovně MFF
  - Elektronicky: `https://ebookcentral.proquest.com/lib/cuni/detail.action?docID=5495854`
- MIT video přednáška: `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/`
- Internet . . .

Moje další výuka

Large-scale optimization: Metaheuristics

- Metody řešení optimalizačních úloh založených na kombinaci různých heuristik umělé inteligence
- Přednáška: Jakub Bulín, pátek, 9:00, S1
- Cvičení: pátek, 10:40, S4

Moje další výuka

Large-scale optimization: Metaheuristics

- Metody řešení optimalizačních úloh založených na kombinaci různých heuristik umělé inteligence
- Přednáška: Jakub Bulín, pátek, 9:00, S1
- Cvičení: pátek, 10:40, S4

Ročníkový projekt, bakalářská a diplomová práce

- Kombinatorika a teorie grafů
- Přírodou inspirované algoritmy
- Optimalizace

Podrobnosti: pátek 20.2., 12:20, pracovna S305

Jak souvisí hledání nejkratší cesty s UI?

Terminologie

- Stav = vrchol: Úplný popis jedné konfigurace
- Akce = hrana: Atomická změna konfigurace
- Cena akce = váha (délka) hrany
- Počáteční stav = počáteční vrchol
- Cíl = množina koncových vrcholů
- Stavový prostor = množina vrcholů
- Transition model = funkce (stav,akce) $\rightarrow$ stav

Jak souvisí hledání nejkratší cesty s UI?

Terminologie

- Stav = vrchol: Úplný popis jedné konfigurace
- Akce = hrana: Atomická změna konfigurace
- Cena akce = váha (délka) hrany
- Počáteční stav = počáteční vrchol
- Cíl = množina koncových vrcholů
- Stavový prostor = množina vrcholů
- Transition model = funkce (stav,akce) $\rightarrow$ stav

Proč měnit terminologii?

Příklady hledání cest v UI

- Loydova patnáctka
- Sokoban
- Rubikova kostka
- Další hlavolamy

Prohledávání grafu

Input: Graf G, počáteční vrchol s a cílový t

```
1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
4     Zvol u libovolný navštívený vrchol
5     for v soused u do
6         if v je nenavštívený then
7             Označ v za navštívený
8     Označ u za prozkoumaný
```

Prohledávání grafu

Input: Graf G, počáteční vrchol s a cílový t

```
1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
4     Zvol u libovolný navštívený vrchol
5     for v soused u do
6         if v je nenavštívený then
7             Označ v za navštívený
8     Označ u za prozkoumaný
```

Poznámky

- Jestliže s a t leží ve stejné komponentě, tak skončíme prozkoumáním t, jinak projdeme celou komponentu obsahující s

Prohledávání grafu

Input: Graf G, počáteční vrchol s a cílový t

```
1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
4     Zvol u libovolný navštívený vrchol
5     for v soused u do
6         if v je nenavštívený then
7             Označ v za navštívený
8     Označ u za prozkoumaný
```

Poznámky

- Jestliže s a t leží ve stejné komponentě, tak skončíme prozkoumáním t, jinak projdeme celou komponentu obsahující s
- Průchod do hloubky: vybíráme poslední navštívený vrchol
- Průchod do šířky: vybíráme první navštívený vrchol

Prohledávání grafu

Input: Graf G, počáteční vrchol s a cílový t

```
1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
4     Zvol u libovolný navštívený vrchol
5     for v soused u do
6         if v je nenavštívený then
7             Označ v za navštívený
8     Označ u za prozkoumaný
```

Poznámky

- Jestliže s a t leží ve stejné komponentě, tak skončíme prozkoumáním t, jinak projdeme celou komponentu obsahující s
- Průchod do hloubky: vybíráme poslední navštívený vrchol
- Průchod do šířky: vybíráme první navštívený vrchol
- Existuje řada variant: více počátečních i koncových vrcholů, nalezení cest do všech vrcholů, ...

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Pro prozkoumané vrcholy u je $d[u]$ délka nejkratší cesty z s to u
- Algoritmus označuje vrcholy za prozkoumané v neklesající vzdálenosti od počátku

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Pro prozkoumané vrcholy u je $d[u]$ délka nejkratší cesty z s to u
- Algoritmus označuje vrcholy za prozkoumané v neklesající vzdálenosti od počátku
- Prozkoumány jsou všechny vrcholy ve vzdálenosti menší než je vzdálenost do cíle

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Pro prozkoumané vrcholy u je $d[u]$ délka nejkratší cesty z s to u
- Algoritmus označuje vrcholy za prozkoumané v neklesající vzdálenosti od počátku
- Prozkoumány jsou všechny vrcholy ve vzdálenosti menší než je vzdálenost do cíle
- Graf může být příliš velký, takže d si pamatujeme jen pro navštívené vrcholy

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Pro prozkoumané vrcholy u je $d[u]$ délka nejkratší cesty z s to u
- Algoritmus označuje vrcholy za prozkoumané v neklesající vzdálenosti od počátku
- Prozkoumány jsou všechny vrcholy ve vzdálenosti menší než je vzdálenost do cíle
- Graf může být příliš velký, takže d si pamatujeme jen pro navštívené vrcholy
- Která města navštívíme při hledání cesty z Prahy do Brna?

Dijkstrův algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s a cílový t

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a cílový vrchol není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u]
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Pro prozkoumané vrcholy u je $d[u]$ délka nejkratší cesty z s to u
- Algoritmus označuje vrcholy za prozkoumané v neklesající vzdálenosti od počátku
- Prozkoumány jsou všechny vrcholy ve vzdálenosti menší než je vzdálenost do cíle
- Graf může být příliš velký, takže d si pamatujeme jen pro navštívené vrcholy
- Která města navštívíme při hledání cesty z Prahy do Brna?
- Vizualizace: `https://qiao.github.io/PathFinding.js/visual/`

A* algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s

1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 Délka nejkratší zatím nalezené cesty z s do u je $d[u] := \infty$ kromě $d[s] := 0$
4 **while** *existuje navštívený vrchol a libovolný cíl není prozkoumaný* **do**
5 $u :=$ navštívený vrchol s nejmenší hodnotou $d[u] + h(u)$
6 **for** v *soused* u **do**
7 **if** $d[v] > d[u] + c(u, v)$ **then**
8 $d[v] := d[u] + c(u, v)$
9 Označ v za navštívený
10 Označ u za prozkoumaný

A* algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a libovolný cíl není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u] + h(u)
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Heuristická funkce $h(u)$ dává odhad vzdálenosti z u nejbližšího cíle

A* algoritmus

Input: Graf G s nezápornou délkou hran c, počáteční vrchol s

1 Všechny vrcholy označ za nenavštívené
2 Počáteční vrchol označ za navštívený
3 Délka nejkratší zatím nalezené cesty z s do u je $d[u] := \infty$ kromě $d[s] := 0$
4 **while** *existuje navštívený vrchol a libovolný cíl není prozkoumaný* **do**
5 $u :=$ navštívený vrchol s nejmenší hodnotou $d[u] + h(u)$
6 **for** v *soused* u **do**
7 **if** $d[v] > d[u] + c(u, v)$ **then**
8 $d[v] := d[u] + c(u, v)$
9 Označ v za navštívený
10 Označ u za prozkoumaný

Poznámky

- Heuristická funkce $h(u)$ dává odhad vzdálenosti z u nejbližšího cíle
- Pokud $h(u) = 0$ pro všechny vrcholy, pak se A* chová stejně jako Dijkstra

A* algoritmus

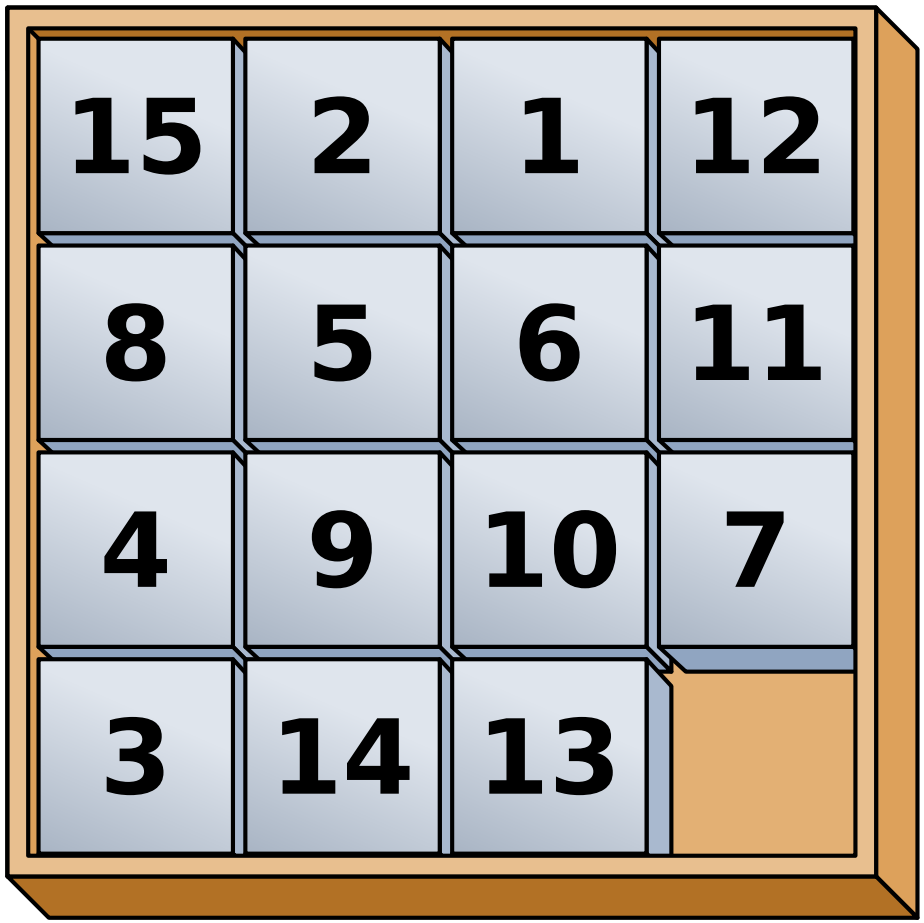
Input: Graf G s nezápornou délkou hran c, počáteční vrchol s

```
1  Všechny vrcholy označ za nenavštívené
2  Počáteční vrchol označ za navštívený
3  Délka nejkratší zatím nalezené cesty z s do u je d[u] := ∞ kromě d[s] := 0
4  while existuje navštívený vrchol a libovolný cíl není prozkoumaný do
5      u := navštívený vrchol s nejmenší hodnotou d[u] + h(u)
6      for v soused u do
7          if d[v] > d[u] + c(u, v) then
8              d[v] := d[u] + c(u, v)
9              Označ v za navštívený
10     Označ u za prozkoumaný
```

Poznámky

- Heuristická funkce $h(u)$ dává odhad vzdálenosti z u nejbližšího cíle
- Pokud $h(u) = 0$ pro všechny vrcholy, pak se A* chová stejně jako Dijkstra
- Heuristiku musíme rychle spočítat, ideálně v $O(1)$, ale přesnou délku nejkratší cesty obvykle nedokážeme rychle určit

Vymyslete heuristiku pro Loydovu 15



Definice heuristik

Definice

Heuristika h je

- přípustná (admissible), jestliže $0 \leq h(u) \leq c^*(u)$
- monotónní (monotonous, consistent), jestliže $0 \leq h(u) \leq h(v) + c(u, v)$

pro všechny vrcholy u a hrany uv a cílové vrcholy t, kde $c^*(u)$ je délka nejkratší cesty z u do nejbližšího cíle.

Definice heuristik

Definice

Heuristika h je

- přípustná (admissible), jestliže $0 \leq h(u) \leq c^*(u)$
- monotónní (monotonous, consistent), jestliže $0 \leq h(u) \leq h(v) + c(u, v)$

pro všechny vrcholy u a hrany uv a cílové vrcholy t, kde $c^*(u)$ je délka nejkratší cesty z u do nejbližšího cíle.

Cvičení

Rozhodněte, zda pro silniční síť jsou následující heuristiky přípustné a monotónní.

- Euklidovská vzdálenost: $h_2(a, b) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$
- Manhattanská metrika: $h_1(a, b) = |b_1 - a_1| + |b_2 - a_2|$
- Maximová metrika: $h_\infty(a, b) = \max\{|b_1 - a_1|, |b_2 - a_2|\}$

Definice heuristik

Definice

Heuristika h je

- přípustná (admissible), jestliže $0 \leq h(u) \leq c^*(u)$
- monotónní (monotonous, consistent), jestliže $0 \leq h(u) \leq h(v) + c(u, v)$

pro všechny vrcholy u a hrany uv a cílové vrcholy t, kde $c^*(u)$ je délka nejkratší cesty z u do nejbližšího cíle.

Cvičení

Rozhodněte, zda pro silniční síť jsou následující heuristiky přípustné a monotónní.

- Euklidovská vzdálenost: $h_2(a, b) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$
- Manhattanská metrika: $h_1(a, b) = |b_1 - a_1| + |b_2 - a_2|$
- Maximová metrika: $h_\infty(a, b) = \max\{|b_1 - a_1|, |b_2 - a_2|\}$

Otázky

- Je každá monotónní heuristika přípustná?
- Je každá přípustná heuristika monotónní?

Definice heuristik

Definice

Heuristika h je

- přípustná (admissible), jestliže $0 \leq h(u) \leq c^*(u)$
- monotónní (monotonous, consistent), jestliže $0 \leq h(u) \leq h(v) + c(u, v)$

pro všechny vrcholy u a hrany uv a cílové vrcholy t, kde $c^*(u)$ je délka nejkratší cesty z u do nejbližšího cíle.

Cvičení

Rozhodněte, zda pro silniční síť jsou následující heuristiky přípustné a monotónní.

- Euklidovská vzdálenost: $h_2(a, b) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}$
- Manhattanská metrika: $h_1(a, b) = |b_1 - a_1| + |b_2 - a_2|$
- Maximová metrika: $h_\infty(a, b) = \max\{|b_1 - a_1|, |b_2 - a_2|\}$

Otázky

- Je každá monotónní heuristika přípustná?
- Je každá přípustná heuristika monotónní?
- Proč potřebujeme, aby heuristika byla monotónní?

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Pozorování

Předpokládejme, že máme monotónní heuristiku.

- Hodnoty $f(u)$ jsou neklesající na všech nejkratších cestách ze startu.

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Pozorování

Předpokládejme, že máme monotónní heuristiku.

- Hodnoty $f(u)$ jsou neklesající na všech nejkratších cestách ze startu.
- A* prozkoumává (uzavírá) stavy v pořadí, ve kterém hodnoty $f(u)$ neklesají.
  Základní verze A* neurčuje pořadí prozkoumávání stavů se stejnou hodnotou $f(u)$.

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Pozorování

Předpokládejme, že máme monotónní heuristiku.

- Hodnoty $f(u)$ jsou neklesající na všech nejkratších cestách ze startu.
- A* prozkoumává (uzavírá) stavy v pořadí, ve kterém hodnoty $f(u)$ neklesají. Základní verze A* neurčuje pořadí prozkoumávání stavů se stejnou hodnotou $f(u)$.
- Při prozkoumávání stavu u je hodnota $d[u]$ rovna $g(u)$.

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Pozorování

Předpokládejme, že máme monotónní heuristiku.

- Hodnoty $f(u)$ jsou neklesající na všech nejkratších cestách ze startu.
- A* prozkoumává (uzavírá) stavy v pořadí, ve kterém hodnoty $f(u)$ neklesají. Základní verze A* neurčuje pořadí prozkoumávání stavů se stejnou hodnotou $f(u)$.
- Při prozkoumávání stavu u je hodnota $d[u]$ rovna $g(u)$.
- A* vždy najde optimální plán (nejkratší cestu).

Vlastnosti A*

Značení

- $h(u)$: heuristika z u do cíle
- $g(u)$: vzdálenost ze startu do cíle
- $f(u) = h(u) + g(u)$
- $d[u]$: proměnná v A* udávající délku nejkratší nalezené cesty

Pozorování

Předpokládejme, že máme monotónní heuristiku.

- Hodnoty $f(u)$ jsou neklesající na všech nejkratších cestách ze startu.
- A* prozkoumává (uzavírá) stavy v pořadí, ve kterém hodnoty $f(u)$ neklesají. Základní verze A* neurčuje pořadí prozkoumávání stavů se stejnou hodnotou $f(u)$.
- Při prozkoumávání stavu u je hodnota $d[u]$ rovna $g(u)$.
- A* vždy najde optimální plán (nejkratší cestu).
- A* prozkoumá všechny vrcholu u splňující $f(u) < C^*$ a některé vrcholy s $f(u) = C^*$, kde C^* je délka nejkratší cesty.

Kombinování heuristik

Otázka

Máme-li dvě nebo více heuristik pro daný problém, jak poznat, kterou máme zvolit?

Kombinování heuristik

Otázka

Máme-li dvě nebo více heuristik pro daný problém, jak poznat, kterou máme zvolit?

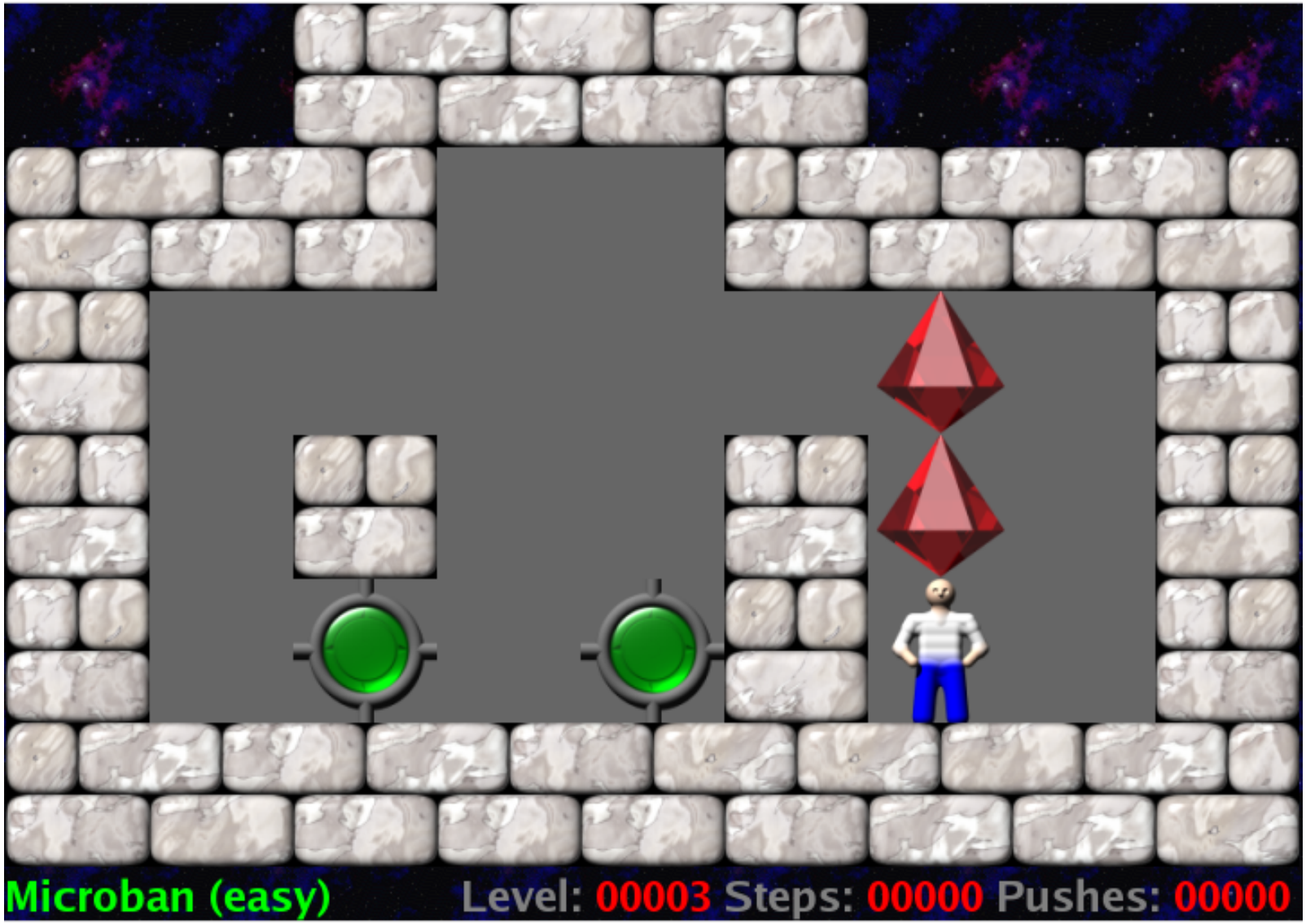
Pozorování

Dokažte, že pro přípustné/monotónní heuristiky h_1 a h_2 je též

$$\max\{h_1, h_2\}$$

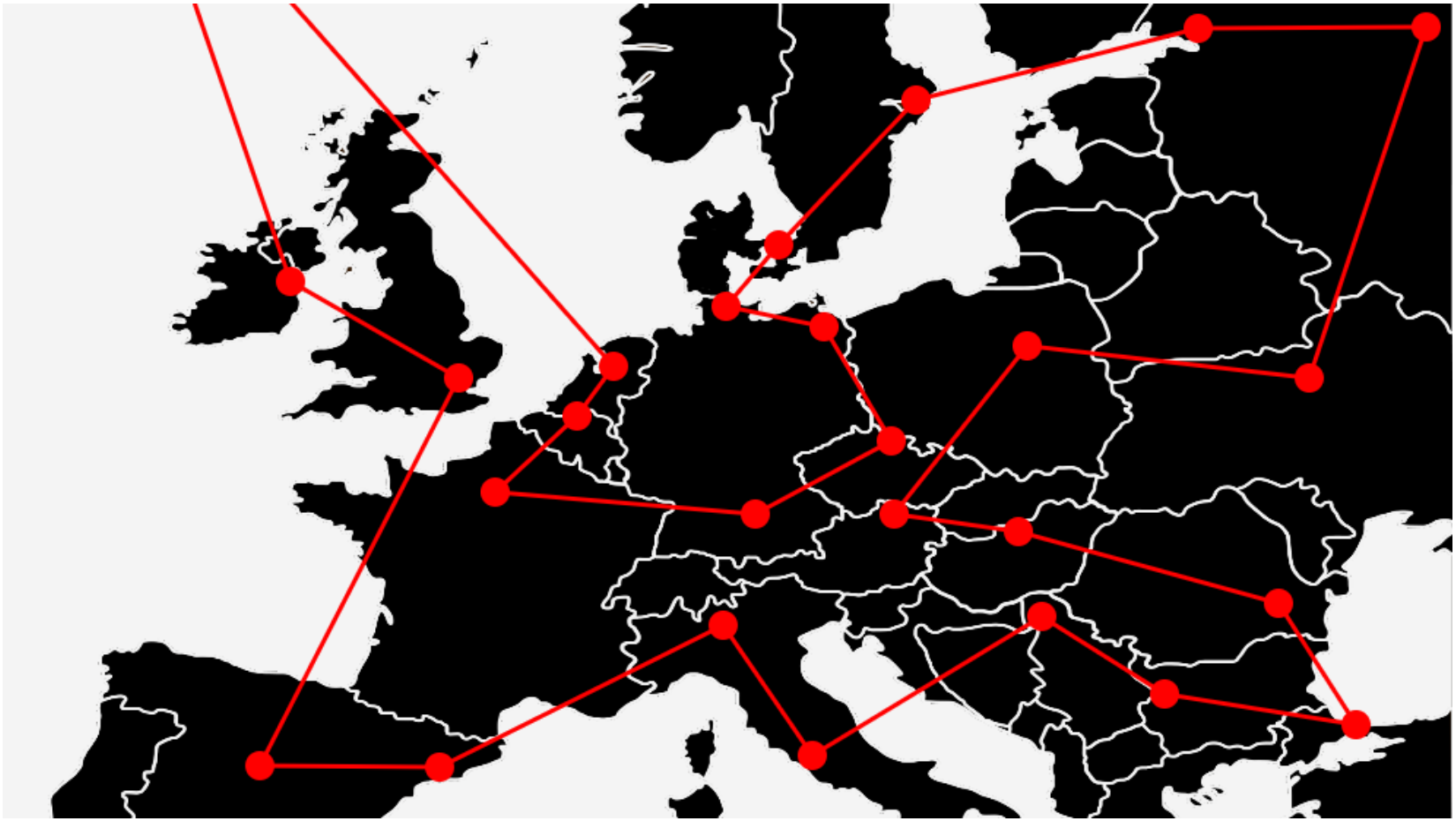
přípustná/monotónní heuristika.

Vymyslete heuristiku pro Sokoban

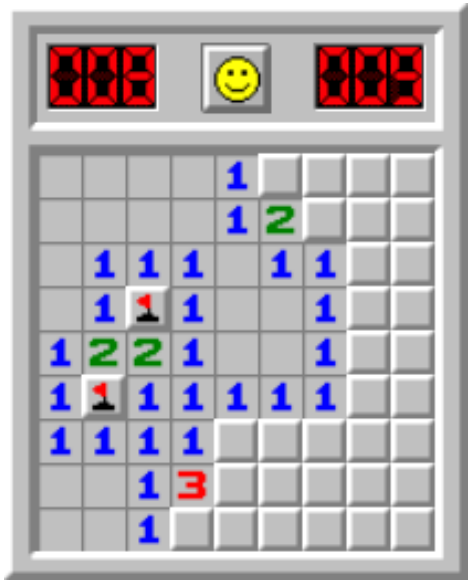


- Animovaná verze
- Přehled postupů

Vymyslete heuristiku pro Obchodního cestujícího

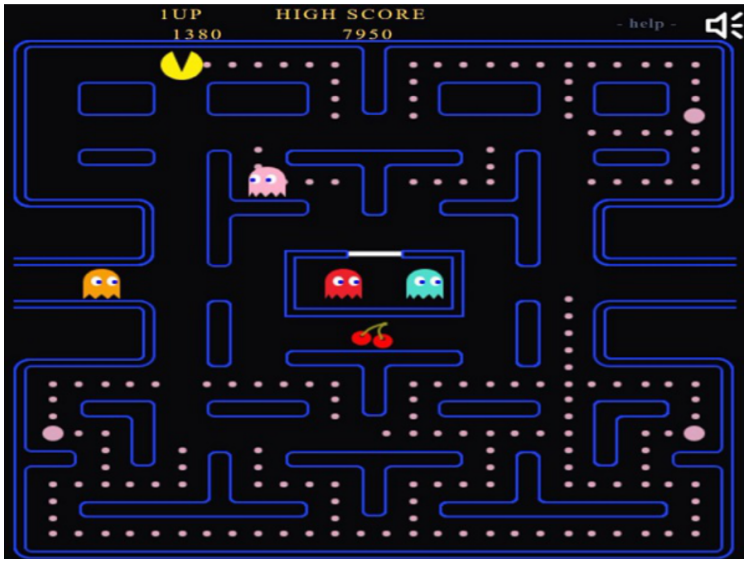


Dokážete vymyslet heuristiky pro tyto hry?



Minesweeper

Šachy



Packman

Kulečník

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka
- Dvourozměrná mřížka obsahující i úhlopříčky
- Třírozměrná mřížka obsahující stěnové i prostorové úhlopříčky

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka
- Dvourozměrná mřížka obsahující i úhlopříčky
- Třírozměrná mřížka obsahující stěnové i prostorové úhlopříčky
- Třírozměrná mřížka obsahující stěnové úhlopříčky ale nikoliv prostorové

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka
- Dvourozměrná mřížka obsahující i úhlopříčky
- Třírozměrná mřížka obsahující stěnové i prostorové úhlopříčky
- Třírozměrná mřížka obsahující stěnové úhlopříčky ale nikoliv prostorové
- Hrany odpovídají právě pohybům věže po šachovnici

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka
- Dvourozměrná mřížka obsahující i úhlopříčky
- Třírozměrná mřížka obsahující stěnové i prostorové úhlopříčky
- Třírozměrná mřížka obsahující stěnové úhlopříčky ale nikoliv prostorové
- Hrany odpovídají právě pohybům věže po šachovnici
- Skokan se pohybuje o 3 políčka v jedné souřadnici a o 2 políčka v druhé souřadnici

1. domácí úkol: A* heuristiky

Zadání (zkráceno)

Implementujte **monotónní** heuristiky pro **A*** algoritmus spuštěný na **podgrafy** následujících nekonečných mřížek.

- Klasická dvourozměrná mřížka
- Klasická třírozměrná mřížka
- Dvourozměrná mřížka obsahující i úhlopříčky
- Třírozměrná mřížka obsahující stěnové i prostorové úhlopříčky
- Třírozměrná mřížka obsahující stěnové úhlopříčky ale nikoliv prostorové
- Hrany odpovídají právě pohybům věže po šachovnici
- Skokan se pohybuje o 3 políčka v jedné souřadnici a o 2 políčka v druhé souřadnici
- Král v sedmimílových botách, který může až o 8 políček horizontálně i vertikálně

Zadání: `https://gitlab.mff.cuni.cz/finkj1am/introai/-/blob/master/01-a_star_heuristic/task.md`

1. domácí úkol: A* heuristiky

Rady

- Úkolem je najít heuristiku, nikoliv přesnou vzdálenost
- Zkuste vymyslet dolní odhad na minimální počet kroků
- Vycházejte z heuristik diskutovaných na cvičení, uzpůsobte je danému grafu a kombinujte je
- Na většinu mřížek stačí 1-2 řádková heuristika, v jednom případě zhruba 5 řádků
- Jestliže celé číslo je větší než 5.5, pak je větší nebo rovno 6
- Nepište nic bez přemýšlení