Příloha bakalářské práce
Logik - algoritmy a strategie

Martin Šimša
Katedra algebry

Vedoucí práce: doc. Mgr. Pavel Růžička, Ph.D.

2025

1 Úvod

Toto je dokumentace k programu na testování algoritmů řešící hru [n,k]-Mastermind v souboru `mastermind-testing-algorithms.py`. Zároveň slouží jako příloha bakalářské práce s názvem Logik - algoritmy a strategie. Ta je dostupná například z https://github.com/martinsimsa/Bachelor-thesis-Mastermind.git. Případně bude k nalezení na stránce https://dspace.cuni.cz/.

V této dokumentaci používám značení z bakalářské práce.

2 Použití

2.1 Získání programu

- Otevřete soubor `mastermind-testing-algorithms.py` dostupný z přílohy bakalářské práce.
- Program je také dostupný na GitHubu, https://github.com/martinsimsa/-mastermind.git. Repozitář naklonujte
  - Clone with HTTPS: https://github.com/martinsimsa/mastermind.git
  - GitHub CLI: gh repo clone martinsimsa/mastermind
- Program běží v jazyce Python
  - https://www.python.org/downloads/
  - https://www.online-python.com/
- Nainstalujte použité knihovny
  - Deque z knihovny collections, která je součástí standardního balíčku Pythonu.
  - Numpy - https://numpy.org/install/

2.2 Spuštění

Po otevření zdrojového kódu zvolte, jakou funkci chcete spustit. Nějaké funkce byly v souboru předpřipraveny a stačí odstranit znak komentáře #.

Funkce `get_results_of_algorithm()` Tato funkce slouží k analýze algoritmu složeného z valuace a strategie. Může běžet několik minut. Tvar volání funkce je následující:

`get_results_of_algorithm(n, k, první tah, valuace, strategie, výběr z kandidátů)`

- n – počet pozic
- k – počet barev
- první tah – pevně zvolený první tah, např pro n = 4, k = 6 [1,1,2,3]
- valuace – použitá valuace
- strategie – použitá strategie
- výběr z kandidátů: True – algoritmus vybírá pouze z kandidátů, False – algoritmus vybírá ze všech kódů.

Přípustné kombinace valuace a strategie jsou:

- `find_max, lower_is_better` – algoritmus Min-max
- `find_entropy, higher_is_better` – algoritmus Max entropy
- `find_number_of_parts, higher_is_better` – algoritmus Most parts.

Příklad volání: `get_results_of_algorithm(4, 6, [1,1,2,2], find_entropy, higher_is_better, False)`

Funkce `solve_one_game()` Funkce `solve_one_game` slouží ke spuštění daného algoritmu pro jeden pevně určený tajný kód. Tvar volání funkce je následující:

`solve_one_game(n, k, tajný kód, valuace, strategie, první tah, výběr z kandidátů)`

- n – počet pozic
- k – počet barev
- tajný kód – tajný kód ve tvaru seznamu, např pro n = 4, k = 6: [5,1,6,3]
- první tah – pevně zvolený první tah, např pro n = 4, k = 6: [1,1,2,3]
- valuace – použitá valuace
- strategie – použitá strategie
- výběr z kandidátů: True – algoritmus vybírá pouze z kandidátů, False – algoritmus vybírá ze všech kódů.

Přípustné kombinace valuace a strategie jsou stejné jako výše. Příklad použití:

`solve_one_game(4, 6, [5,1,6,3], find_max, lower_is_better, [1,1,2,2], False)`

3 Technická dokumentace

3.1 Časté typy

- Kódy jsou interpretovány jako seznamy přirozených čísel (např. [1,1,2,3]).
- Množiny kódů jsou seznamy, jejichž prvky jsou kódy. Tedy to jsou vnořené seznamy.
- Ohodnocení je uchováváno jako seznam o dvou pozicích.

3.2 Proměnné

- `len_pegs` – označuje počet pozic, v bakalářské práci tuto hodnotu značíme písmenem n.
- `len_colours` – určuje počet barev (v práci jako k).
- `all_scores` – seznam všech ohodnocení v $H_{n,k}$
- `all_codes` – seznam všech kódů $H_{n,k}$
- `start_code` – volba prvního pokusu, aby algoritmus nemusel procházet všechny kódy
- `possible_codes` – Množina kandidátů aktuálního stavu.
- `partition` – Množina potomků aktuální množiny kandidátů
- `partition_table` – Velikosti množin potomků množiny kandidátů
- `secret_code` – tajný kód
- `partition_table_function` – proměnná uchovávající aktuálně používanou valuaci (`find_max, find_entropy, find_number_of_parts`)
- `compare_function` – proměnná uchovávající aktuálně používanou strategii (`higher_is_better, lower_is_better`)
- `choose_from_candidates` – proměnná, která udává, zda algoritmus vybírá kódy pro další pokusy pouze z množiny kandidátů (True), anebo z celého prostoru kódů (False)
- `partition_table_value` – uchovává hodnotu valuace pro danou množinu kandidátů a kód.

3.3 Funkce

`evaluate_codes` Funkce `evaluate_codes` na vstupu dostane dva kódy a počet pozic a barev. Vrátí ohodnocení těchto dvou kódů. Nejprve spočítá počet černých kolíčků a počty výskytů barev v obou kódech. Následně vypočítá počet bílých kolíčků podle definice ohodnocení.

generate_all_codes Funkce `generate_all_codes` vygeneruje seznam všech kódů podle zadaného počtu pozic a počtu barev. Prochází všechna čísla od nuly do $k^n - 1$ a každé číslo konvertuje do kódu, který v lexikografickém pořadí odpovídá danému číslu.

generate_all_scores Tato funkce vygeneruje seznam všech možných ohodnocení pro zadaný počet pozic `len_pegs`. Nezohledňuje počet barev, a tedy pro dvě barvy generuje i ohodnocení, která mají lichý počet bílých kolíčků. Díky použitým valuacím a strategiím to ale na algoritmech nic nezmění.

find_max Funkce pro zadané velikosti $|K_{u,r}|$ vrátí maximální hodnotu.

find_entropy Funkce pro zadané velikosti $|K_{u,r}|$ vrátí entropii tohoto rozdělení. Entropie je před vrácením výsledku zaokrouhlena na 7 desetinných míst, protože při různém pořadí výpočtu entropie se tento součet zaokrouhluje jinak a nevracel by rovnosti pro stejné rozdělení potomků.

find_number_of_parts Funkce pro zadané velikosti $|K_{u,r}|$ vrátí počet neprázdných potomků.

lower_is_better Tato funkce slouží na místo strategie. Porovnává hodnoty valuací a vrací pravdivostní hodnoty, zda je první množina menší než druhá. Ve výsledku napomáhá k nalezení minimální hodnoty valuace.

higher_is_better Tato funkce slouží na místo strategie. Porovnává hodnoty valuací a vrací pravdivostní hodnoty, zda je první množina větší než druhá. Ve výsledku napomáhá k nalezení maximální hodnoty valuace.

create_next_partition Tato funkce bere jako argumenty množinu kandidátů K (`possible_codes`) a další pokus u. Vrátí počty prvků v potomcích $K_{u,r}$ pro všechna ohodnocení $r \in S_{n,k}$.

find_best_guess `find_best_guess(possible_codes, len_pegs, len_colours, all_scores, all_codes, partition_table_function, compare_function, start_code=None, choose_from_candidates=False)`

Tato funkce pro aktuální stav a zvolenou valuaci a strategii vrátí odpovídající další pokus. Vychází z proměnné `possible_codes` - množiny kandidátů. Projde celý prostor kódů (případně pouze množinu kandidátů) a pro každý kód (`code`) nalezne potomky množiny kandidátů vzhledem k tomuto kódu (`temp_partition`) a jejich velikosti (`temp_partition_table`). O to se stará funkce `create_next_partition`. Dále nalezne valuaci aktuálního kódu (`temp_partition_table_value`) a porovná ji s aktuální nejlepší hodnotou z hlediska zvolené strategie (`compare_function`). Ve chvíli, kdy je aktuální hodnota valuace menší, respektive větší (podle zvolené strategie) než průběžná nejlepší hodnota valuace (`best_partition_table_value`), program aktualizuje nejlepší hodnoty valuace, potomků a velikostí potomků. Pokud se valuace aktuálního kódu rovná průběžné nejlepší hodnotě valuace, program zkontroluje, jestli byl průběžně

nejlepší kód kandidát. Pokud nebyl a aktuální kód je kandidátem, algoritmus aktualizuje nejlepší hodnoty valuace, potomků a velikostí potomků.

Díky tomu, že algoritmy vybírají lexikograficky nejmenší kódy z množiny kódů s nejlepší valuací (případně průniku této množiny s množinou kandidátů), tak stačí uchovávat pouze první kód s nejlepší hodnotou valuace. Případně stačí kontrolovat, zda náleží do množiny kandidátů.

Ve chvíli, kdy program projde všechny kódy, ze kterých vybírá, vrátí zvolený nejlepší kód pro tento stav (`best_next_guess`). Společně s ním vrací i proměnné `best_partition_table` a `best_partition`.

`get_results_of_algorithm` Toto je hlavní funkce, která testuje algoritmy. Funkce postupně prochází strom algoritmu. Aktuální proces uchovává ve frontě `partition_queue` vytvořené pomocí funkce `deque` z knihovny `collections`. Jednotlivé prvky fronty jsou množiny kandidátů aktuálních stavů, podle kterých algoritmus vybírá následující tah. Nejprve do této fronty přidá všechny neprázdné potomky $H_{n,k}$ vzhledem ke zvolenému prvnímu tahu. Dále funkce běží, dokud je fronta neprázdná. Pro každý stav A reprezentovaný ve frontě množinou kandidátů K nalezne další pokus u_A pomocí funkce `find_best_guess`. Dále do fronty přidá všechny neprázdné potomky množiny K vzhledem k u_A, které nebyly potomky vzhledem k ohodnocení (n,0). Potomci A vzhledem k ohodnocení (n,0) totiž odpovídají koncovým stavům. Program pro ně zapíše aktuální počet pokusů do seznamu s četnostmi počtů pokusů (`all_len_guesses`). V případě, že nějaká množina kandidátů K ve frontě je jednoprvková, program pro urychlení automaticky přičte k aktuálnímu počtu pokusů jeden a zapíše tento počet do seznamu s četnostmi počtů pokusů (`all_len_guesses`). Program končí ve chvíli, kdy je fronta prázdná, a tedy program prošel celý strom algoritmu. Stačí pouze zobrazit výsledky.

`solve_one_game` Tato funkce je implementací algoritmu 1 v bakalářské práci. Na vstupu vezme počet pozic a barev, tajný kód, valuaci, strategii, případný pevně stanovený první pokus a True/False hodnotu, jestli se další pokus vybírá pouze z kandidátů. Dále běží podle předpisu algoritmu 1. Jediná změna je ale ve výběru nejlepšího dalšího tahu. Zde program vybírá další tah pomocí funkce `find_best_guess`. Pokud vybraný kód dostane ohodnocení (n,0), tak se shoduje s tajným kódem a hra končí.

`get_valuation_of_first_guess` Tato funkce vrátí hodnotu valuace pro nějaký první pokus a zadanou valuaci. Slouží k hledání prvního pokusu, který daný algoritmus zahraje.

4 Kontakt

Martin Šimša, MFF CUNI, 3. roč., 2025, Matematika pro informační technologie, simsa.martin@email.cz, martin.simsa926@student.cuni.cz