



Semestrální práce z předmětu
Úvod do znalostního inženýrství

Hledání optimální strategie pro hru NIM

18. prosince 2024

Josef Zetek
A22B0202P
jzetek24@students.zcu.cz

Obsah

1	Úvod	2
1.1	Zadání	2
2	Analýza úlohy	3
2.1	Standardní varianta	3
2.2	Upravená varianta	4
3	Popis řešení	5
4	Popis programu	7
4.1	Volba programovacího jazyka	7
4.2	Použití knihoven	7
4.3	Volba datových struktur	8
4.4	Blokové schéma jednotlivých programových kroků	8
4.5	Programová dokumentace	9
4.5.1	CommunicationModule	9
4.5.2	ActionType	10
4.5.3	GameState	11
4.5.4	Move	12
4.5.5	Explanation	13
4.5.6	InferenceModule	13
4.5.7	Knowledge	18
4.5.8	Vývojový diagram hlavní části programu	19
4.5.9	Obecný vývojový diagram	20
5	Popis obsluhy programu	21
5.1	Základní nastavení	21
5.2	Hra	22
5.3	Závěr hry	23
6	Rozbor výsledků a závěr	24

Kapitola 1

Úvod

1.1 Zadání

Je dáno N hromádek ($3 \leq N \leq 10$) zápalek, ze kterých střídavě odebírají dva hráči zápalky tak, že mohou odebrat libovolný počet zápalek z jedné hromádky, případně odebrat stejný počet zápalek ze dvou hromádek. Vyhrává ten, kdo odebral poslední zápalku.

Napište program, který hledá optimální strategii pro jednoho hráče.

Úlohu lze naprogramovat a odladit v jednom z následujících programovacích jazyků:

- Python
- Prolog
- Java

Kapitola 2

Analýza úlohy

2.1 Standardní varianta

Hra NIM je hra pro dva hráče. Zpočátku obsahují hromádky libovolné počty zápalek, viz. příklad níže:

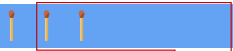


1. hromádka - 3 zápalka
2. hromádka - 5 zápalky
3. hromádka - 7 zápalek

Hráči se střídají a při každém tahu mohou vybrat z libovolné hromádky libovolný kladný počet zápalek. Optimální metoda je relativně přímočará - spočívá v jednoduchém párování dvojic napříč hromádkami.

Každé hromádce se přiřadí hodnota počtu a přes všechny hromádky se provede exkluzivní součet. Pokud budeme vycházet z výchozího stavu, viz. výše, provedeme následující výpočet.

$$NIM = 3 \oplus 5 \oplus 7 = 1 \quad (2.1)$$

Výsledná hodnota 1 je takzvaný NIM součet. Optimální strategie spočívá v tom, že hráči se navzájem snaží svého protivníka dostat do nepříznivé situace. Nepříznivá situace se vyznačuje NIM součtem 0, a proto by v tomto případě byl ideální krok z libovolné hromádky odebrat 1 zápalku.

ČÍSLA HROMÁDEK	POČTY ZÁPALKY	BINÁRNÍ REPREZENTACE	VIZUÁLNÍ REPREZENTACE
1	3	11	
2	5	101	
3	7	111	

Obrázek 2.1: Diagram řešení exkluzivního součtu u standardní varianty

2.2 Upravená varianta

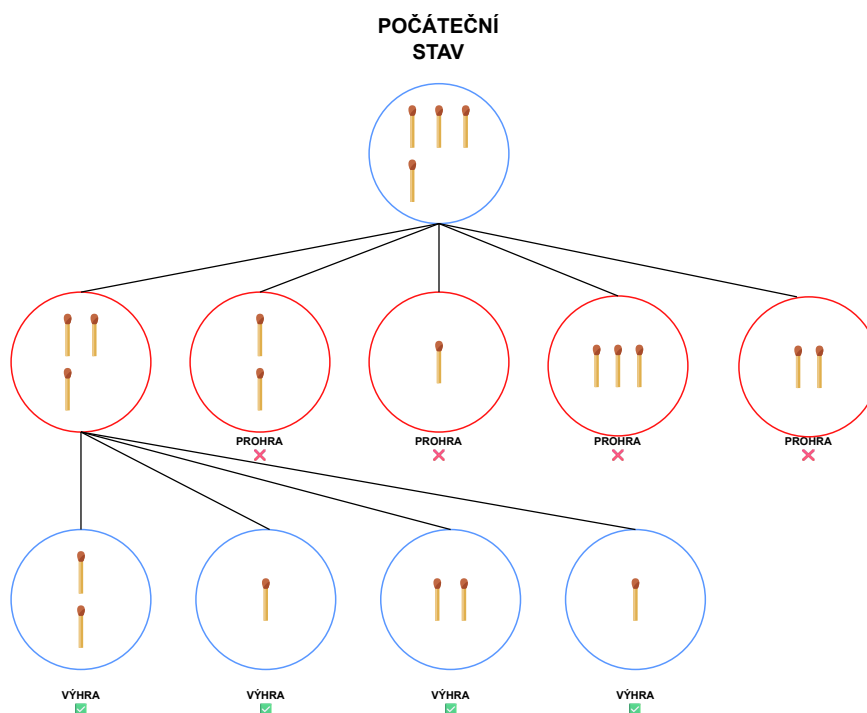
Výše v kapitole 2.1 jsou zmíněna pravidla standardní varianty. Tato úloha má oproti tomu úpravu spočívající v možnosti odebrat stejný počet zápalek ze dvou hromádek. Díky tomu se nedá využít předchozí strategie, protože ta staví na základu, že hráč nebude mít možnost odebírat z více než jedné hromádky v jednom kole.

Z výše uvedeného důvodu je proto potřeba zvolit obecnější řešení, a to prohledávání grafem. Známý a používaný algoritmus je Minmax, jehož princip fungování je popsán v následující kapitole 3.

Kapitola 3

Popis řešení

1. Simuluje všechny možné vývoje hry (hraje za oba hráče)
2. Předpokládá, že oba hráči volí vždy optimální taktiku
3. Když se dokončí simulovaná hra, vyhodnotí se výsledek (který hráč vyhrál) a propaguje se na všechny předešlé úrovně (na předky)
4. Algoritmus poté ve finále zjistí, k jakému výsledku vedou jednotlivé tahy a zvolí ten, který je výhodnější (pokud takový není, zvolí libovolný)



Obrázek 3.1: Stromová struktura prohledávání

Jak je z obrázku vidět, algoritmus je výpočetně náročný, a proto byla potřeba zajistit urychlení. Při postupu se mohou některé stavy opakovat, zvláště pak u větších úloh, kde se počet opakování znásobuje.

Proto jsem se rozhodl, že budu ukládat stavy s jejich výsledky do slovníku. Slovník má výhodu, že nehledá stavy uvnitř sekvenčně, ale pomocí vypočtené hodnoty (hash). Výsledkem je prohledávání v čase $O(1)$, což samo o sobě také přispívá k efektivitě programu.

Jedno úskalí, které přináší ukládání mezivýsledků, je rozdílná interpretace v závislosti na tom, jestli byl stav vypočten pro minimalizujícího hráče (protihráče) nebo maximalizujícího (ten, pro kterého se počítá optimální tah). Řešení je proto následující:

1. Data jsou uložena vždy vzhledem k maximalizujícímu hráči
 - Vypočítá se, jaký je výsledek za předpokladu, pokud by toto byl výchozí stav pro maximalizujícího hráče (výhra = 1, prohra = -1)
2. Při čtení hodnoty z paměti se kontroluje, jestli hráč není minimalizující - pokud ano, tak uložená hodnota se musí invertovat

Tento jev je zároveň vidět i na obrázku výše, kde například jedna zápalka vyvolává rozdílný výsledek (výhra nebo prohra) v závislosti na úrovni, ve které se nachází.

Jak bylo zmíněno výše, výhra se vyznačuje návratovou hodnotou 1 a prohra -1. Program pak rekurzivně hledá v první úrovni možnost, která poskytne maximální skóre. Následující volání se naopak „hraje za protihráče“ a proto se hledá minimální skóre (protihráč se snaží o prohru svého protivníka). Když se hra dostane do koncového stavu, tak se propaguje výsledek o úroveň výše, kde se dosazuje do funkcí minima, respektive maxima a tímto způsobem se poté vrátí výsledky až ke kořeni stromu (počátečnímu stavu), který pak rozhodne na základě stejné logiky (hledá maximální tah s maximálním ohodnocením), který tah je ten nejlepší.

Kapitola 4

Popis programu

4.1 Volba programovacího jazyka

Pro realizaci programu byl zvolen programovací jazyk Python, protože umožňuje všechny funkce, zajišťuje přenositelnost napříč platformami a byl primárně využíván v rámci předmětu Úvod do znalostního inženýrství.

4.2 Použití knihoven

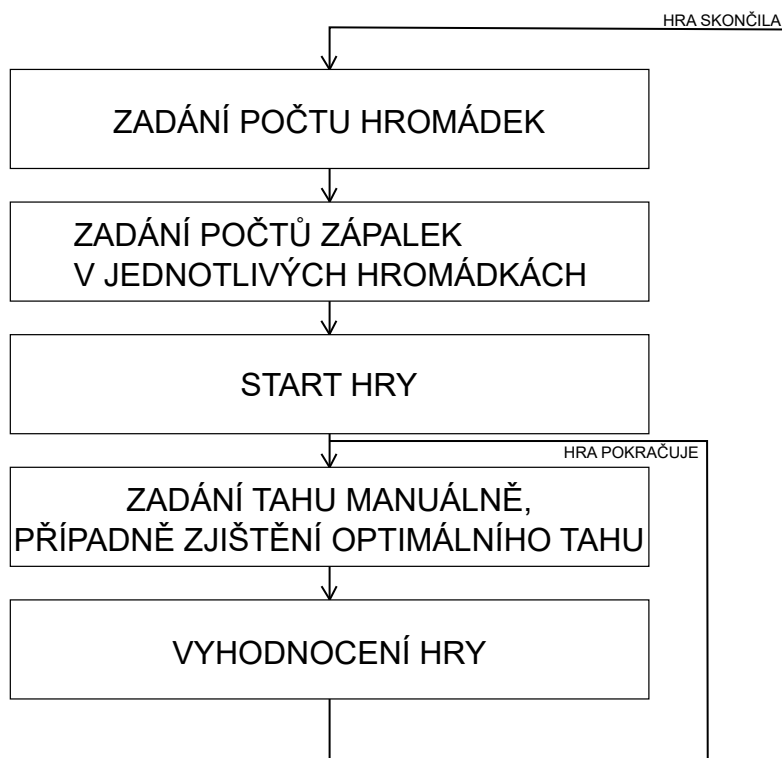
V rámci Pythonu existuje spousta knihoven, z kterých byly použity knihovny níže:

- Enum
 - Již součástí základní instalace
 - Deklarace výčtových typů
- Dict
 - Rovněž již součástí základní instalace
 - Uložení výsledků pro již zpracované stavy hry
 - Zajišťuje efektivitu při kontrole, jestli byl již stav zpracován
- Re
 - Také již součástí základní instalace
 - Ošetření načítání vstupů pomocí regulárních výrazů

4.3 Volba datových struktur

- N-ární strom
 - Nepřímo byl použit graf (konkrétně n-ární strom), který je využit v rámci rekurzivního prohledávání, viz. podkapitola 2.2
- Slovník
 - Pro uložení předešlých výsledků rekurzivního hledání byl využit slovník - výhodou je vyhledávání v $O(1)$
- Seznam
 - Seznam byl použit pro zjištění a uložení platných tahů pro daný stav

4.4 Blokové schéma jednotlivých programových kroků



Obrázek 4.1: Blokové schéma programu

4.5 Programová dokumentace

4.5.1 CommunicationModule

Obsahuje komunikaci s uživatelem zahrnující vstupy a výstupy (kromě vysvětlovacího výpisu, který vypisuje vysvětlovací modul).

Metoda „user_choice“

Metoda slouží k výběru ze seznamu možností. Pro číslici, která je mimo rozsah vybraných hodnot, se vypíše nápověda s minimální a maximální přípustnou hodnotou. Metoda neskončí, dokud uživatel nezadá přípustnou hodnotu.

Název parametru	Význam
attribute_name	Název vybíraného atributu ze seznamu
choices	Možnosti (volby), které jsou na výběr pro uživatele
Návratová hodnota	Zadaná číselná hodnota při výběru

Tabulka 4.1: Tabulka parametrů

Metoda „numeric_input“

Metoda slouží k číselnému vstupu. Po zadání „h“ se vypíše nápověda. Metoda neskončí, dokud uživatel nezadá přípustnou hodnotu.

Název parametru	Význam
attribute_name	Název zadávaného atributu
help_text	Text, který se zobrazí uživateli po stisknutí „h“
Návratová hodnota	Zadaná číselná hodnota

Tabulka 4.2: Tabulka parametrů

Metoda „numeric_input_bounded“

Metoda slouží k číselnému vstupu, který je avšak omezen hranicemi. Po zadání „h“ se vypíše nápověda. Metoda neskončí, dokud uživatel nezadá přípustnou hodnotu.

Název parametru	Význam
attribute_name	Název zadávaného atributu
help_text	Text, který se zobrazí uživateli po stisknutí „h“
lower_bound	Minimální přípustná hodnota „h“
upper_bound	Maximální přípustná hodnota „h“
Návratová hodnota	Zadaná číselná hodnota

Tabulka 4.3: Tabulka parametrů

Metoda „string_input“

Metoda slouží k textovému vstupu. Při zadání textu délky 0 (například odřádkování bez jediného znaku) metoda znovu vynutí zadání nového řetězce.

Název parametru	Význam
attribute_name	Název zadávaného atributu
Návratová hodnota	Zadaný řetězec

Tabulka 4.4: Tabulka parametrů

Metoda „write_output“

Metoda slouží k výpisu informace.

Název parametru	Význam
message	Vypisovaná zpráva

Tabulka 4.5: Tabulka parametrů

4.5.2 ActionType

Výčtová třída obsahující pouze dvě hodnoty reprezentující, z kolika hromádek budou odebrány zápalky

- SINGLE
- DOUBLE

4.5.3 GameState

Třída sloužící k reprezentaci stavu hry.

Název parametru	Význam
piles	Pole celých čísel reprezentující zápalky a hromádky

Tabulka 4.6: Tabulka atributů třídy

Metoda „__eq__“

Metoda umožňuje porovnávání tříd pomocí operátoru ==. Třídy (stavy) jsou stejné, pokud hromádky obsahují stejné počty zápalek neohledně na pořadí (permutaci) hromádek.

Název parametru	Význam
other	Porovnávaná instance třídy GameState
Návratová hodnota	Výsledek porovnání instancí

Tabulka 4.7: Tabulka parametrů

Metoda „__hash__“

Metoda vytvoří hash (hodnotu), která co nejvíce reprezentuje daná data a zároveň je pro ně svým způsobem unikátní. Musí platit, že pro permutovaná data bude hash stejný. Proto se hash odvodí pomocí seřazeného seznamu hromádek podle počtu zápalek.

Název parametru	Význam
piles	Pole celých čísel reprezentující zápalky a hromádky
Návratová hodnota	Hash dané instance

Tabulka 4.8: Tabulka parametrů

Metoda „__str__“

Metoda převádí herní pole do textové pole - vrací text o tom v jaké hromádce je kolik zápalek.

Název parametru	Význam
Návratová hodnota	Stav hry vypsáný v textové podobě

Tabulka 4.9: Tabulka atributů třídy

4.5.4 Move

Třída reprezentující tah.

Název parametru	Význam
action_type	Typ akce, viz. podkapitola 4.5.2

Tabulka 4.10: Tabulka parametrů

Metoda „__str__“

Metoda převádí data v třídě na textovou reprezentaci. Automaticky je zavolána při

Název parametru	Význam
Návratová hodnota	Tah vypsáný v textové podobě

Tabulka 4.11: Tabulka parametrů

4.5.5 Explanation

Vysvětlovací modul, umožňuje výpis vysvětlení.

Metoda „show_explanation“

Vypisuje vysvětlení.

Název parametru	Význam
message	Zpráva s vysvětlením

Tabulka 4.12: Tabulka parametrů

4.5.6 InferenceModule

Třída zajišťuje základní logiku programu včetně zadávání vstupů a hlavní algoritmus - minmax, který hledá optimální tahy.

Název parametru	Význam
knowledge	Třída, která uchovává výsledky již zpracovaných stavů hry

Tabulka 4.13: Tabulka atributů třídy

Metoda „__get_moves“

Metoda počítá a vrací všechny možné tahy pro daný stav hry.

Název parametru	Význam
game_state	Instance třídy reprezentující stav hry
Návratová hodnota	Pole všech validních tahů, které lze provést v daném stavu

Tabulka 4.14: Tabulka parametrů

Metoda „get_piles_occupied“

Metoda počítá počet hromádek, které obsahují alespoň jednu zápalku.

Název parametru	Význam
game_state	Instance třídy reprezentující stav hry
Návratová hodnota	Počet hromádek, které obsahují alespoň jednu zápalku

Tabulka 4.15: Tabulka parametrů

Metoda „game_ended“

Metoda kontroluje, jestli je každá z hromádek prázdná a tím pádem je konec hry.

Název parametru	Význam
game_state	Instance třídy reprezentující stav hry
Návratová hodnota	True, pokud jsou všechny hromádky prázdné, jinak False.

Tabulka 4.16: Tabulka parametrů

Metoda „end_state“

Metoda kontrolující, jestli je předaný stav hry ve výherní pozici. Ta může být dosažena následujícími způsoby.

- Pouze jedna hromádka má nenulový počet zápalek
- Pouze dvě hromádky mají nenulový počet zápalek a navíc počet zápalek v obou hromádkách je stejný

Tato metoda ve výsledku zrychluje prohledávání stromem, protože stavy, které vyhovují těmto podmínkám, nejsou dále děleny a jsou uzavřeny jako konec hry pro účely prohledávání stromu.

Název parametru	Význam
game_state	Instance třídy reprezentující stav hry
Návratová hodnota	True, pokud stav hry vyhovuje alespoň jedné podmínce, jinak False.

Tabulka 4.17: Tabulka parametrů

Metoda „`apply_move`“

Metoda použije (aplikuje) tah na stav hry, který je jí předán společně s tahem jako parametr

Název parametru	Význam
<code>game_state</code>	Instance třídy reprezentující stav hry
<code>move</code>	Instance třídy reprezentující tah

Tabulka 4.18: Tabulka parametrů

Metoda „`revert_move`“

Metoda použije (aplikuje) inverzní tah na stav hry, čímž vrátí hru do původního stavu. Používá se při rekursivním hledání, kdy se aplikuje tah, začne se rekursivně prohledávat strom (potomci), a pak se tah musí vrátit zpátky.

Název parametru	Význam
<code>game_state</code>	Instance třídy reprezentující stav hry
<code>move</code>	Instance třídy reprezentující tah

Tabulka 4.19: Tabulka parametrů

Metoda „`minmax`“

Metoda rekursivně hledá výsledky potomků aktuálního stavu předaného jako parametr. Z nich najde ten, který má pro hráče, za kterého hraje (minimalizující/maximalizující), největší hodnotu a výsledek následně vrátí. Důkladnější popis je v kapitole 3

Název parametru	Význam
<code>game_state</code>	Instance třídy reprezentující stav hry
<code>is_maximizing</code>	Příznak, jestli se jedná o maximalizujícího hráče

Tabulka 4.20: Tabulka parametrů

Metoda „find_best_move“

Metoda hodnoty stavu pomocí metody minmax a vybírá z nich ten, který má největší skóre (je optimální).

Název parametru	Význam
game_state	Instance třídy reprezentující stav hry
Návratová hodnota	Tah, který je ten nejlepší v dané situaci

Tabulka 4.21: Tabulka parametrů

Metoda „inference_game_init“

Metoda, která se ptá na počet hromádek a počty zápalek v jednotlivých hromádkách

Návratová hodnota	Třída popisující stav hry (GameState)
-------------------	---------------------------------------

Tabulka 4.22: Tabulka parametrů

Metoda „get_move_choices“

Metoda zjišťuje, které možnosti může v daném kole navrhnout uživateli jako výběr platných akcí (v některých situacích nelze vybrat ze dvou hromádek současně).

game_state	Třída popisující stav hry (GameState)
Návratová hodnota	Seznam možností uživatelské interakce v jednotlivých kolech

Tabulka 4.23: Tabulka parametrů

Metoda „choose_pile“

Metoda nechá uživatele vybrat libovolnou hromádku s omezením, že hromádka musí obsahovat alespoň jednu zápalku.

game_state	Třída popisující stav hry (GameState)
name	Jméno, použité pro dotaz („Zadejte první/druhou hromádku...“)
Návratová hodnota	Seznam možností uživatelské interakce v jednotlivých kolech

Tabulka 4.24: Tabulka parametrů

Metoda „inference_move“

Metoda se ptá, z jaké/jakých hromádek se má sebrat kolik zápilek a vrací instanci reprezentující tah.

game_state	Třída popisující stav hry (GameState)
Návratová hodnota	Tah, který byl zadán uživatelem.

Tabulka 4.25: Tabulka parametrů

Metoda „start_game“

Metoda zapíná hru a volá ostatní metody, prostřednictvím kterých interaguje s uživatelem. Zároveň vypisuje kdo je na tahu, kdo vyhrál a stav hry.

Metoda „infere“

Metoda spouští start hry. Po ukončení se zeptá uživatele, jestli chce pokračovat v další hře nebo ukončit aplikaci.

4.5.7 Knowledge

Třída zajišťuje ukládání stavů, které již byly zpracovány společně s výsledky.

Název parametru	Význam
knowledge	Slovník uchovávající stavy a výsledky, kte kterým vedou

Tabulka 4.26: Tabulka atributů třídy

Metoda „add_state“

Metoda si uloží stav společně s výsledkem a převede jej vzhledem k maximalizujícímu hráči, viz. kapitola 3.

Název parametru	Význam
state	Stav, který se má uložit
result	Výsledek, ke kterému se došlo
is_maximizing	Jakým způsobem výsledek interpretovat

Tabulka 4.27: Tabulka parametrů

Metoda „contains_state“

Metoda kontroluje, jestli stav už nemá uložený a vrací podle toho True nebo False.

Název parametru	Význam
state	Stav, který se má uložit
Návratová hodnota	True, pokud byl stav již uložen, jinak False

Tabulka 4.28: Tabulka parametrů

Metoda „get_result“

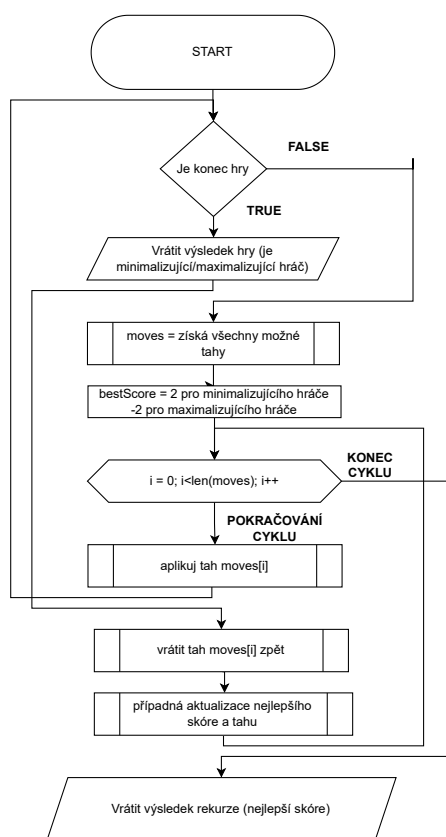
Metoda zjišťuje výsledek pro předaný stav a vrací ho přizpůsobený parametru is_maximizing (interpretuje ho podle uživatele).

Název parametru	Význam
state	Stav, který se má uložit
is_maximizing	Příznak, jak interpretovat výsledek
Návratová hodnota	Skóre, které je interpretované podle příznaku

Tabulka 4.29: Tabulka parametrů

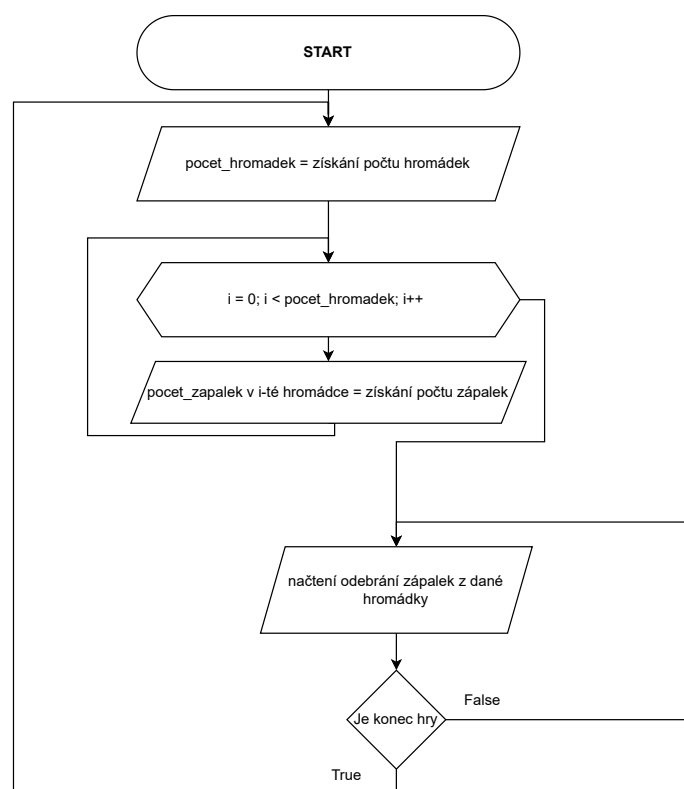
4.5.8 Vývojový diagram hlavní části programu

Níže přikládám vývojový diagram algoritmu minmax, který naznačuje rekurzi a způsob fungování.



Obrázek 4.2: Blokové schéma programu

4.5.9 Obecný vývojový diagram



Obrázek 4.3: Obecné fungování programu

Kapitola 5

Popis obsluhy programu

5.1 Základní nastavení

Po startu aplikace se zobrazí tyto hlášky

```
Zadejte pocet hromadek, pripadne h pro vysvetleni, co delat: 2
Zadejte pocet zapalek pro 1. hromadku, pripadne h pro vysvetleni, co
delat: 1
Zadejte pocet zapalek pro 2. hromadku, pripadne h pro vysvetleni, co
delat: 3
Zadejte jmeno prvniho hrace: Pepa
Zadejte jmeno druhého hrace: Ondra
```

5.2 Hra

Při každém kole je potřeba zvolit volbu, kterou vás program provede, viz. text níže.

Vysvetlení:

Na tahu je hrac: Pepa

Stav hry - hromadky:

Hromadka 1: 1

Hromadka 2: 3

Vyberte si jednu z nabízených možností pro druh tahu zadáním čísla volby:

- 1) Odebrat z jedné hromadky libovolný počet zápalek
- 2) Odebrat ze dvou hromadek stejné množství zápalek
- 3) Vybrat automaticky optimální krok
- 4) Napoveda

Zadejte číslo volby: 2

Následuje například tento výstup, do kterého se doplňují hodnoty, které jsou vyznačeny tučně.

Následuje například tento výstup, do kterého se doplňují hodnoty, které jsou vyznačeny tučně.

Zadejte číslo volby:

Zadejte číslo první hromadky případně h pro vysvětlení, co dělat: 1

Zadejte číslo druhé hromadky případně h pro vysvětlení, co dělat: 2

Zadejte počet zápalek případně h pro vysvětlení, co dělat: 4

Vysvětlení:

Špatný vstup, zadejte celé číslo v rozsahu: 1 až 1.

Zadejte počet zápalek případně h pro vysvětlení, co dělat: 1

Vysvětlení:

Hrac Pepa provedl tah: Vezmete 1 z hromadek 1 a 2.

5.3 Závěr hry

Závěrem hry se vypíše výsledek hry a uživatel si může zvolit jestli pokračovat nebo ukončit aplikaci.

Vyhrál hráč Ondra

Vyberte si jednu z nabízených možností pro funkci aplikace zadáním čísla volby:

- 1) Pokračovat v další hře
- 2) Ukončit aplikaci
- 3) Napoveda

Kapitola 6

Rozbor výsledků a závěr

Výsledky byly vyzkoušeny a program funguje dle předpokladů. Díky bázi znalostí, která obsahuje tabulku pro ukládání výsledků, je program mnohonásobně rychlejší než bez ní. Dalším urychlením pak bylo vyhodnocení už před posledním tahem. Například u dvou hromádek velikosti 8 by se standardně procházelo i řešení odebrání 16 zápalek po jedné, tj. 16 potomků společně s dalšími kombinacemi.

Nedostatky nebyly zjištěny žádné, program funguje správně. Navíc díky obecnému algoritmu minmax se dají snadno upravit pravidla hry a hledat tak pomocí stejného algoritmu (pouze změněných pravidel) optimální strategii i u jiné, upravené verze.