Porovnání konečných automatů pro rozpoznávání podposloupností řetezce

KIV/PRO – Semestrální práce

Student: Ladislav Čákora
Osobní číslo: A23B0149P
Email: cakoral@students.zcu.cz
Datum: 30. listopadu 2025

Obsah

1 Zadání

Cílem této práce je ověření paměťové úspory konstrukce konečného automatu pro rozpoznávání subsekvencí řetězce uvedených v článku *Subsequence Automata with Default Transitions*. Postup ověření bude implementace standartního automatu a automatů uvedných v článku a následné zpracování několika různých řetězců danými automaty. Výsledky se poté porovnají s teoretickými hodnotami.

2 Teoretické pozadí

2.1 Subsekvenční automat (SA)

Klasický subsekvenční automat pro řetězec S délky n má stavy $0, 1, \ldots, n$, kde stav s reprezentuje prefix délky s. Přechod pro znak a ze stavu s míří do nejmenšího $t > s$, kde $S[t] = a$.

SA přijímá řetězec P právě tehdy, pokud je P subsekvencí S.

Velikost SA je $\mathcal{O}(n\sigma)$, kde σ je velikost abecedy.

2.2 Automaty s defaultními přechody

Level automaty využívají defaultní přechod:

- používá se pouze tehdy, když žádný explicitní přechod neodpovídá,
- nespotřebovává aktuální znak,
- zaručuje, že automat zůstává deterministický.

Defaultní přechody umožňují dramaticky zmenšit velikost automatu, za cenu tzv. *delay* – maximálního počtu defaultních skoků před nalezením explicitního přechodu.

2.3 Level automat

Level automat zavádí pro každý stav i *úroveň*:

$$\text{level}(i) = \max\{x \mid i \bmod k^x = 0\},$$

přičemž

$$\text{level}(i) \leq \lceil \log_k(\sigma) \rceil.$$

Výsledná velikost automatu je:

$$\mathcal{O}(nk \log_k \sigma)$$

a zpoždění (delay) je

$$\mathcal{O}(\log_k \sigma).$$

2.4 Alphabet aware automat

Tato varianta zohledňuje specifickou strukturu abecedy a dále redukuje velikost:

$$\text{size} = \mathcal{O}(n \log \sigma), \qquad \text{delay} = \mathcal{O}(\log \sigma).$$

Jedná se o variaci level automatu s $k = 2$ a optimalizací při velkém intervalu mezi stavy.

3 Architektura programu

Program je rozdělen do několika modulů:

`SubsequenceAutomaton.py` Abstraktní základní třída.

`GeneralAutomaton.py` Implementace klasického SA.

`LevelAutomaton.py` Implementace level automatu s parametrem k.

`AlphabetAwareAutomaton.py` Implementace alphabet-aware automatu.

`AutomatonData.py` Datová struktura pro statistiky.

`main.py` Hlavní spouštěcí skript.

Každý automat má:

- přechodovou tabulku reprezentovanou seznamem slovníků,
- metodu `Compute()` pro vyhodnocení subsekvence,
- metodu `GetInfo()` pro získání statistik: počet stavů, přechodů, podíl defaultních přechodů, úsporu oproti SA.

3.1 Popis modulů

3.1.1 SubsequenceAutomaton

Abstraktní třída definující rozhraní:

- `Compute(inputStr)` – rozhoduje, zda je vstup přijat,
- `GetInfo()` – vrací objekt se statistikami.

Slouží jako rodič pro všechny implementace.

3.1.2 GeneralAutomaton

Implementuje klasický subsekvenční automat. Při zpětném průchodu řetězcem udržuje pro každý znak nejbližší další výskyt. Automat ale kompletně neodpovídá definici, jelikož nemá všude definované přechody pro všechny znaky. V konstrukci chybí stav, do kterého by automat spadl v případě neplatného vstupu. Tento stav jsem se rozhodl odstranit, aby počet stavů mezi všemi automaty zůstal stejný. Proto je i u klasického automatu nenulová úspora, přesně $2 \times \sigma$ hran a jeden stav.

Výhody:

- žádné defaultní přechody,
- jednoduché a rychlé vyhodnocování.

Nevýhody:

- velikost $\mathcal{O}(n\sigma)$,
- nevhodné pro velké abecedy.

3.1.3 LevelAutomaton

Parametr k umožňuje plynulé ladění mezi velikostí a zpožděním. Při konstrukci se automat staví pozpátku, udržuje se:

- nejbližší stav vyšší úrovně,
- nejbližší výskyt každého znaku.

Výhodou je dramaticky menší velikost pro rozumně velké k.

3.1.4 AlphabetAwareLevelAutomaton

Varianta optimalizovaná pro malé abecedy. Při velkém intervalu mezi stavy se přidají přechody pro celou abecedu, což snižuje celkovou úroveň a zpoždění.

3.2 Hlavní skript `main.py`

Skript:

1. načte vstupní textový soubor,
2. vygeneruje:
   - validní subsekvenci (výběr vzestupných indexů),
   - nevalidní subsekvenci (znaky mimo abecedu nebo nadlimitní multiplicita),
3. vytvoří:
   - GeneralAutomaton,
   - LevelAutomaton pro $k = 2, 3, 5, 50$,

- AlphabetAwareLevelAutomaton,

4. otestuje přijímání subsekvencí,
5. vypíše statistiky.

Statistiky zahrnují:

- počet stavů,
- počet přechodů,
- podíl defaultních přechodů,
- úsporu proti plnému SA.

3.3 Problémy a jejich řešení

3.3.1 Reprezentace přechodové funkce automatu

Prvním problémem byl výběr vhodné implementace přechodové funkce automatu. Jako řešní se nabízely dvě možnosti:

- matice přechodu, kde sloupce odpovídají vstupním symbolům, řádky stavům a hodnoty cílovému stavu daného přechodu
- seznam přechodových tabulek pro každý stav

První možnost by byla implementována dvourozměrným polem. Tato volba by byla vhodná pro standartní automat, který má přechod pro každý symbol u každého stavu. Všechny testované automaty ale budou přechody vynechávat, což by se v této situaci projevilo pouze prázdnými místy v matici a nepřineslo žádnou úsporu, naopak algoritmus pro stavbu a vyhodnocení automatu by byl výrazně složitější. Proto byl využit druhý přístup, který tento problém řeší vynecháním záznamů v přechodové tabulce. Takto implementovaná přechodová funkce využívá pole přechodových tabulek, které jsou implemntovány slovníkem, kde klíč je vstupní symbol a hodnota je index cílového stavu, a tedy i přechodové tabulky pro cílový stav.

3.3.2 Stavba automatu

Konstrukce běžného automatu probíhá procházením původního řetězce od konce a využívá slovníku posledních výskytů znaků, kde se ukládají nejbližší pozdější výskyty znaků. Pro každý znak se provede:

1. na konec pole přechodové tabulky se přidá kopie slovníku posledních výskytů
2. ve slovníku se upraví hodnota pro aktuálně zpracovaný znak

Na konec se to samé vykoná i pro stav odpovídající prázdnému podřetězci a výsledné pole slovníků se otočí, aby pořadím odpovídalo původnímu řetězci. Tato implementace naznačuje problém neefektivity tohoto přístupu, jelikož mezi po sobě jdoucími tabulkami je změna v pouze jedné hodnotě.

Konstrukce ostatních automatů začíná vypočítáním úrovní pro jednotlivé stavy. Algorimus opět využívá slovníku s posledními výskyty a zároveň pole posledních výskytů úrovní, kde index odpovídá úrovni a hodnota poslednímu výskytu. Pro každý znak se tedy vykoná:

- vybere se defaultní přechod podle pravidel v článku
- do tabulky se přidají přechody přeskočené defaultním přechodem
- tabulka se přidá na konec pole
- aktualizují se poslední výskyty znaků a úrovní

Na konec se to samé vykoná i pro stav odpovídající prázdnému podřetězci a výsledné pole slovníků se otočí, aby pořadím odpovídalo původnímu řetězci. Tento algoritmus je stejný pro Level i Alphabet-aware automaty, liší se pouze ve výpočtu úrovní.

3.4 Spuštění programu

Pro spuštění je třeba mít nainstalované běhové prostředí jazyka Python. Vstupní bod programu je soubor "src/main.py". Program přijímá jeden parametr, cestu k textovému souboru se vstupním textem. Soubory použité k testování jsou ve složce "testFiles".

4 Měření

4.1 Vstupní data

- speech.txt - výňatek z knihy Echoes of Eternity (ISBN: 9781800266216)
- ascii.txt - náhodný řetězec znaků ascii abecedy o délce 10 000 znaků
- dna.txt - sekvence DNA o délce 100 000 znaků
- dnaShort.txt - sekvence DNA o délce 10 000 znaků
- short.txt - náhodný řetězec znaků anglické abecedy o délce 20 znaků
- code.c - zdrojový kód části semestrální práce z předmětu KIV/PC
- a.txt - 10 000x znak "a"
- lorem.txt - Lorem ipsum o délce zhruba 10 000 znaků

4.2 Výsledky

Dataset	Automat	Stavů	Hran	Výchozí	Explicitní	Výchozí %	Ušetřeno %	Validní OK
Krátký text (20 znaků, abeceda 10)								
Short	SA	21	126	0	126	0%	42.73%	Ano
Short	Level k=2	21	55	19	36	34.55%	75.00%	Ano
Short	Level k=3	21	63	17	46	26.98%	71.36%	Ano
Short	Level k=5	21	75	17	58	22.67%	65.91%	Ano
Short	Level k=50	21	123	0	123	0%	44.09%	Ano
Short	Alphabet-Aware	21	57	19	38	33.33%	74.09%	Ano
ASCII text (10000 znaků, abeceda 256)								
ASCII	SA	10001	2 495 588	0	2 495 588	0%	2.536%	Ano
ASCII	Level k=2	10001	57 431	9 961	47 470	17.34%	97.757%	Ano
ASCII	Level k=3	10001	64 993	9 983	55 010	15.36%	97.462%	Ano
ASCII	Level k=5	10001	81 493	9 985	71 508	12.25%	96.817%	Ano
ASCII	Level k=50	10001	287 133	9 997	277 136	3.48%	88.786%	Ano
ASCII	Alphabet-Aware	10001	57 439	9 961	47 478	17.34%	97.757%	Ano
Programový kód (7585 znaků, abeceda 80)								
Code	SA	7586	518 163	0	518 163	0%	14.630%	Ano
Code	Level k=2	7586	30 903	7 525	23 378	24.35%	94.909%	Ano
Code	Level k=3	7586	35 986	7 487	28 499	20.81%	94.071%	Ano
Code	Level k=5	7586	41 071	7 521	33 550	18.31%	93.233%	Ano
Code	Level k=50	7586	118 017	7 547	110 470	6.39%	80.556%	Ano
Code	Alphabet-Aware	7586	30 907	7 525	23 382	24.35%	94.908%	Ano
Jednopísmenný text (10000 znaků, abeceda 1)								
a-only	SA	10001	10 000	0	10 000	0%	0.01%	Ano
a-only	Level k=2	10001	9 999	0	9 999	0%	0.02%	Ano
a-only	Level k=3	10001	9 999	0	9 999	0%	0.02%	Ano
a-only	Level k=5	10001	9 999	0	9 999	0%	0.02%	Ano
a-only	Level k=50	10001	9 999	0	9 999	0%	0.02%	Ano
a-only	Alphabet-Aware	10001	10 000	0	10 000	0%	0.01%	Ano
DNA sekvence – krátká (10000 znaků, abeceda 4)								
DNA-short	SA	10001	39 993	0	39 993	0%	0.038%	Ano
DNA-short	Level k=2	10001	26 879	7 501	19 378	27.91%	32.816%	Ano
DNA-short	Level k=3	10001	28 778	8 889	19 889	30.89%	28.069%	Ano
DNA-short	Level k=5	10001	31 695	8 001	23 694	25.24%	20.778%	Ano
DNA-short	Level k=50	10001	47 420	9 801	37 619	20.67%	-18.526%	Ano
DNA-short	Alphabet-Aware	10001	26 882	7 501	19 381	27.90%	32.808%	Ano
DNA sekvence – dlouhá (100000 znaků, abeceda 4)								
DNA	SA	100001	399 989	0	399 989	0%	0.0047%	Ano
DNA	Level k=2	100001	268 779	75 001	193 778	27.90%	32.807%	Ano
DNA	Level k=3	100001	287 335	88 889	198 446	30.94%	28.168%	Ano
DNA	Level k=5	100001	316 057	80 001	236 056	25.31%	20.987%	Ano
DNA	Level k=50	100001	474 194	98 001	376 193	20.67%	-18.546%	Ano
DNA	Alphabet-Aware	100001	268 783	75 001	193 782	27.90%	32.806%	Ano
Knižní text (6930 znaků, abeceda 55)								
Speech	SA	6931	328 457	0	328 457	0%	13.850%	Ano
Speech	Level k=2	6931	29 122	6 821	22 301	23.42%	92.362%	Ano
Speech	Level k=3	6931	32 001	6 843	25 158	21.38%	91.607%	Ano
Speech	Level k=5	6931	37 232	6 873	30 359	18.46%	90.235%	Ano
Speech	Level k=50	6931	98 702	6 861	91 841	6.95%	74.112%	Ano
Speech	Alphabet-Aware	6931	29 129	6 821	22 308	23.42%	92.360%	Ano
Lorem text (10011 znaků, abeceda 41)								
Lorem	SA	10012	390 378	0	390 378	0%	4.9095%	Ano
Lorem	Level k=2	10012	40 606	9 852	30 754	24.26%	90.109%	Ano
Lorem	Level k=3	10012	44 883	9 885	34 998	22.02%	89.067%	Ano
Lorem	Level k=5	10012	52 447	9 929	42 518	18.93%	87.225%	Ano
Lorem	Level k=50	10012	138 563	9 801	128 762	7.07%	66.248%	Ano
Lorem	Alphabet-Aware	10012	40 610	9 852	30 758	24.26%	90.108%	Ano

Tabulka 1: Výsledky subsekvenčních automatů pro všechny datasety.

5 Závěr

Program implementuje tři různé třídy subsekvenčních automatů, které se liší velikostí a paměťovou náročností.

Z výsledků je patrných několik poznatků. Nejvýznamějším je vliv velikosti abecedy a délky řetězce. Jak je vidět z případu sekvencí DNA, procentuální úspora u obou je efektivně stejná. Naopak velikost abecedy má vliv velký, jak ukazuje porovnání všech 10 000 znaků dlouhých datasetů. U jednoznakové abecedy byla úspora téměř nulová, u 4 znaků už byla přes 30%. U větších abeced už úspora dosáhla 90%. Toto tvrzení podporuje i porovnání zdrojového kódu s knižním textem. I když je zdrojový kód zhruba o 500 znaků delší a běžný SA má o necelých 200 000 hran více, větší abeceda umožnila lepší kompresi a v případě Alphabet-aware automatu nebo Level(k=2) automatu se rozdíl zmenšil na necelé 2 000.

Dalším zajímavým poznatkem je podobnost výsledků mezi Level automatem pro $k = 2$ a Alphabet-aware automatem, kde se úspora často liší až na druhém nebo třetím desetinném místě a to vždy ve prospěch Level automatu. Zdůvodnil bych to tím, že konstrukce obou automatů je podobná. Vyšší hodnoty parametru k již úsporu snižují, jelikož mezi stavy s vyšší úrovní, do kterých by vedly "úsporné"defaultní přechody, je více stavů, které si musí udržovat explicitní přechody mezi sebou a zároveň si všechny musí udržovat daný defaultní přechod.

Použitá literatura

[1] BILLE, P., GØRTZ, I. L., SKJOLDJENSEN, F. R. *Subsequence Automata with Default Transitions.* Journal of Discrete Algorithms, 42 (2017), 48–55. Elsevier. DOI: 10.1016/j.jda.2017.01.005