Sekáček

Programátorská dokumentace

Co je Sekáček?

Sekáček je program na dělení českého textu na slabiky. Slova dělí podle jejich struktury a podle vzájemných písmen, z kterých se skládá. Sekáček zná nějaká pravidla, podle kterých to lze provádět. Problematická slova, na která pravidla neplatí, porovnává s jejich databází a dělí jinak. Ta databáze bude zřejmě vždy neúplná, je na uživateli, aby si ji doplnil.

Co patří k Sekáčku?

K Sekáčku patří tyto soubory:
- `sekacek.py` – zdrojový kód
- `.sekacek` – původní soubor výjimek, který Sekáček otvírá otvírá při každém spuštění
- `dokumentace.pdf` – uživatelská dokumentace. Je v ní popsáno ovládání Sekáčku a jeho možné využití, doporučuji k pročtení dříve, než začnete číst tuto dokumentaci.
- `dokumentace-programatorska.pdf` – tato programátorská dokumentace
- vzorové vstupy: `Knuth.txt`, `objev.txt`
- vzorové soubory výjimek `vyjimky_Knuth.txt`, `vyjimky_chyba1_Knuth.txt`, `vyjimky_chyba2_Knuth.txt`.

V čem je napsán?

Sekáček je napsán v Pythonu 3.[1] Tento skriptovací jazyk jsem vybral z několika důvodů. Jednak proto, že ho ovládám na dostatečné úrovni, proto, že má mnoho funkcí a knihoven jako například regulární výrazy, které mohu použít a nemusím je psát sám, dále je velmi stručný, je multiplatformní, jednoduchý a intuitivní, o spoustu věcí se programátor nemusí starat. V Pythonu lze napsat například výraz „`0 <= x <= 10`“, to v jiných jazycích takto snadno nelze.

Nevýhodou Pythonu je, že potřebuje interpretr a není proto tak rychlý jako kompilované programovací jazyky. Účelem Sekáčku ale není dělit slova na slabiky nejrychleji, jak se to dá s použitím počítače zvládnout. Cílem je napsat program tak, aby fungoval a byl použitelný pro rozumně velké vstupy. To Sekáček je.

Celý zdrojový kód Sekáčku je soubor sekacek.py. Má dvě hlavní části. Prvních zhruba 140 řádků jsou funkce, které zajišťují hlavní práci programu, samotné sekání na slabiky. Zbytek programu, zhruba 150 řádků, se stará o načtení vstupu, kontrolu jeho správnosti a o komunikaci s uživatelem.

Poznámka k diakritice

Programátora, který si otevře sekacek.py, na první pohled zarazí funkce s názvem *rozliš.* Není zvykem pojmenovávat funkce a proměnné s diakritikou, přestože Python 3 to umožňuje. Sekáček je ale program na sekání českého textu, proto jsem zvolil názvy v českém jazyce. Nepředpokládám, že by si jeho zdrojový kód prohlížel a chtěl pochopit někdo, kdo neumí česky a neumí si správně zobrazit soubor v UTF-8. Podobná klíčová slova jako *sténá* a *stěna,* která

[1] `https://www.python.org/`

by se lišila pouze v diakritice a ten rozdíl by tedy byl snadno přehlédnutelný, nepoužívám, takže doufám, že tato volba nikomu neublíží.

Použitý algoritmus Sekáčku

Sekáček využívá postup pro strojové dělení slov na slabiky, který vychází z článku Dělení slov v češtině pomocí strojů z časopisu Naše řeč[2].

Pravidla pro strojové dělení slov na slabiky

Slova v textu se skládají z grafémů (jinak řečeno písmen), dají se rozdělit podle významu v konkrétním slově na vokály a konzonanty.

Vokály jsou samohlásky, a, e, i, y, o, u, á, é, ě, í, ý, ó, ů, ú. Budu je značit znakem V.

Konzonanty, K, jsou souhlásky, b, c, č, d, ď, f, g, h, ch, j, k, l, m, ň, p, q, r, ř, s, š, t, ť, v, w, x, z, ž.

V článku jsou ještě další skupiny grafémů, slabičné konzonanty, což jsou slabikotvorná l a r, souhláskové grafémy, což jsou souhlásky kromě slabikotvorných l a r a dvojhlásky ou, eu, a ještě další skupiny. Ty ale umím převést na tyto dvě, takže je prozatím nebudu uvažovat.

I. skupiny KV

a) Pokud se vyskytuje před V jeden souhláskový grafém K, připojuje se k následujícímu vokálu a tvoří tak jednu slabiku, to je upřednostňuje se otevřená slabika. Například kolo, KV/KV.

II. skupiny souhlásek

a) Skupina K na začátku slova se připojuje celá k následující slabice.
b) Pokud na konci slova následuje za posledním vokálem více souhlásek, připojují se všechny k předchozí samohlásce, např. ve/li/kost.
c) Stojí-li skupina K uprostřed slova, připojuje se k následující samohlásce poslední souhláska skupiny.

III. nedělitelné skupiny

Některé skupiny souhlásek jsou nedělitelné, vždy jsou součástí jedné slabiky. Ty se pro Sekáček vyjádří, jako by byly jen jedno písmeno, pak se aplikují ostatní pravidla.

Tyto skupiny jsou

a) Souhláskové digramy: sp, st, th, Kr, Kl, Kř (za K je zde jakýkoliv konzonant kromě l, r, ř), Kv.
b) Souhláskové trigramy: str, stř, skv
c) Nedělitelné digramy poziční, ty jsou nedělitelné pouze v případě, že se před nimi nachází souhláska: sk a št, například náchod(sk)ý, arab(št)ina.
   Sám jsem přidal jsem ještě skupiny zd a zdn, v článku zmíněny nejsou.

IV. skupiny samohlásek

a) Skupiny au, ou, ai, oi, považuji za nedělitelné, k tomu ještě eu, pokud se vykytuje na začátku slova.
b) Ostatní výskyty dvou samohlásek vedle sebe se vždy oddělují.

[2] Dostupný na Internetu zde: `http://nase-rec.ujc.cas.cz/archiv.php?art=5348`

Výjimky

Existují slova, která se podle těchto pravidel nebudou dělit správně, například pododdělení se podle pravidla I má dělit po/dod/dě/le/ní. Tato slova tvoří výjimky a je třeba je ošetřit zvlášť.

Jak funguje Sekáček?

Jak Sekáček rozdělí slovo na slabiky?

Stará se o to funkce *sekejslovo.* Udělá si ze slova masku, k tomu je funkce *rozliš.* Maska je řetězec stejné délky jako slovo, v kterém jsou ale znaky K na těch místech, kde jsou ve slově souhlásky, V jsou na místech samohlásek. Na místě znaků, které se mají připojovat k jiným (nedělitelné skupiny, „h“ v písmenu „ch“) je znak 0. Například slovo chobot má masku K0VKVK.

Slabikotvorná r a l mají značku V, přistupuje se k nim jako k samohláskám. Funkce *rozliš* je umí rozeznat od obyčejných, neslabikotvorných.

Když je maska vytvořená, naseká se podle pravidel popsaných výše, přesněji řečeno na hranici slabik do masky vloží „/“. Dělá to funkce *sekejmasku.* Znaky 0 v masce přitom přeskakuje.

Poté se spárují znaky z nasekané masky se znaky z původního slova a na správné pozice se vloží oddělovač slabik. Toto slovo se pak vypíše.

Málokdy ale Sekáček dostane na vstupu samostatné slovo, ve specifikaci je, že na vstupu může být český text. Ten může být složený z více slov oddělených mezerami, konci řádku, tečkami, čárkami, uvozovkami a podobně. Funkce *sekejtext* tedy text rozdělí na slova, na to si zavolá funkci *oddělslova.* Od ní dostane seznam, ve kterém je každý prvek řetězec. Ty prvky, které obsahují písmena (přesněji řečeno vyhovují regulérnímu výrazu `\w`), naseká, protože to jsou slova, ostatní jenom vypíše v původním pořadí mezi ně.

Ještě zbývá popsat, jak Sekáček ošetřuje výjimky. Poté, co *sekejslova* získá seznam se samostatnými slovy, zavolá na každé slovo *rozlišvýjimkyazbytek.*

Tato funkce porovná slovo s každým ze seznamu výjimek. Nalezenou shodu ohraničí z obou stran pomocným znakem 0x4885 a za tento znak dá ještě jeden 0x8906. Podle toho prvního znaku rozdělí slovo na samostatná slova. Ten druhý znak je značka, že toto slovo je výjimka. Funkce *sekejtext* na něj pak nebude volat *sekejslovo,* ale *zpracujvýjimky.*

Ty dva znaky jsou náhodné čínské znaky, o kterých předpokládám, že se na vstupu nikdy nevyskytnou. Pokud je nějaký uživatel zadá, Sekáček nemusí fungovat správně. Předpokládám ale, že pokud se na vstupu vyskytnou, nebude se jednat o český text, ale cizojazyčný, a ten Sekáček správně neseká tak jako tak.

Univerzálně správným řešením by bylo projít vstup a najít nějaké dva znaky z celého Unicodu, které se na vstupu nevyskytují, a ty použít. Tento postup jsem ale nepoužil, protože pak by byl Sekáček příliš komplikovaný.

Stejně tak to, že *rozlišvýjimkyazbytek* porovnává každé slovo ze seznamu výjimek s každým, není optimální. To má příliš velkou asymptotickou složitost, lepší by bylo použít nějaké rychlejší algoritmy či datové struktury, to by ale bylo velmi složité a o to v tomto programu nejde.

Další věc, kterou Sekáček ošetřuje, ale ještě jsem ji nepopsal, jsou neslabičná slova. Ta patří k nějaké slabice, ale jsou oddělená mezerami. Jedná se konkrétně o předložky k, s, v a z. Ty Sekáček připojí k následujímu slovu ještě ve funkci *sekejtext* před čímkoliv jiným.

Komunikace s uživatelem

Jak Sekáček komunikuje s uživatelem je popsáno v uživatelské dokumentaci.

Časová a paměťová složitost

Počet slov textu, který má být nasekán, označím N, počet slov v souborech výjimek je V. Předpokládám, že každé slovo má nějaký malý počet písmen, řádově zhruba 20, jak to v českých textech bývá, a tento počet tedy mohu považovat za konstantu.

Časová složitost Sekáčku je pak $\mathcal{O}(N)$ na rozdělení na slova, $\mathcal{O}(NV)$ na hledání výjimek, $\mathcal{O}(V^2)$ na sekání výjimek a $O(N)$ na sekání obyčejných slov. Celková časová složitost je tedy $\mathcal{O}(NV + V^2)$, šla by ale zlepšit až na $\mathcal{O}(NV)$ a s použitím nějakých rychlých stringových algoritmů snad až na $\mathcal{O}(N + V)$.

Možná vylepšení do dalších verzí

- exe pro Windows – Sekáček spolehlivě funguje jen na Linuxu, skripty v Pythonu 3 ale mohou běžet i na ostatních operačních systémech. Existují programy jako py2exe, které z pythonovského skriptu udělají exe soubor pro Windows, takže na spuštění není potřeba interpretr. Takový program, který by pracoval s Pythonem 3 jsem ale dosud nenašel.
- GUI
- nastavení přístupu k dělení – u některých slov je možno dělit by/strý i bys/trý, Sekáček by mohl zvládat oba přístupy a dát uživateli na výběr
- počítání slabik – nástroj sc, syllable count, který vypíše počet slabik v textu
- náhodné básničky – S pomocí Sekáčku se dá vytvořit seznam všech českých slabik, z těchto slabik by se pak daly skládat básničky, které budou mít naprosto přesný rytmus a rým podle zadaného vzoru, ale budou bez významu z náhodných slabik.

O autorovi

Sekáček vznikl jako zápočtový program předmětu Programování I na Matematicko-fyzikální fakultě UK v Praze v zimním semestru 2013/14.

Autorem je Dominik Macháček. Kontakt na autora: gldkslfmsd@gmail.com.