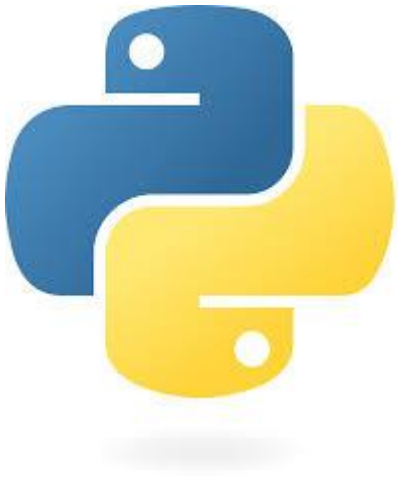
Dokumentace programu HledačSouborů.py

Zápočtový program



zimní semestr 2024/2025

Obsah

Anotace

Program umí rychle vyhledávat soubory. Dopředu si projde disk a vytvoří si index, podle kterého je schopen vyhledávat rychleji než tradiční programy typu file manager. Vyhledávat umí podle jména nebo data úpravy souboru.

Přesné zadání

Vytvořit jednoduchý skript, který bude hledat soubory na disku podle zvolených kategorií jako například datum poslední úpravy souboru nebo jména souboru. Čas od času si sám prohlédne disk a do externího souboru si uloží potřebné informace, aby následné vyhledávání bylo rychlejší.

1. Datové struktury

1.1. Názvy souborů

Program si dopředu projde disk a vytvoří si index, podle kterého je schopen vyhledávat rychle. Cesty k souborům si ukládá do pole. Jména souborů si ukládá do datové struktury trie (písmenkový strom), kde koncový vrchol má hodnotu s indexem do dříve zmíněného pole. Prvek v poli odpovídá adresáři, kde je soubor uložen.

Příklad: C:\ProgramFiles\photos\kočka

[C:\ProgramFiles,

.

.

.

C:\ProgramFiles\mydir,

C:\ProgramFiles\photos ← pole[76]

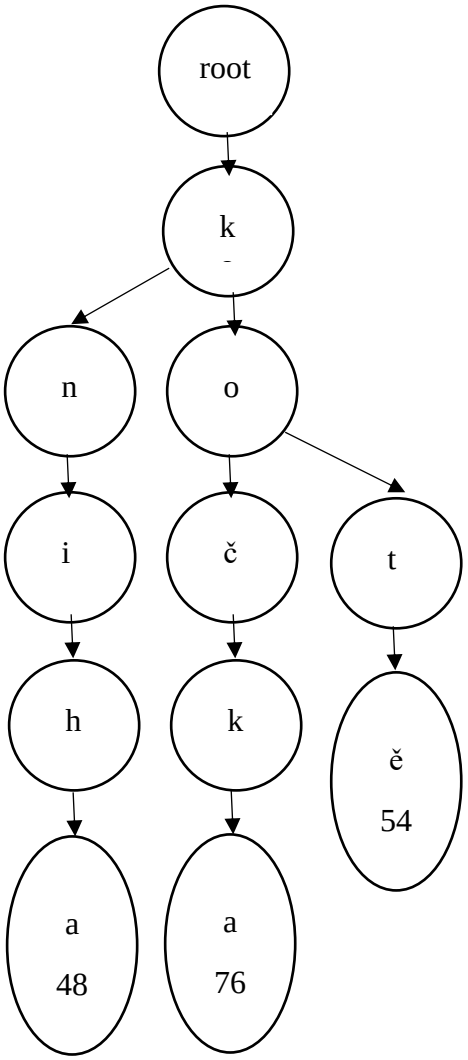
C:\Users\karel,

.

.

.

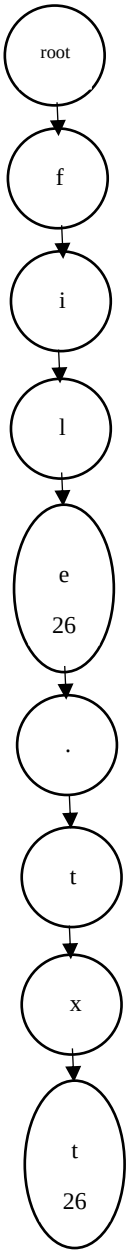
C:\ProgramFiles\photos]



Každý uzel má proměnou pole dlouhou 128, ve které jsou odkazy na následující existující uzly nebo na hodnotu None. Charakter následujícího uzlu je převeden přes ASCII kódování na číslo, které odpovídá indexu v poli. Při hledání slova v trie se stačí postupně dívat na indexy znaků ve slově a procházet s nimi uzly. Pokud taková cesta neexistuje a narazí se na hodnotu None, tak slovo v trie není. Vyhledávání není závislé na délce vstupu, ale na délce slova. Navíc pokud začínají dvě slova na stejnou předponu, tak se část cesty k nim ukládá do stejných uzlů a šetří se tím místo.

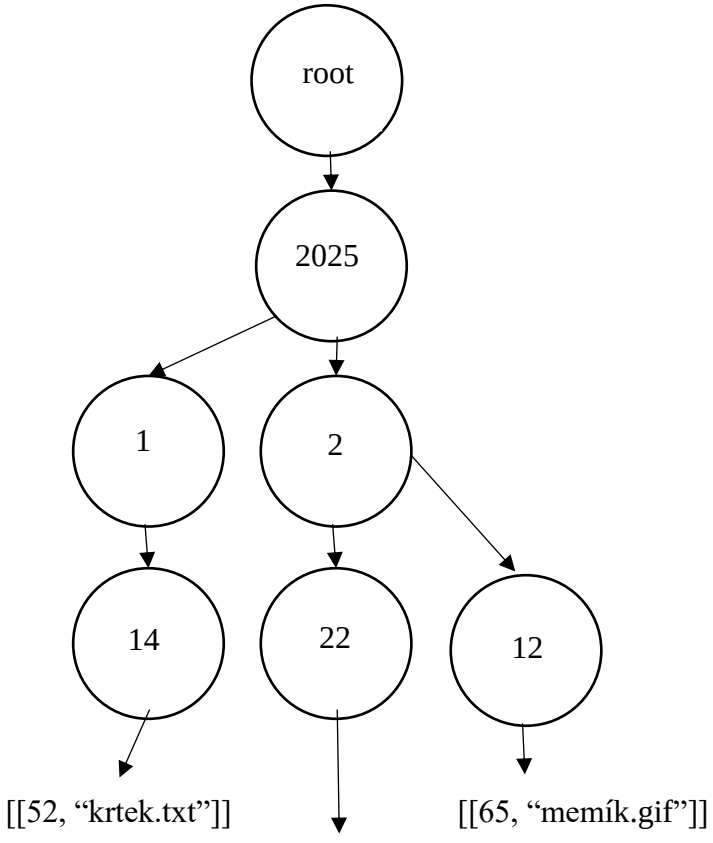
Index do listu s cestami k souborům se ukládá do trie dvakrát jednou s příponou a podruhé bez přípony. Soubor se tak může vyhledat jen pomocí jeho názvu a bez přípony.

Příklad: file.txt se uloží jako dvakrát file a file.txt



1.2. Data modifikací

Pro ukládání času modifikace souboru vyžívám stromovou strukturu. Strom má tři hladiny a ukládá se pouze den, měsíc a rok. V první hladině jsou uzly s hodnotami roků. Ty ukazují na uzly s hodnotami měsíců a nápodobně na uzly s hodnotami dnů. Při procházení stromu se program dívá na všechny odkazy na nižší hladinu a porovnává se s hledanou hodnotou. Celkový počet uzlů však není velký. Počet uložených let bude v praxi maximálně deset, uzlů s měsíci může být až dvanáct a uzly s dny maximálně třicet jedna. Tedy v nejhorším případě 53 porovnání. Uzly na třetí hladině obsahují list, kam se ukládají informace o souborech modifikovaných v odpovídajících datech. To jest jméno souboru a index do listu s cestami k souborům.



2. Vstupní a výstupní data

2.1 Komunikace s programem

Komunikace s programem se vede pomocí konsole, kam se zadávají příkazy. Příkazy lze psát pouze pokud je na to program připravený. To se pozná tak, že vypíše řádek `Ready for new command`. Pokud poslední vypsaný řádek není `Ready for new command`, tak program hledá soubory na disku nebo nahrává externí soubor na vyhledávání a na nové příkazy nebude nereagovat.

Některé příkazy potřebují i jeden dodatečný argument, tedy slovo, které specifikuje chování programu. Od příkazu se oddělují mezerou a mají pevně určený formát, ve kterém se musí zadávat.

Příklad: `find HledačSouborů.py` Slovo find je příkaz a HledačSouborů.py je argument, který říká, že chci vyhledávat soubor *HledačSouborů.py*.

2.2. Vytvoření indexu na vyhledávání

Index je potřebný pro vyhledávání souborů. Vytvoří se zadáním příkazu `crtable` s cestou ke složce, kterou chci prohledávat. Pokud cesta není správná program vypíše `Directory is not valid` a příkaz se neprovede. Po zadání správné složky program prohledá disk a vytvoří dodatečný binární soubor *VyhledávacíData*, který umožňuje hledání souborů.

Příklad: `crtable C:\` prohledá disk C

2.3 Hledání souboru

Hledání lze provést pouze tehdy pokud program má nahraný soubor `VyhledávacíData` nebo se dříve provedl příkaz `crtable`.

2.3.1 Hledání souboru názvem

Příkaz `find` slouží k vyhledávání souborů pomocí jména. Jako argument se použije jméno souboru, který chci vyhledávat. Jméno souboru mohu zadat ve dvou formách, s příponou nebo bez přípony.

Příklad: Mohu použít `find krtek.txt` nebo `find krtek`.

Pokud při hledání *krtek.txt* nic nenajdu, tak určitě vím, že *krtek* také nenajdu. Program si totiž ukládá obě formy a buďto tam jsou obě nebo žádná. Jediný rozdíl je takový, že vyhledávání bez přípony nemusí vyhledat soubor, který požadujeme. Dva soubory mohou mít stejné jméno, ale jinou příponu.

Příklad: Místo souboru *krtek.txt* mohu najít *krtek.png*.

Odpověď však není nesprávná, pokud chci najít požadovaný soubor musím ho lépe specifikovat. Tato funkčnost na vyhledávání souborů bez přípony usnadňuje vyhledávání většiny souborů, protože jména bývají unikátní.

Po úspěšném nalezení souboru program vypíše absolutní cestu, jak se k souboru dostat.

2.3.2 Hledání souboru datem poslední modifikace

Příkazem `fdate` lze vyhledávat soubory skrze datum poslední modifikace souboru. Datum modifikace se napíše jako argument příkazu, a to v pevně daném formátu D/M/R, kde D jsou dny, M roky a R roky.

Příklad: `fdate 22/2/2025`. Hledám soubory s datem modifikace 22.února 2025

Jako výstup jsou cesty ke všem souborům, které byly posledně modifikované na zadaném datu.

2.4 Kořenový adresář hledání

Příkaz `root` vypíše složku, ve které hledání začalo, pokud tedy nějaké proběhlo.

2.5 Nastavení hloubky hledání

Příkazem `depth` lze nastavit hloubku hledání. Argument je celé číslo. Hloubka 0 znamená, že se soubory budou hledat pouze ve zvolené složce. Hloubka 1 znamená, že program prohledá i složky v této složce atd. Výchozí nastavení je 5.

2.6 Ostatní příkazy

Příkaz `help` vypíše uživateli informace o dostupných příkazech.

Příkaz `kill` ukončí program.

3.Diskuse výběru algoritmu

Datovou strukturu trie jsem zvolil, protože je vhodná pro ukládání slov, a navíc se v ní rychle hledá. Původně jsem přemýšlel o ukládání souborů pozpátku, protože by pak výsledná struktura měla méně uzlů. Více souborů sdílí stejnou příponu (.txt, .png), a proto by pak měly stejnou část cesty. V tomto řešení uživatel musí znát i příponu souboru, a to není vždy praktické.

Ukládání dat modifikací jsem chtěl vyřešit pouze přes pole, ve kterém bych binárně vyhledával. Potencionálně bych soubor našel rychleji, ale nenašel bych všechny soubory se stejným datem úpravy. To bych pak musel řešit jinak a vyhledávání by se mohlo zpomalit.

Program vrací pouze jednu odpověď na vyhledávání názvem. To muže být někdy problematické, protože dva soubory mohou mít stejné jméno a být v jiných složkách. To se typicky nestává, ale program v tomto případě je schopný najít pouze jeden z těchto souborů. Při vytváření indexu se po nalezení souboru s jménem, které už bylo nalezeno, index přepíše. Lze to vyřešit tak, že si místo jedné hodnotu bude uzel pamatovat list s hodnotami. Tím by se ale zvětšilo místo, které zabírá index, který je už tak dost velký.

Závěr

Na začátku jsem si představoval, že implementuji více způsobů jak vyhledávat soubory. Základní myšlenka byla taková, že to bude jednoduchý skript, který bude na málo řádků a budu muset dodělávat další funkce, aby program nebyl moc krátký. Naopak jsem se základní funkčností programu vyčerpal víc řádek, než jsem myslel, že mi bude stačit. Původně jsem chtěl dodělat i drobné grafické rozhraní pro uživatele, ale tím bych nejspíše už přesáhl zadaný rozsah práce.

S programem jsem spokojený a myslím, že jsem svůj cíl splnil. Jediné, co jsem nedodělal, bylo to, aby se program čas od času zapnul a prohledal disk. Nemyslím si, že by to teď bylo náročné dodělat, když mám celý program hotový jen mi to nepřišlo tak důležité. Navíc lze to zařídit i mimo program. V tomto projektu mi šlo, hlavně o procvičení práce se soubory v pythonu a implementaci nových datových struktur a toho jsem i dosáhl.