Vyhľadávanie v texte 2013: Domáca úloha
15% celkovej známky + nepovinné bonusové otázky
Termín odovzdania: streda 27.3.2013 o 14:50

Cieľom tejto domácej úlohy je implementovať dve rôzne dátové štruktúry a porovnať čas ich výpočtu pre náhodné aj reálne dáta. Môžete použiť ľubovoľný bežný programovací jazyk, napríklad C/C++/C#, Java, Python alebo Pascal. Časti B-D a nepovinné bonusy odovzdajte v papierovej forme na prednáške alebo v kancelárii M163 (ak je zamknuté, vsuňte úlohu popod dvere). Zdrojový kód vašich programov (časti A, C a bonusové otázky) odovzdajte v jednom zozipovanom súbore prostredníctvom systému Moodle. Dbajte na prehľadnosť a čitateľnosť programu a ak máte viac ako jeden súbor, pripojte krátke README vysvetľujúce význam jednotlivých súborov. Neodovzdávajte dátové súbory. Neopisujte kód ani text od spolužiakov ani z internetu alebo iných zdrojov.

Budeme uvažovať abstraktný dátový typ, ktorý udržuje množinu slov a pre každé slovo počítadlo jeho výskytov, pričom umožňuje nasledujúce operácie:

- init(σ): vytvorí prázdnu štruktúru pre použitie abecedy veľkosti σ (táto operácia môže byť implementovaná ako konštruktor určitej triedy)
- add(w): zvýši počítadlo pre slovo w. Ak w ešte nie je v dátovej štruktúre, najskôr ho vloží
- frequent(k): vráti všetky slová, ktoré sa vyskytujú aspoň k krát a počet výskytov každého z týchto slov

Môžete predpokladať, že abeceda pozostáva zo symbolov $\{0,1,\ldots,\sigma-1\}$. Slová môžete reprezentovať napr. ako polia celých čísel.

Časť A, 6 bodov.

Implementujte dve verzie dátovej šturktúry popísanej vyššie s tým istým rozhraním. Prvá implementácia je založená na všeobecnej implementácii asociatívneho poľa (slovníka), ktorý má slová ako kľúče a počítadlá ako hodnoty pre jednotlivé kľúče. Môžete použiť implementáciu poskytnutú v štandardných knižniciach vami zvoleného jazyka (napr. map v knižnici STL pre C++ alebo TreeMap v Jave). Ak váš programovací jazyk neposkytuje takúto dátovú štruktúru, implementujte binárne vyhľadávacie stromy. Táto implementácia bude opakovane porovnávať slová podľa lexikografického usporiadania.

Druhá implementácia používa lexikografické stromy, pričom každý vrchol má smerníky na synov uložené v poli dĺžky σ. Túto dátovú štruktúru implementujte sami, nepoužívajte hotové knižnice.

Operácia frequent(k) môže prehľadať celú dátovú štruktúru a vrátiť slová splňujúce podmienku, nemusíte sa teda snažiť udržiavať štruktúry umožňujúce rýchle nájdenie často sa vyskytujúcich slov.

Použite vaše dve implementácie v programe, ktorý načíta na vstupe postupnosť slov, jedno po druhom ich vloží ich do dátovej štruktúry a na konci vypíše zoznam slov, ktoré sa vyskytovali aspoň k krát a počty ich výskytov. Slová môžete vypísať v ľubovoľnom poradí. Program by mal takisto merať čas potrebný na vkladanie slov do štruktúry (nezahrňujte čas potrebný na načítanie a spracovanie vstupu, volanie operácie frequent(k) a formátovanie výstupu).

Poznámky k implementácii. Na stránke predmetu nájdete kostru programu v jazyku C++, ktorá obsahuje načítanie vstupu a meranie času. Tento kód môžete použiť, ak plánujete pracovať v jazyku C++.

Ak váš program beží veľmi krátko, môže byť ťažké spoľahlivo odmerať čas výpočtu. Preto zopakujte celý algoritmus niekoľko krát a merajte celkový čas. Všetky experimenty spúšťajte na tom istom počítači a za podobných podmienok.

Vstupný súbor môže obsahovať ľubovoľný text. Znaky a-z a A-Z z anglickej abecedy prekódujte na symboly 0..25 vstupnej abecedy, pričom 'a' and 'A' reprezentujú symbol 0, 'b' a 'B' symbol 1, atď. Ľubovoľné iné znaky (medzery, čísla, špeciálne znaky atď) považujte za oddeľovače slov. Budeme predpokladať, že $\sigma \leq 26$.

Okrem hlavného vstupného súboru s textom by Váš program mal dostať (napríklad ako nastavenia na príkazovom riadku) veľkosť abecedy σ, prah k, ktorú implementáciu má použiť a ďalšie potrebné parametre.

Výstupný súbor na prvom riadku obsahuje čas výpočtu vášho programu a iné užitočné štatistiky v ľubovoľnom formáte, aký si zvolíte. Každý z ďalších riadkov má obsahovať jedno z výstupných slov malými písmenami, medzeru a počet výskytov tohto slova.

Príklad: Uvažujme $\sigma = 26$, $k = 2$. Nasledujúci vstupný súbor obsahuje tri výskyty slova ma, dva výskyty slova mama a jeden výskyt slov ema, emu, mamu, sa. Jeden z možných správnych výsledkov je vpravo.

```
Vstup:                          Výstup:

Mama123ma Emu                   time: 1s
Ema_*ma_mamu. Mama sa ma.       mama 2
                                ma 3
```

Časť B, 3 body

Na stránke predmetu nájdete dva súbory, každý obsahujúci prvých 20000 slov z jednej knihy. Vo vašej úlohe pre každý z týchto súborov uveďte, ktoré slová sa vyskytujú aspoň $k = 250$ krát a počet ich výskytov. Tiež uveďte, koľko rôznych slov je v každom súbore ($k = 1$). Uveďte aj čas výpočtu obidvoch implementácií pre každý súbor.

Časť C, 4 body

Napíšte program, ktorý vygeneruje z náhodných slov, každé dĺžky L nad abecedou σ s nezávislými a rovnomerne rozdelenými znakmi. Uvažujte $z = 20000$, $L \in \{4, 10\}$ a $\sigma \in \{2, 26\}$. Pre každú z týchto štyroch kombinácií z, L a σ spustite obidve vaše implementácie na 10 náhodných vstupoch. Spočítajte priemerný čas výpočtu a smerodajnú odchylku (standard deviation) pre každú implementáciu a nastavenie parametrov. Výsledné štatistiky uveďte v prehľadnej tabuľke.

Časť D, 2 body

Okomentujte výsledné časy namerané v častiach B a C. Aké z nich vyvodzujete závery? Napríklad, akú implementáciu by ste odporučili pre ktoré hodnoty parametrov? Správajú sa náhodné

slová podobne ako prirodzený jazyk? Zodpovedajú namerané hodnoty vašim očakávaniam, alebo vás niečim prekvapili? Ako by ste prekvapivé javy zdvôvodnili?

Nepovinné bonusové otázky

Za nasledujúce otázky môžete dostať bonusové body (tieto body sa nezapočítavajú do limitu 10% na bonusové body za aktivitu). Bonusové otázky budú bodované prísnejšie, neočakávajte čiastkové body za neúplné alebo nesprávne riešenia.

Časť E, 3 body

Naimplementujte tretiu verziu dátovej štruktúry, ktorá je tiež založená na lexikografickom strome, ale zoznam detí udržuje v poli, ktorého veľkosť sa dynamicky zväčšuje podľa počtu detí (môžete použiť napr. dátovú štruktúru vector). Smerníky na deti udržujte v poli utriedené podľa abecedy. V operácii add na vyhľadanie príslušného dieťaťa použite binárne vyhľadávanie. Spustite testy z časti B a C a okomentujte výsledky.

Časť F, 3 body

V časti B a C nemerajte iba čas výpočtu, ale aj pamäť. Popíšte, ako ste to robili, uveďte namerané výsledky a okomentuje ich.

Časť G, 5 bodov

Spravte ďalšie experimenty na náhodných alebo reálnych dátach a pokúste sa objaviť nejaké zaujímavé trendy. Napríklad môžete experimentovať so zložitejšími modelmi náhodných slov, porovnávať rôzne prirodzené jazyky, preskúmať viac hodnôt parametrov L, z a σ, skúsiť meniť nastavenia kompilátora alebo robiť malé zmeny v implementácii. Navrhnite vaše experimenty tak, aby odpovedali nejakú netriválnu otázku. Zdôvodnite návrh experimentov a rozdiskutujte namerané hodnoty. Pri bodovaní je dôležitejší vhodný návrh experimentov a vyvodené závery, než len počet experimentov, ktoré ste spravili.

Časť H, 4 body.

Popíšte ako rozšíriť lexikografický strom z časti A tak, aby v operácii frequent(k) nemusel prehľadávať všetky vrcholy. Podrobne popíšte vašu dátovú štruktúru, ale nemusíte ju implementovať. Mala by spĺňať nasledujúce požiadavky:

- Pre každý vrchol lexikografického stromu pridá pamäť najviac veľkosti $O(1)$ a ďalšiu celkovú pamäť $O(1)$, bez ohľadu na počet výskytov slov.
- Čas výpočtu funkcie add(w) zvýši najviac o $O(1)$.
- Podporuje prechádzanie cez všetky slová v poradí podľa klesajúceho počtu výskytov, pričom čas na navštívenie jedného slova bude iba $O(1)$.

Môžete využiť fakt, že počítadlo počtu výskytov je pre každé slovo na začiatku nula a zvyšuje sa iba operáciou add(w).