Tema 4. Dicționar T9

Termen de predare: 23.05.2014, 23:55 - **deadline hard**.
Tema se va trimite pe vmchecker.

Obiective: Dupa realizarea acestei teme de casa studentul va fi capabil sa:

- lucreze cu structuri de date avansate.
- îmbine mai multe structuri de date pentru rezolvarea unei probleme.
- să aleagă structuri de date potrivite pentru a rezolva optim o problemă

Cerintă

Tocmai ați fost angajați pentru de o firmă ce produce telefoane mobile pentru a scrie un software de scriere a mesajelor text. Tastatura telefoanelor produse este una standard, existând o corespondență una-la-mai-multe între cifra corespunzătoare unei taste și literele asociate:

Valoarea numerica a tastei	Litere asociate
2	a b c
3	d e f
4	g h i
5	j k l
6	m n o
7	p q r s
8	t u v
9	w x y z

Atunci când utilizatorul scrie un mesaj softul primeste doar cifra corespunzătoare tastei apăsate. Spre exemplu, dacă s-a primit șirul de taste `2247` acesta poate semnifica cuvântul `cais`, dar și cuvântul `acip`(bineînțeles și încă multe altele). Pentru a ști ce cuvânt să se returneze softul menține o listă cu toate cuvintele posibile și cu numărul de apariții ale acestora în mesajele precedente. Atunci când se primește un șir de taste apăsate, se caută cuvântul cu cele mai multe apariții ce se potrivește cu șirul primit. Pentru a permite utilizatorilor să folosească cuvinte și cu un număr mai mic de apariții precedente, s-a introdus folosirea caracterului `*`. Dacă imediat după șirul de taste apăsate apare un șir de forma `*x`, unde x reprezintă un număr, atunci se dorește afișarea celui de-al `x+1`-lea cuvânt în ordinea descrescătoare a aparițiilor ce se potrivește cu șirul de taste primit.

Spre exemplu, dacă lista de cuvinte și a numărului de apariții a softului este:

```
defra 4
feepc 10
pimu 3
rgot 2
rhou 1
pgnt 1
```

atunci șirul `33372` va genera `feepc`, `33372*1` va genera `defra`, `7468` va genera `pimu`, `7468*2` va genera `pgnt`, `7468*4` va genera `pimu`.

Se observă că:

- în caz de număr de apariții egale se afișează șirul mai mic din punct de vedere lexicografic
- daca dupa `*` apare `x` și avem mai puțin de `x+1` cuvinte ce se potrivesc cu șirul primit, atunci afișarea se reia de la început, cu cuvântul cu numărul cel mai mare de apariții.

Deoarece se dorește ca softul să fie personalizat în funcție de preferințele utilizatorilor, atunci când un cuvânt este folosit softul poate să rețină acest lucru și să incrementeze numărul de apariții asociate cu acel cuvânt. Spre exemplu, dacă avem următoarea listă:

```
dilo 2
egln 3
```

primirea șirului `3456*1` va rezulta afisarea cuvantului `dilo` si incrementarea numarului de aparitii la 3, iar dupa aceea primirea sirului `3456` va duce tot la afisarea `dilo` (ambele cuvinte au acelasi numar de aparitii, 3, dar`dilo` este mai mic din punct de vedere lexicografic). Datoria voastră este să implementați eficient un soft care, pentru un șir de taste apăsate să returneze cuvântul corespunzător.

Detalii de implementare si punctare

Lista de cuvinte de care dispune softul este una foarte mare, în care nu ne permitem să facem căutari pentru fiecare șir introdus. Tocmai de aceea va trebui folosită o structură de date denumită `trie`, care permite adăugarea și căutarea unui cuvânt dintr-o mulțime de cuvinte intr-un timp egal cu `O(lungime_cuvant)`. Dat fiind că folosirea unui cuvânt determină incrementarea numărului de apariții ale acestuia, trebuie să folosiți o structură de `treap` în care să determinați în mod optim care este al `x`-lea cuvânt în ordinea descrescătoare a numărului de apariții și să și puteți modifica acest număr. Pentru mai multe detalii, putei consulta lista de resurse utile.

Astfel, fiecare nod al structurii trie corespunde unui posibil șir de cifre, dar cum unui șir de cifre îi pot corespunde mai multe cuvinte din dicționar, va trebui ca în fiecare nod să rețineți un treap pentru a gestiona aceste cuvinte și numărul lor de apariții. Pentru evaluare vor exista în total 9 fișiere de test, a câte 10 puncte fiecare, încă 10 puncte fiind acordate pentru coding-style, comentarii și README.

Format fișiere de intrare

Fișierul de intrare din care veți citi atât lista de cuvinte, cât și șirurile venite de la tastarură se va numi`date.in` și va avea următorul format:

- a doua linia va conține `N`, dimensiunea listei de cuvinte avută la dispoziție
- următoarele `N` linii vor conține fiecare un cuvânt (format din litere mici ale alfabetului englez) urmat de un spațiu și de numărul de apariții ale acestuia
- următoarea linia va conține `M`, numărul de șiruri venite de la tastarură
- următoarele `M` linii vor conține un șir venit de la tastatură format din cifre de la `2` la `9` și un posibil caracter `*` urmat de un numar.

Fișierul de ieșire se va numi `date.out` și va conține `M` linii, pe cea de-a `i`-a linie aflându-se răspunsul pentru al`i`-lea șir venit de la tastatură.

Exmple

date.in

```
5
tara 4
varb 3
ucsu 3
aaaa 10
bbbb 11
8
8272
8272*2
8272*2
8272*1
2222*1
2222*1
2222
2222*2
```

date.out

```
tara
tara
tara
varb
aaaa
bbbb
bbbb
bbbb
```

date.in

```
4
aad 3
bbf 10
aabeda 13
ace 4
5
223
223*2
222332
223*1
223*5
```

date.out

```
bbf
aad
aabeda
aad
ace
```

Restricții și precizări suplimentare

Pentru datele de intrare:

- $1 \leq N, M \leq 100.000$
- lungimea unui cuvânt este de cel mult 20 de caractere
- lungimea unui sir de intrare este de cel mult 20 de caractere (excluzând $*$ și numărul de îi urmează). Numărul x de după $*$ este mai mic de 10^9
- numărul de apariții ale unui cuvânt este mai mic decât 10^9

Pentru execuția soluției voastre pe un test există o limita de timp de 6 secunde. Dacă execuția dureaza mai mult de atât, aceasta va fi întreruptă și veți primi 0 puncte pe testul respectiv.

Arhiva cu rezolvarea temei trebuie sa fie .zip și să conțină:

- Fișierul/fișierele sursă
- Fișierul Makefile
- Fișierul README

Fișierul pentru make trebuie denumit **obligatoriu** Makefile și trebuie să conțină următoarele reguli:

- build, care va compila sursele și va obține executabilul
- run, care va rula executabilul
- clean, care va șterge executabilul generate.

Atenție NU este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).

Atenție Scopul temei este familiarizarea cu structurile de date prezentate mai sus. **NU** se vor puncta solutii ce folosesc alte structuri de date/algoritmi (hashing, cautare binara, STL, etc.). Este permisă (și încurajată) folosirea clasei `string` din STL.
O arhivă cu testele folosite pentru evaluare o puteți găsi aici. Din acestă arhivă, testele 3-11 vor fi folosite pentru evaluarea temei, primele două având rolul de a vă ajuta la debug. Trecerea testului 4 nu este condiționată doar de un răspuns corect, ci și de lipsa leak-urilor de memorie.