Rečnik

Problem ima tri celine od kojih se sastoji:

1. PRONALAŽENJE reči u rečniku,
2. DODAVANJE reči u rečnik,
3. PREBROJAVANJE manjih reči od tražene.

BRUTE FORCE

Memorijsta složenost N * (max duzina stringa).
Vremenska složenost O(Prosečna dužina stringa * N^2).

OPIS ALGORITMA:

Za svaki red ulaza radimo:
Isecamo reč iz komande i prebacujemo je u sva velika slova.
Ako je komanda LESS,
onda ako reč nije u rečniku
onda je DODAJEMO u rečnik
inače preskačemo dodavanje
inače ako reč nije u rečniku
onda ispisujemo "no such word"
inače PREBROJAVAMO koliko ima manjih od zadate reči u rečniku i ispisujemo to.

Činjenica da su reči u rečniku leksikografski poslagane može da prevari i da Vas navede da sortirate rečnik nakon dodavanja reči. Nema potrebe.

PRONALAŽENJE reči se može odraditi u jednom prolazu kroz rečnik. To je O(N).
DODAVANJE reči se može odraditi u konstantnom vremenu. To je O(1). Reč dodajemo na kraj rečnika.
PREBROJAVANJE reči se može odraditi u jednom prolazu kroz rečnik. To je O(N).

(kod komande LESS prva i treća operacija se mogu u jednom prolazu odraditi)

Po opisanom algoritmu imamo O(N) * (O(N) + O(1) ili O(N)) pa je kompletna složenost je $O(N^2)$. Sve ovo pomnožimo sa prosečnom dužinom stringa što je oko 20.

Ovakvo rešenje donosi 50 bodova.

PRISTUP s kumulativnim tabelama (BIT Binary Indexed Tree)

Memorijsta složenost 2 * N * (max duzina stringa) .
Vremenska složenost O(Prosečna dužina stringa * N*log(N)) za pripremu a O(N*log(N)) za obradu.

OPIS ALGORITMA:

Učitamo sve u niz komandi.
Isecamo reč iz komande i prebacujemo je u sva velika slova i ubacimo u rečnik one koje su uz ADD.
Sortiramo reči u rečniku po leksikografskom poretku zajedno sa mestima pojavljivanja na ulazu.
Izbacujemo duplicate iz rečnika, ostavljajući one koje su se prve pojavile.
U nizu komandi dodajemo mesto reči nakon sortiranja.

Za svaki red ulaza sada radimo:
Ako je komanda LESS,
onda ako reč nije u BIT (njeno mesto u sortiranom rečniku)
onda DODAJEMO u BIT jedinicu na mesto koje ta reč ima u sortiranom nizu.
inače preskačemo dodavanje
inače ako reč nije u rečniku
onda ispisujemo "no such word"
inače PREBROJAVAMO koliko ima manjih u BIT od zadane i ispisujemo to.

SORTIRANJE nekim brzim sortom O(N*LogN)

PRONALAŽENJE reči se može odraditi u dve sum operacije u BIT. To je O(2*log(N)).
DODAVANJE reči se može odraditi u jednoj add operaciji u BIT. To je O(log(N)).
PREBROJAVANJE reči se može odraditi u dve sum operacije u BIT. To je O(2*log(N)).

Pošto ima N komandi drugi deo algoritma je složenosti O(N*2*log(N)),

Sve u svemu O(2*N*2*log(N))). ==➔ O(N*log(N)).

Naravno opet sve ovo * 20 što je prosečna dužina u najgorem slučaju.
(Test primeri su podešeni da ovaj pristup prodje za 2 sec sa 100 000 članova bez hash funkcija.)

Ovakvo rešenje je nosilo 100 bodova.

Link za BIT na top coderu

U prilogu su kodovi Ivana Stošića iz Niša u C++u i Duška Obradovića iz Sombora u PASCAL-u.

PRISTUP s TRIE strukturom

Uz ograničenja u zadatku i na ovaj način je bilo moguće osvojiti 100 bodova.

Link za TRIE na Wikipediji

U prilogu je kod Demjana Grubića iz Novog Sada u C++u.